Electron Beam Dynamics for the ISIS Bremsstrahlung Beam Generation System

by

Robert E Block

SB, Nuclear Science and Engineering (2010)
Massachusetts Institute of Technology

Submitted to the Department of Nuclear Science and Engineering
in partial fulfillment of the requirements for the degree of

Master's of Science in Nuclear Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2011

Author .... [illegible] ....
Department of Nuclear Science and Engineering
May 13, 2011

Certified by .... [illegible] ....
Timothy A Antaya
Group Leader, MIT PSFC Technology & Engineering
Thesis Supervisor

Certified by .... [illegible] ....
Jeffrey P Freidberg
Korea Electric Power Professor of Nuclear Science and Engineering
Thesis Reader

Accepted by .... [illegible] ....
Mujid S Kazimi
TEPCO Professor of Nuclear Engineering
Chair, Department Committee for Graduate Students

Electron Beam Dynamics for the ISIS Bremsstrahlung Beam Generation System

by

Robert E Block

Submitted to the Department of Nuclear Science and Engineering
on May 13, 2011, in partial fulfillment of the
requirements for the degree of
Master's of Science in Nuclear Science and Engineering

Abstract

An electron beam transport system was designed for use in the Bremsstrahlung Beam Generation System of the Integrated Stand-off Inspection System (ISIS). The purpose of this electron transport system was to provide for electron beam diagnostics and energy selection while also positioning the electron beam on a target down range.

The transport system and its component magnets were designed using the *TRANSPORT*, *Poisson*, and *Opera 3D* codes, as well as several custom *Python* scripts. By implementing several methods in each part of the design process, it was possible to design the electron transport system to the exact specifications of the ISIS electron beam. This careful and iterative design process was documented in such a way to facilitate future beam transport design both at the MIT Plasma Science and Fusion center and elsewhere.

This design process resulted in a beam transport system composed of three iron-dominated resistive-coil electromagnets. The system was designed for beam momentum up to 60 MeV/c and emittance of order 20 mm-mrad. Through magnetic field simulation and beam transport in 3D, a 1D matrix code which tracks individual particles was developed. This code agreed with more detailed beam calculations and should allow for rapid beam simulation during system testing and operation.

Thesis Supervisor: Timothy A Antaya
Title: Group Leader, MIT PSFC Technology & Engineering

Thesis Reader: Jeffrey P Freidberg
Title: Korea Electric Power Professor of Nuclear Science and Engineering

Acknowledgments

I would like to begin by expressing my thanks to my advisor, Dr Timothy Antaya. His guidance and feedback inspired me to explore new topics and to constantly seek ways to improve my work. I feel privileged to have been given the opportunity to learn from such a talented and caring individual. I also gratefully acknowledge my thesis reader, Professor Freidberg, whom should be congratulated on his retirement.

In a thesis project which involves engineering as much as it does theory, having a strong group of mechanical and electrical engineers to support my work was essential. I thank Valery Fishman, Tony Bistany, and Philip Michael for transforming the theoretical BBGS design into a functional system. These three engineers share countless hours spent on Solidworks design and material procurement for the BBGS. I also thank Rick Lations for his involvement in the fabrication of the system.

My research could not have been completed had I not been introduced to the field of nuclear non-proliferation by Dr Michael Hynes and Dr Richard Lanza. I am very grateful to them for stimulating my interest in the subject and for introducing me to Dr Antaya and his work. I also appreciate the feedback, calculations, and discussions I shared with Dr Brandon Blackburn and Dr Erik Johnson, each graduates of the MIT NSE department and currently working as research scientists at Raytheon.

I will be forever appreciative of CAPT Curtis Stevens and Charles Frantz for their strong encouragement and support of my graduate education at MIT. It is certain that without their support I could not have pursued my graduate education.

Finally, I thank my parents Ed and Judy, my wife Meghan, and all of my family for their unending support, encouragement, and love throughout both my undergraduate and graduate years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In a world where the number of nuclear weapons states is increasing and the sheer number of nuclear warheads prevents the international community from accounting for all of these warheads, the use of nuclear weapons by rogue states or terrorist organizations has become an important concern for US national security(Bunn and Wier, 2006; Don Daigler, 2010; Hecker, 2006; Jenkins, 2006). Preventing a clandestine nuclear attack on the United States will involve improving current international safeguards against the theft of nuclear materials and nuclear weapons, but as this happens it will also be important to prevent any materials which have already been stolen from being employed against the United States (Hecker, 2006; Kramer, 2008). To prevent the movement of nuclear weapons and materials across its borders, the US will need improved detection systems.

The Defense Threat Reduction Agency (DTRA) is currently seeking ways to detect concealed strategic nuclear material (SNM) from a distance up to 100 m. The Integrated Standoff Interrogation System (ISIS) has been designed in order to meet this challenge. Since the radiation which results from the nuclear decay of most SNM can be easily shielded from detection, the ISIS design uses an active detection system, similar to those outlined in B. Blackburn et al. (2006, 2007); James L Jones et al. (2007). In an active system, a beam of photons or particles is used to induce nuclear fission in the target material. Detection of the material occurs when delayed photons and neutrons, the signatures of nuclear fission, are detected by the system.

1.2 ISIS System Overview

The basic concept of the ISIS SNM detection system is to use a high energy photon beam in order to induce photonuclear fission in concealed special nuclear material. Once this fission is induced, detectors will be employed to collect signals generated by the fission reaction. As outlined in figure 1-1 below, the photon beam is generated by stopping a high-energy electron beam. A linear accelerator (LINAC) is used to accelerate electrons from 10 to 60 MeV. These electrons are then transported to an assembly of converter plates where the electrons are stopped and their kinetic energy is converted through collisions into heat in the plates and Bremsstrahlung radiation. Bremsstrahlung radiation is a term which describes the radiation generated by the acceleration of charged particles as they pass by atoms. A thick collimator after the converter plates shapes the secondary photon (Bremsstrahlung) beam for long distance transport. The portion of the total system from the end of the LINAC through the converter/collimator assembly is known as the Bremsstrahlung Beam Generation System.

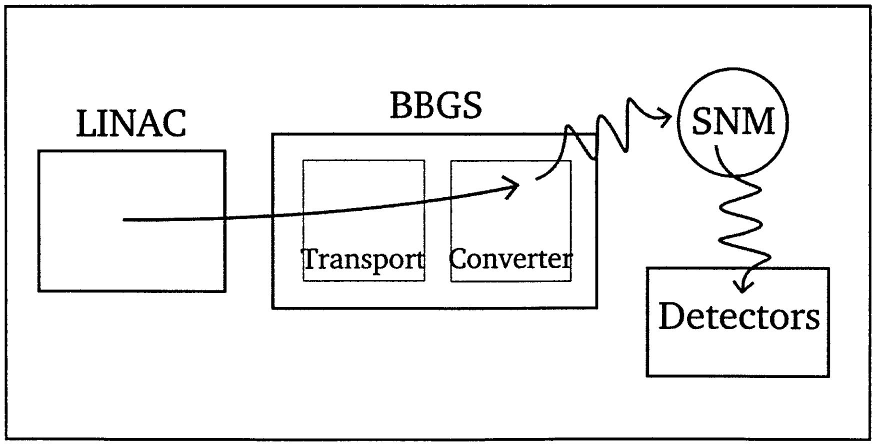


Figure 1-1: Basic system diagram of ISIS.

In order for the ISIS Bremsstrahlung Beam Generation System (BBGS) to succeed, it requires a robust and compact electron beam transport system to deliver the electron beam at precise energies from a LINAC to the converter/collimator. The focus of this master's thesis project was to design, and time permitting, assemble and verify an electron beam transport system for ISIS which meets the design requirements set by Raytheon project managers. This was an iterative design process in which each design iteration attempted to meet the system requirements while minimizing the size and power requirements of the components.

1.3 Thesis Objectives

The primary objective of this thesis project was as follows:

> To design an electron transport system for the ISIS BBGS which utilizes the full electron beam and obtains a final photo spot size no greater than 1 m at a distance of 100 m from the collimator, while also providing photon beam steering capability of $\pm 5°(\pm 15°)$ vertical(horizontal).

Secondary objectives included:

- Assemble the electron transport system for on-site testing and verification.
- Verify the transport system design using an equivalent momentum charged particle beam available at MIT facilities.

In addition to these educational objectives, it was important to provide detailed documentation of the electron transport system's physical specifications in order to facilitate its incorporation into the ISIS system design.

1.4 Report Organization

Chapter 2 provides an overview of Bremsstrahlung Beam Generation System including final design specifications and layout.
Chapter 3 includes a detailed review of the system-level beam transport design, complete with both 1D and full 3D beam simulations.

Chapter 4 details the magnetic and physical design of each component magnet.

Chapter 5 describes the calculations which supported the power system and water cooling design for the BBGS.

Chapter 6 discusses a work plan to quantitatively characterize the BBGS system performance using an electron cyclotron resonance ion source.

Chapter 7 presents conclusions of this work as well as insights for future work in the field.

Chapter 2

ISIS BBGS Overview

This chapter provides an overview of the final design of the Bremsstrahlung Beam Generation System for ISIS. This design is a result of many iterations and detailed analyses which will be described in more depth in the chapters to follow. The purpose of this chapter is to acquaint the reader with both the components of the BBGS and the constraints which guided the design process.

2.1 Design Constraints

Unlike many beam transport systems, the ISIS BBGS is a part of a mobile and deployable system which is assembled in a conventional ISO container. The requirement for portability played an important role in the design process since it put physical constraints on the size and weight of the BBGS.

Figure 2-1 shows a simplified diagram of the spaced allotted for the BBGS. The BBGS must fit into the 1.5 by 2.0 by 1.5 m area while also maintaining weight, power, and cooling requirements which can be supported by the portable systems within the ISO container. Considerations of these constraints were important to the BBGS design as they limited the size and number of beam-line components available to shape the electron beam. For example, designs for a focusing solenoid as well as a beam alignment magnet were removed from the BBGS due to concerns about system size.

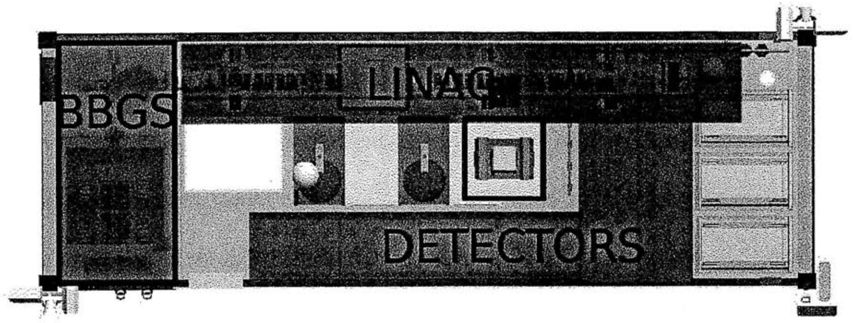


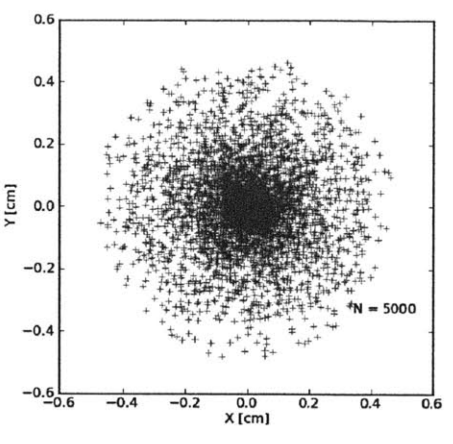
Figure 2-1: ISO container for ISIS with the BBGS area highlighted.

Although the BBGS is responsible for shaping the final electron and photon beam for ISIS, the input electron beam does not fall under the domain of the BBGS design. The electron beam is formed by a linear accelerator designed by Advanced Energy Systems (AES), and hence the electron beam was a constraint which drove much of the BBGS layout and component design. To aid in the BBGS design, AES provided the results of electron beam simulations for three selected energies. These results were provided as electron position and momentum data evaluated at the exit of the LINAC. The operating parameters and an example beam cross section as measured at the exit of the LINAC are described in table 2.1 and figure 2-2, respectively.

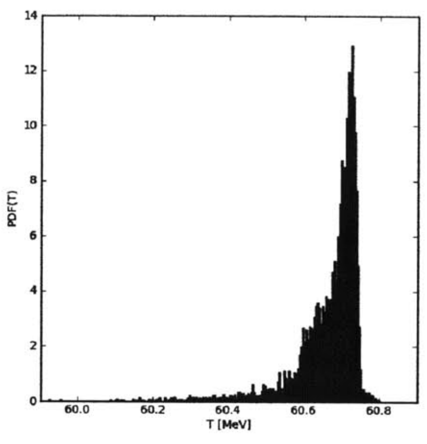
Table 2.1: Beam Properties of AES Parmela Simulated Beam. Dimensions in cm, angles in mrad.

$\langle T \rangle$ MeV	$\sigma_T(\%)$	x_{rms}	x_{max}	y_{rms}	y_{max}	$\% > rms$	x'_{rms}	y'_{rms}
60.66	0.21	0.136	0.489	0.137	0.487	43.3	0.483	0.486
30.57	0.23	0.159	0.510	0.161	0.511	46.9	0.669	0.675
6.178	0.38	0.167	0.509	0.168	0.511	51.7	0.420	0.419

The percentage of the beam outside the *rms* radius coupled with the relatively large variance in beam energy (σ_T^2) presented a challenge for shaping and focusing the electron beam onto the converter plate assembly. This task needed to be completed with minimal beam loss so as to reduce the amount of unshielded radiation generated within the container. The negative effects of excess radiation in the container could include degrading equipment stability and also limiting maintenance access.



(a) Cross Section



(b) Energy Histogram

Figure 2-2: Beam cross section and energy histogram for the 60.7 MeV beam from AES Parmela simulations of the LINAC.

2.2 System Specifications

Diagrams of the final design for the Bremsstrahlung Beam Generation System are provided in figures 2-3 and 2-4. The components labeled are as follows: beam defining slits (1), straight-through beam port (3), 90° dipole magnet (2), steering magnets (4), converter assembly (5), and photon collimator (6). The detailed designs for each system component are elaborated in the following section. However, a basic description of the purpose for each component is described below.

The object (1a) and image (1b) slits are included in the BBGS to provide for diagnostic operation. By closing each slit aperture to a fine hole, it will be possible to perform two main functions: (a) reduce the beam transverse divergence and (b) calibrate the dipole to the appropriate beam energy.

It was important to include a pass-through (3) in the dipole magnet for more than one reason. The pass through will primarily be used as a diagnostic port for the electron beam directly from the linear accelerator. In addition, in the event of an emergency during full-power operation, the beam will be fully stopped in a beam dump located at the end of the pass-through. This is particularly important should the power supply for the dipole magnet malfunction.

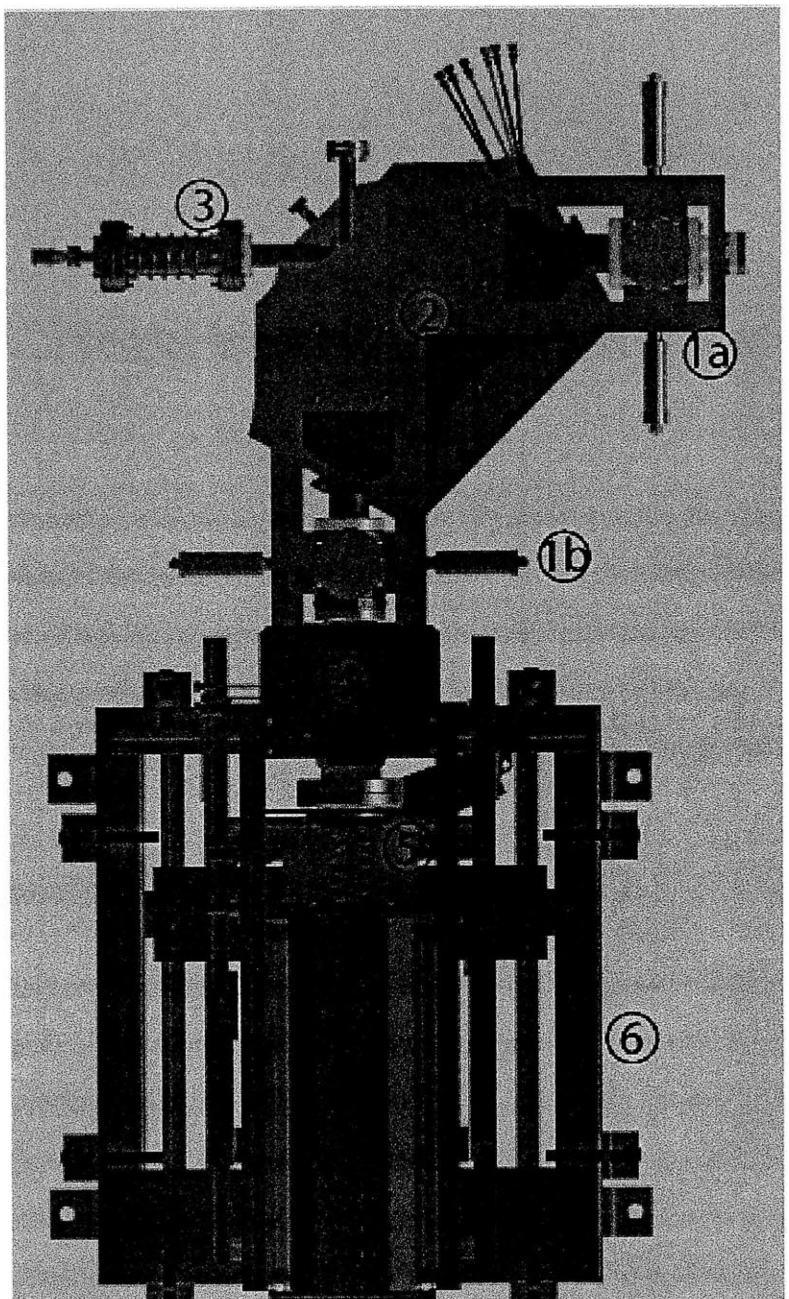


Figure 2-3: ISIS BBGS system layout courtesy T. Bistany. 1a(b): beam object(image) slits, 2: 90° dipole, 3: straight-through beam port, 4: steering magnets, 5: converter assembly, 6: photon collimator

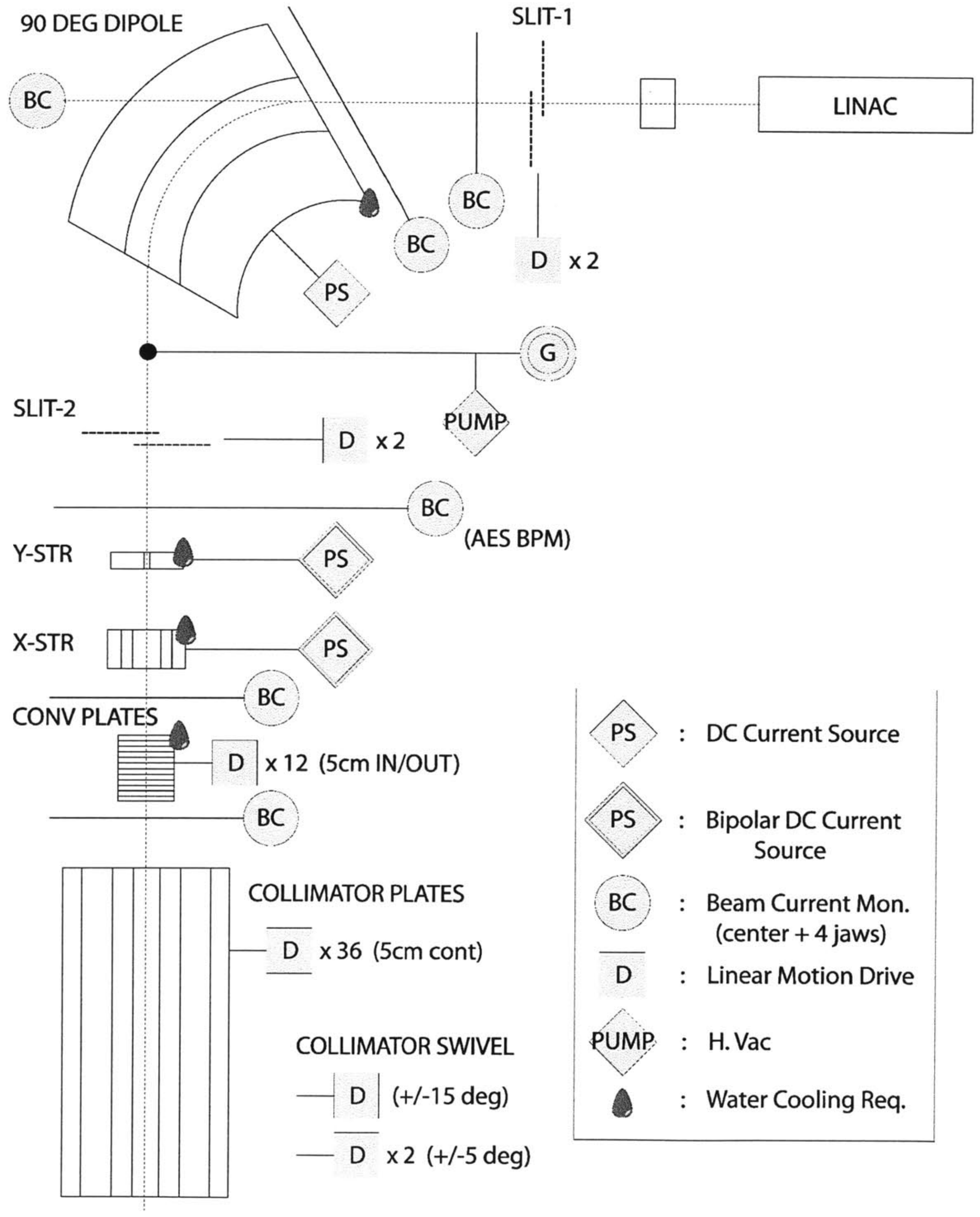


Figure 2-4: ISIS BBGS system instrumentation and controls schematic diagram.

The 90° dipole (2) was included in the system based on lessons learned from a previous active interrogation experiment to ISIS where the beam energy was not well known (James L Jones et al., 2007). The dipole combined with the closed beam slits (to fix the bending radius) sets the momentum of transmitted particles, which allows for electron beam energy selection. The dipole also provides both radial and axial beam focusing.

Independent vertical and horizontal steering magnets (4) were included in the design in order to meet the requirements for $\pm 5°$ vertical and $\pm 15°$ horizontal steering of the secondary photon beam. Since the photons cannot be directed with magnets, the electrons are steered before they collide with the converter plates. The secondary photon beam which is generated then propagates within a conical envelope in the direction of the primary beam with an opening half-angle $1/\gamma$, where γ is the relativistic factor of the electrons. When integrated with the ISIS controls system, these magnets will allow for target tracking.

The converter plate assembly (5) was designed to stop the full energy 60 MeV electron beam. The modular design allows for spent plates to be replaced or for the operation of the beam with a variable depth of stopping material in the beam-line.

Finally, the collimator (6) was included in order to maintain the desired 1 m photon beam spot size at a target which lies 100 m from the ISIS container. The collimator was designed with an adjustable aperture in order to provide some measure of tunable photon beam shape, should conditions deem that desirable.

2.3 Description of BBGS Components

2.3.1 Object and Image Beam Slits

The beam slit system was incorporated into the BBGS design in order to make a point beam at low current for diagnostics and calibration of the system. This point beam should transport free of aberrations and have low energy spread. A diagram of the actuator and plates is shown in figure 2-5. The slits are each built from two MDC

660006 electronic actuators and 304 stainless steel plates. The plates have a 4.93 cm diameter circular cut such that when the actuators are closed, a variable sized hole is left for the beam to pass through.

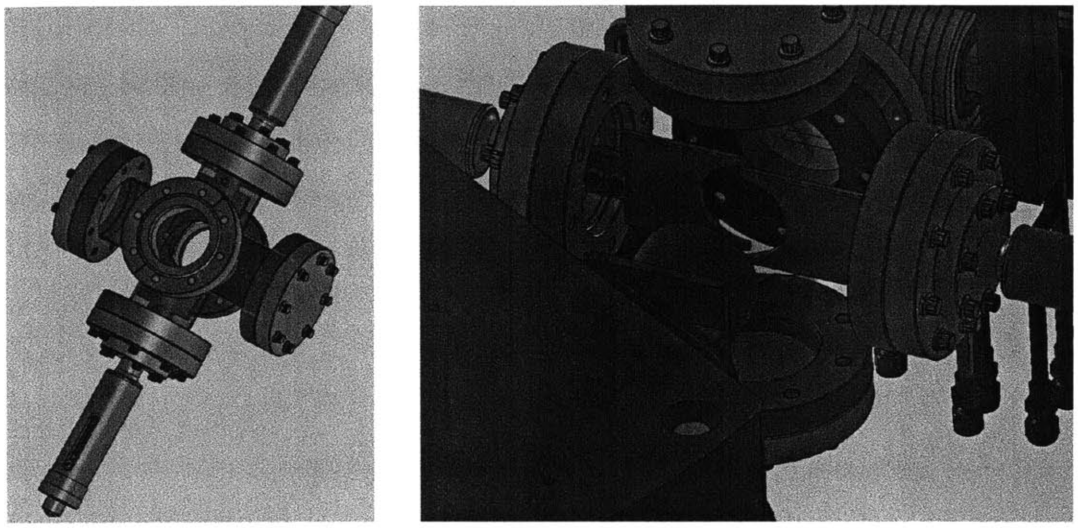


Figure 2-5: 3D view of actuators and plates used in the beam slit system.

Since the purpose of the slit system is to ensure that the dipole is calibrated to point the electron beam on target, it will be important to understand the sensitivity of beam alignment to the slit size. A detailed analysis is carried out in section 3.4, but a more simplified approach was used to estimate basic feasibility. Figure 2-6 shows a simplified diagram of the geometry used to estimate how small the object and image slits will need to be closed to achieve a given tolerance in beam position at the converter assembly.

In the diagram, f is the focal distance of the dipole, d_2 is the half-width of the image slit, and x_f is uncertainty in the final position of the beam. The function describing the focal distance of a dipole magnet can be found in Livingood (1969) to be:

$$f = \frac{r_0(\cos\phi + t_1 \sin\phi)}{\sin\phi - (t_1 + t_2)\cos\phi - t_1 t_2 \sin\phi} \tag{2.1}$$

where r_0 is the magnet bending radius, ϕ is the angular extent of the magnet, and t_1 and t_2 are the tangents of the entrance and exit angles, respectively. The edge angles are the angles the magnet edge makes with respect to the perfect angular sector symmetry.

Using a bending radius of 28.75 cm and identical edge angles, we find the focal distance to be a positive function of edge angle. From the diagram, when the focal distance is exactly 20.5 cm (edge angles of 27.5°), it is clear that it will be impossible to use the beam slits to control the beam position at the converter plates. Two equations can be used to find the ratio between the image slit size, d_2, and the beam position uncertainty, x_f. For focal distances less than 20.5 cm:

$$\frac{x_f}{d_2} = \frac{32.0}{20.5 - f} \tag{2.2}$$

And for focal distances greater than 20.5 and less than 32.0:

$$\frac{x_f}{d_2} = \frac{32.0 - f}{f - 20.5} \tag{2.3}$$

As an example calculation, assume that the image slit has a minimum aperture of approximately 0.05 cm. To obtain resolution in the position of the beam at the target within ±0.2 cm, the focal distance must be less than 16.5 or greater than 21.8 cm. This type of calculation, though simple, was very important to keep in mind throughout the design process. Since iterations brought changes to the bending radius, slit placement, and desired beam size, it was always necessary to go back and check the most fundamental calculations before proceeding with design changes.

That these slits do not have fine control of the image spot size and displacement is no surprise, since the slits were designed primarily to cut the angular divergence of the beam and not to create a fine spot size at the converter plates. It was the dipole which was designed to control image size, as will become clear in what follows. In any case, it was important that the slits not be located directly at the dipole focal point.

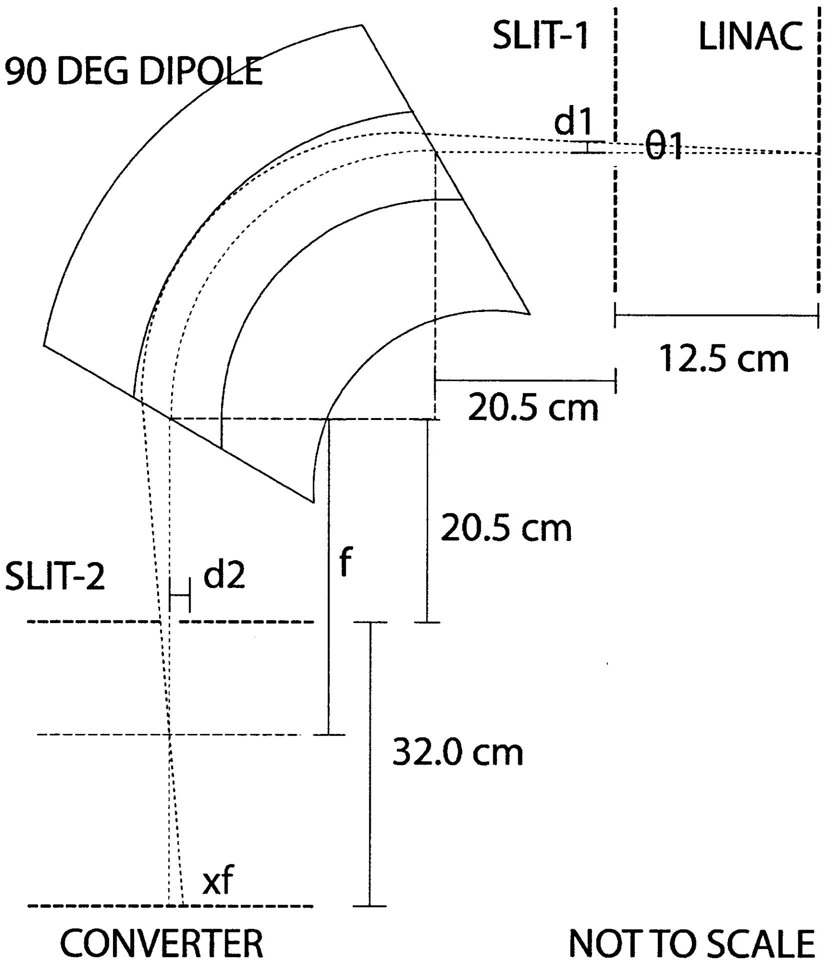


Figure 2-6: Variables used to estimate the effects of slit size on final beam position.

2.3.2 90° Dipole Magnet

The 90° dipole magnet is the central component of the BBGS. The dipole steers the electron beam to the converter assembly and it serves as a tool for beam diagnostics and calibration. The dipole also provides radial and axial[1]beam focusing to ensure the proper beam size at the converter plates. A schematic diagram of the dipole magnet is shown in figure 2-7.

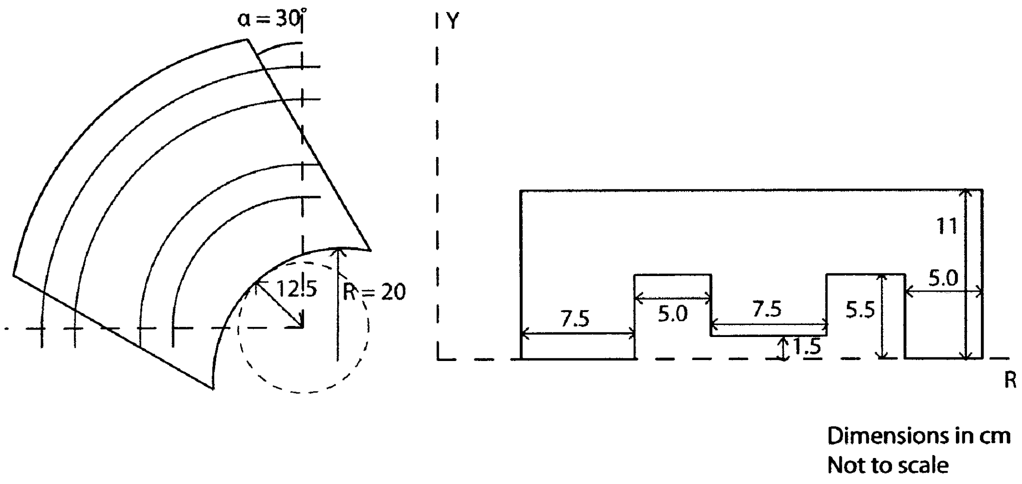


Figure 2-7: Dipole magnet layout and important parameters. Left shows a top-down view and right shows a cross-section which splits the magnet through the middle. Right image has mirror symmetry about R axis.

The BBGS dipole has a bending radius r_o = 28.75 cm with final entrance and exit edge angles of 30°. These edge angles provide axial focusing while the dipole itself provides radial focusing. The beam aperture half-height and half-width are 1.5 cm and 3.5 cm, respectively. These dimensions were optimized to minimize power requirements while leaving a significant tolerance for inconsistencies in beam size.

The coils of the dipole are wound with 6.35 mm square copper conductor and are water cooled through a 3.15 mm bore, as shown in figure 2-8. The conductor was sized to optimize power requirements, water cooling, and peak current density.

[1]The terms axial and radial refer to the cylindrical geometry of a steering magnet. The $\hat{\theta}$ direction points in the beam direction, $\hat{\mathbf{r}}$ parallel to the direction of steering (acceleration), and $\hat{\mathbf{z}}$ mutually perpendicular to both the beam propagation direction and steering direction. Radial: $\hat{\mathbf{r}}$, axial: $\hat{\mathbf{z}}$.

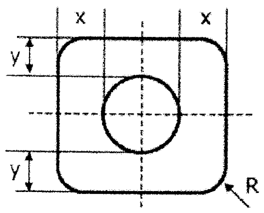


Figure 2-8: Conductor diagram from *Luvata*. The BBGS uses conductor 8417 with $R = 1.0, x = y = 1.6, OD = 6.35, ID = 3.15$ mm.

2.3.3 Beam Steering Magnets

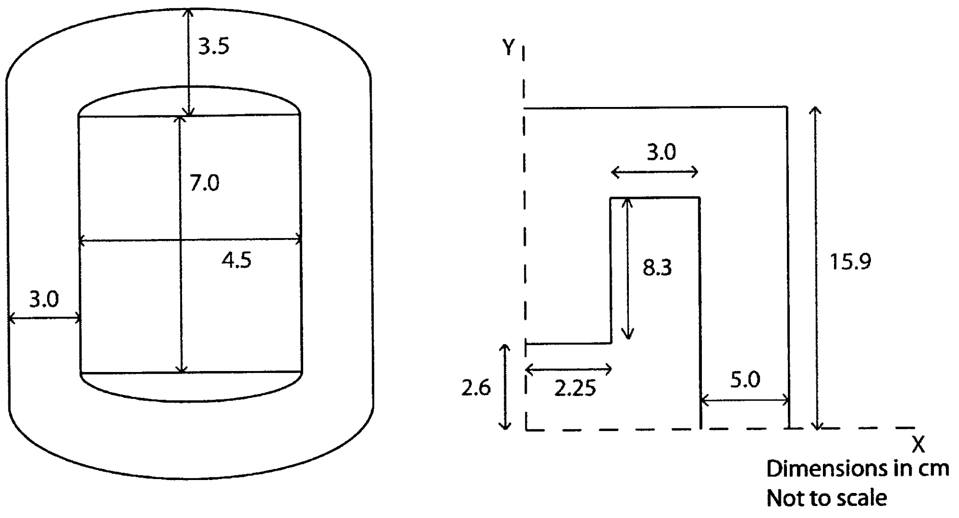
Schematic diagrams of the horizontal (x) and vertical (y) steering magnets are provided in figure 2-9. The purpose of the steering magnets is to align the electron beam with the converter plates for a horizontal/vertical displacement of $\pm 15/5°$. Since these magnets must steer over a relatively short distance, they require high current density and use the same hollow copper conductor as the BBGS dipole.

The apertures of the steering magnets are designed to accept the full beam with a small tolerance for inconsistencies between the actual electron beam and the simulated beam from AES. A discussion of this optimization is provided in chapter 4.

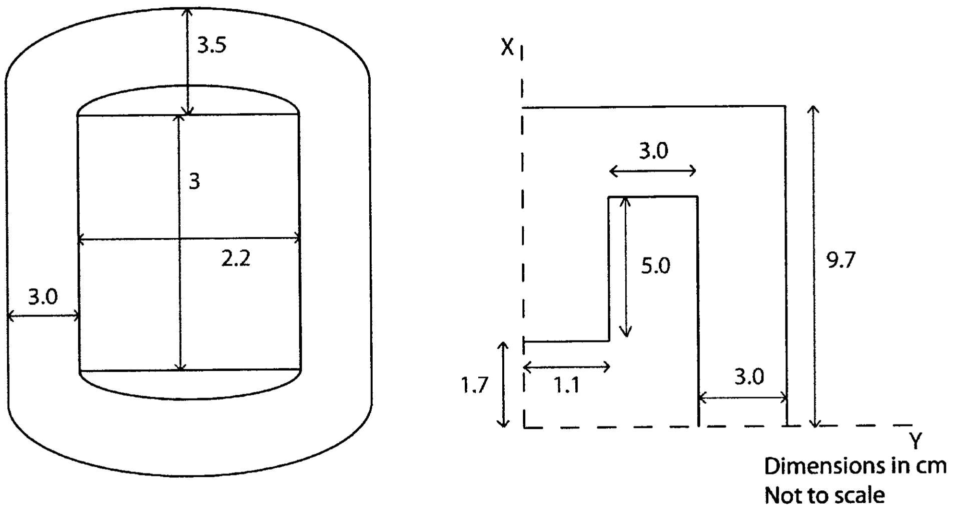
2.3.4 Converter and Collimator Assembly

The physical designs for the converter plate and collimator assemblies were completed primarily by PSFC mechanical engineer V. Fishman. Included in figure 2-10 is a view of the two assemblies. The primary goal of the converter assembly is to provide for both online and offline adjustments to the effective stopping depth of the converter system. By using 12 individual plates, the converter depth can be varied from 1 to 12 cm according to the optimal length for a given electron beam energy.

Candidate materials for the converter plates where Carbon and Aluminum due to their ability to create highly forward-peaked Bremsstrahlung radiation from the electron beam (T.A. Antaya, 2010). Nuclear grade graphite was chosen as the final material for its high density, its good thermal conductivity, and because activation of the Aluminum by secondary neutrons and photons presented radiation concerns.



(a) X Magnet



(b) Y Magnet

Figure 2-9: Steering magnet dimensions. Magnets are an H-magnet design with simple racetrack coils. Coil and pole shapes are specified on the left and one quarter of the yoke is shown on the right. Mirror symmetry about both x and y axes is assumed.

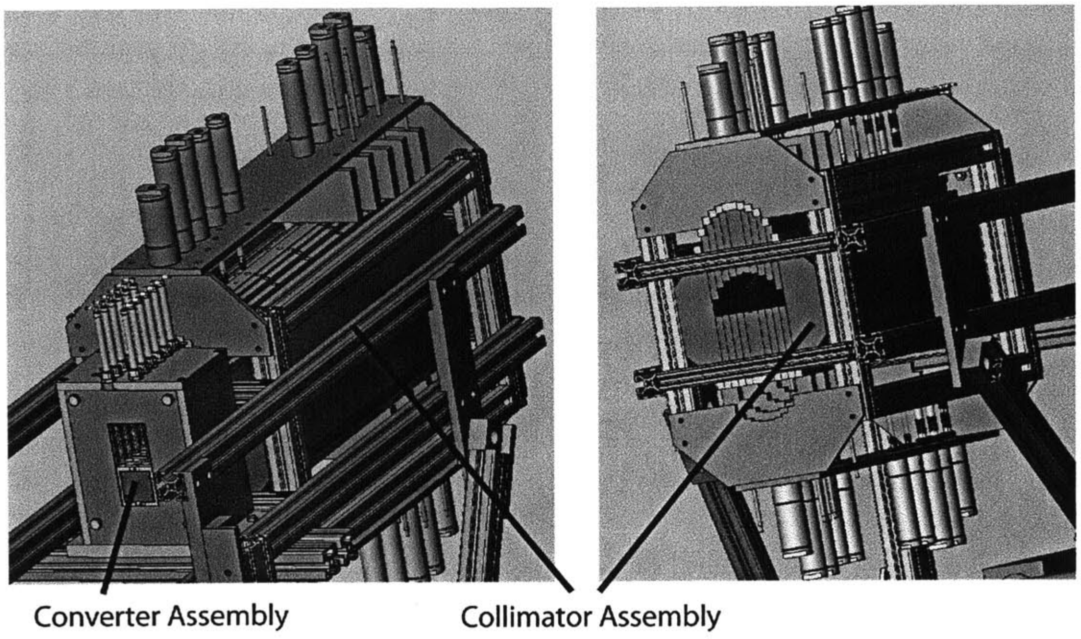


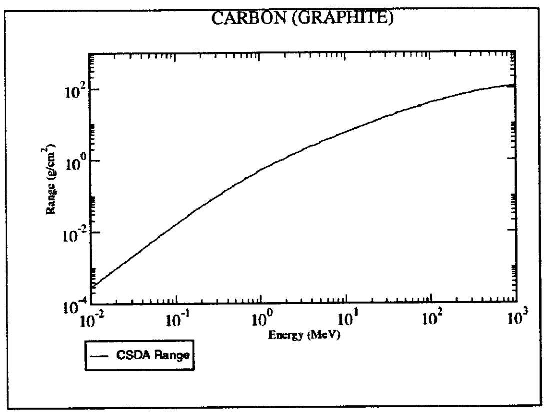
Figure 2-10: View of converter and collimator assemblies. Left shows view from behind the converter assembly, right shows collimator from front (down range). The electron beam enters the converter on the left and the collimated secondary photon beam exits the aperture on the right.

In order to set the total target thickness for a given electron beam energy, the E-STAR electron range database was used. (NIST, 2010) The continuous slowing-down approximation (CSDA) range and Bremsstrahlung yield from E-STAR are shown in figure 2-11. For an electron energy of 60 MeV and graphite density of 1.7 g/cm^3, the E-STAR database gives a CSDA range, R, of 25.98 g/cm^2. The average total distance travelled by the electron is then:

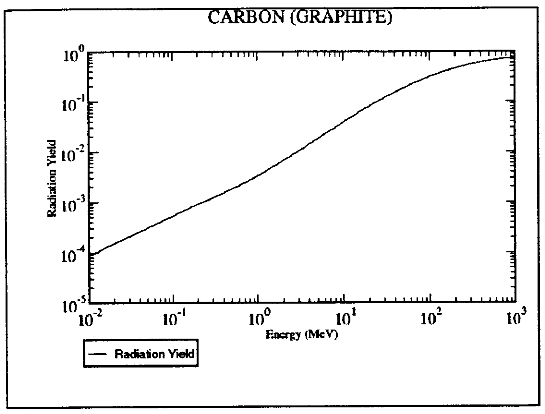
$$x = R/\rho = 15.3 \text{ cm} \tag{2.4}$$

The issue with using the CSDA range for electron range estimation is that the path of electrons in matter is actually highly irregular. For a more accurate estimation of the range with respect to the front plane of the material, data from Tabata et al. (1996) was used.

Using the correlation provided in Tabata et al. (1996) for the 'extrapolated range,' which is the correct range to use for estimating electron range in materials, we find



(a) CSDA Range



(b) Radiative Yield

Figure 2-11: Electron stopping data from E-STAR for graphite, $\rho = 1.7$ g/cm^3.

$R = 31.97$ g/cm^2. This assumes a mean excitation energy (from NIST) of 78.0 eV. This result, which agrees with the data presented in Tabata et al. (1996), says that for high energy electrons in Carbon, the effective range is actually *longer* than the CSDA range.

Given that the stopping range of 60 MeV electrons in graphite is on the order of 15-18 cm, it may seem inconsistent that the converter plate assembly is only a total of 12 cm long. This is the first example of the small discrepancies that arise when large systems are designed by multiple parties. In their preliminary calculations, Raytheon, responsible for secondary radiation simulations, used a density for graphite equal to 2.2 g/cm^3. This results in a CSDA stopping range of 11.8 cm, within the converter assembly length of 12 cm. However, when faced with the task of procuring nuclear-grade graphite for the BBGS, we found that the most readily available products had densities in the range from 1.7 to 1.8 g/cm^3.

The good news is that even though the entire beam will not be stopped in the converter assembly, more detailed simulations (see section 5.3.4) have shown that the amount of energy deposited into the collimator will be a small and manageable percentage of the total beam power.

Chapter 3

BBGS Beam Transport Design

3.1 Transport design process

The design process for the Bremsstrahlung beam generation system began with beam transport design. Before it was possible to design individual components, it was first necessary to create a simplified but encompassing model of the entire system.

The design for this system-level transport model began with intuition gained from previous work, then it evolved as the individual components were designed and iterated upon. Since the ISIS project is compartmentalized among several design teams, changes to external constraints did not always propagate as quickly as would be desirable. This meant that multiple solutions were created before the final product presented in chapter 2 was complete. It is the purpose of this and the following chapters to highlight important pieces of the iteration process which led to the final BBGS design.

This chapter focuses on the design of the main beam-line, which includes the dipole bending magnet, the object and image slits, and the relative distances between all other objects. The goal of the beam design was to meet the Raytheon-specified requirement of approximately 1x (unit) magnification of the LINAC beam at the converter plates. This means that the electron beam at the converter plate should be about the same size as the initial beam. Since the exact specifications of the LINAC beam were unknown when this requirement was specified, a soft requirement of a beam width 'less than 0.5 cm but greater than 0.05 cm' was also used as an acceptable limit throughout the design process.

While the steering magnets were an important consideration during the beam-line design, it was primarily the case that the dipole and beam line design drove the steering magnet design. Hence, the design of those components is not included in this chapter.

3.2 1D Beam Transport using TRANSPORT code

The LINAC beam properties were presented as constraints in chapter 2, but in reality they began as unknowns. The initial beam transport design was started well before the final beam parameters from the linear accelerator were known. Hence, the system design started with simplified methods in order to create a big picture design which could be later modified to fit exact specifications.

Though the exact beam properties were unknown, it was possible to use typical LINAC electron properties in order to begin to create a beam transport solution. The tool used to parameterize the electron beam properties is the phase-space ellipse, illustrated in figure 3-1. The phase-space ellipse is a standard formulation used to characterize charged particle beams. (Livingood, 1969; Reiser, 1994) Since it is often impossible to model all individual particles in a beam, the phase-space ellipse is used to represent the envelope of a charged particle beam. The ellipse is used to describe the displacement (x) of particles about the beam centerline as well as the angle each particle's trajectory makes with the beam centerline (x'). The central idea of the phase-space ellipse formulation is that many charged particle beams, because of the way they are formed and accelerated, can be described by an equivalent beam envelope. The thesis is that since accelerators operate on a charged particle beam with periodic focusing forces, the horizontal and vertical displacement of any particle from the beam centerline inside an accelerator will be sinusoidal in time, provided that the displacements are small.

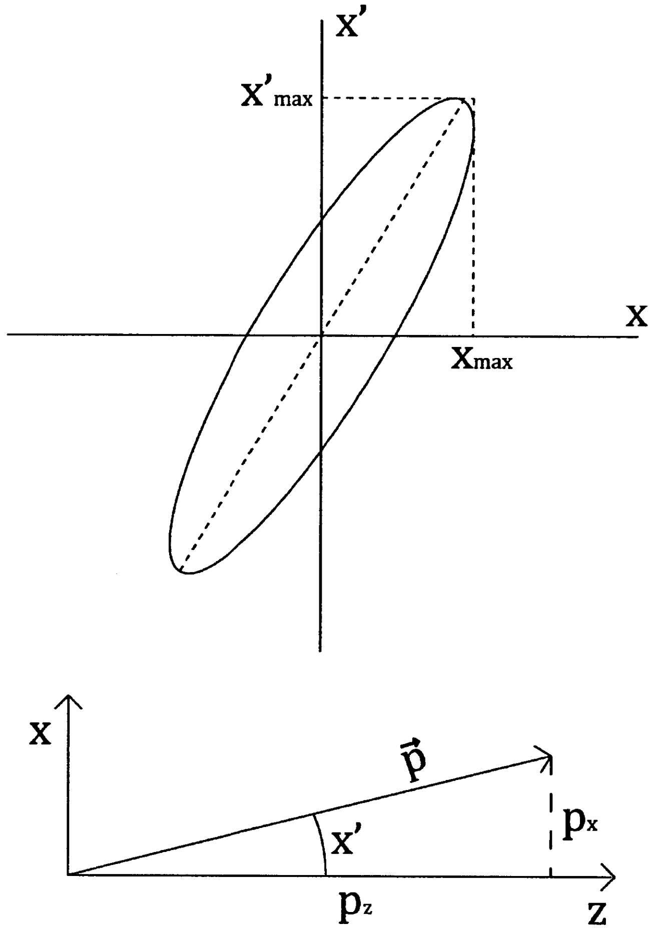


Figure 3-1: Top: Phase-space ellipse used to parameterize the properties of the electron beam. Bottom: coordinate system for beam where z is in the average direction of all particles in beam.

So, the position of a given particle in the accelerator can be described as:

$$x(t) = x_o \sin \frac{2\pi v}{\lambda} t \tag{3.1}$$

where v is the beam velocity and λ is the wavelength of oscillation.

Now, the angle the particle makes with the beam centerline, x', is given by:

$$\begin{aligned} x'(t) &= \frac{dx}{dt}\frac{dt}{dz} \\ &= \frac{dx}{dt}\frac{1}{v} \\ &= x_o \frac{2\pi}{\lambda} \cos \frac{2\pi v}{\lambda} t \qquad (3.2) \\ x'(t) &= x'_o \cos \frac{2\pi v}{\lambda} t \qquad (3.3) \end{aligned}$$

Since $\sin^2\theta + \cos^2\theta = 1$, we can write an equation for the phase-space ellipse within an ideal accelerator:

$$\frac{x^2}{x_o^2} + \frac{x'^2}{x_o'^2} = 1 \tag{3.4}$$

Hence, particle motion in an accelerator results in an upright phase-space ellipse. This result allows for a particle beam simulation to be completed before the final beam properties are known. It is possible to estimate reasonable maximum values for the beam width (x_o) and divergence (x'_o), and with those estimates to build a full simulation of the electron beam transport system.

In the first iteration of the BBGS beam transport design, the matrix-based TRANSPORT (Sta, 1972) code was used to model the dipole within the electron beam-line. TRANSPORT is a very common beam transport code that has been used in the design of many beam systems. Based on preliminary numbers from AES, the beam parameters at the interface between the LINAC and the BBGS were chosen such that the beam emittance was identical in both transverse spaces (x and y) with variable beam width[1]in the range [0.05, 0.5] cm and divergence in the range [0.5, 5.0] mrad, where the beam emittance in a given transverse direction is defined as the area of the

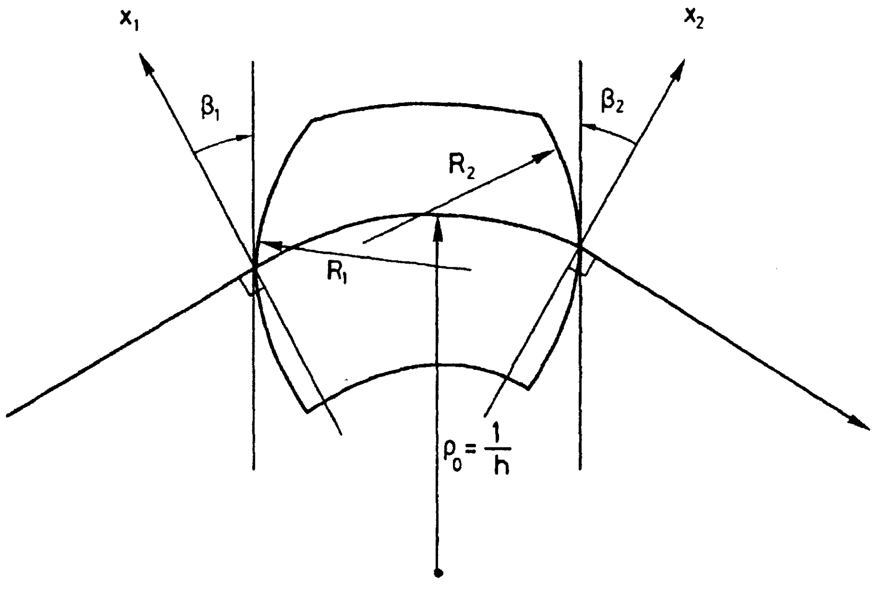


Figure 3-2: Parameters available to characterize a dipole magnet in the TRANSPORT code. BBGS sets $R_1 = R_2 = \infty$. Figure adapted from (Sta, 1972).

phase-space ellipse for that coordinate direction.

The first task was to characterize the beam size at the converter plates for several combinations of beam size and divergence. In order to do so, a basic model for the dipole magnet was required. The magnet was chosen with a radius of 30 cm, and to start with edge-angles (see figure 3-2) $\beta_1 = \beta_2 = 0$ degrees. The results of this first analysis for a beam energy of 60 MeV are shown in table 3.1 and figure 3-3. The beam inputs required by TRANSPORT are x, x', y, y', z, dp, p. The bunch length, z, along with the energy variance, dp, were set to 0. For the momentum (in GeV/c), the following equation was used:

$$p(\text{Gev/c}) = 1\text{E}^{-}3\sqrt{T^2 + 2E_oT} \tag{3.5}$$

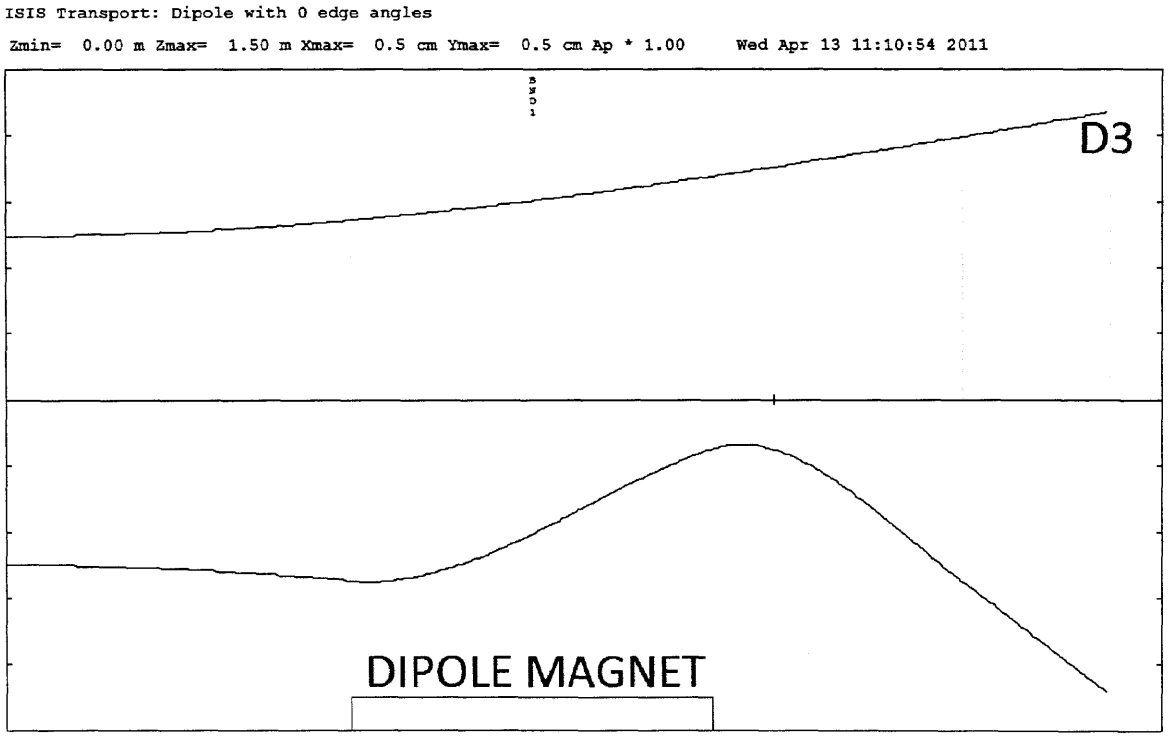
where T is the kinetic energy of the beam and E_o is the electron rest mass energy, equal to 0.511 MeV. This gave an input beam momentum of 0.0605 GeV/c.

Taking a look at the results, the simple dipole without edge angles tends to over-

[1]Here and elsewhere in this text, beam width denotes the distance from the beam center to the maximum extent in a given direction. This is sometimes referred to as the 'half-width' in other texts. For the full beam size measured from one extent to the other, this text uses 'beam diameter.'

Table 3.1: Table of results from 90° dipole with $r_o = 30$ cm and no edge angles.

x_o/y_o	x'_o/y'_o	x_f	x'_f	y_f	y'_f	Label
	0.5	0.089	1.83	0.087	0.50	aa
0.05	2.5	0.145	4.10	0.361	2.50	ab
	5.0	0.249	7.67	0.718	5.0	ac
	0.5	0.428	8.37	0.260	0.50	ba
0.25	2.5	0.443	9.14	0.437	2.50	bb
	5.0	0.487	11.2	0.758	5.0	bc
	0.5	0.854	16.7	0.505	0.50	ca
0.50	2.5	0.862	17.1	0.615	2.50	cb
	5.0	0.885	18.3	0.873	5.0	cc



```
ISIS Transport: Dipole with 0 edge angles
Zmin=  0.00 m Zmax=  1.50 m Xmax=  0.5 cm Ymax=  0.5 cm Ap * 1.00       Wed Apr 13 11:10:54 2011
```

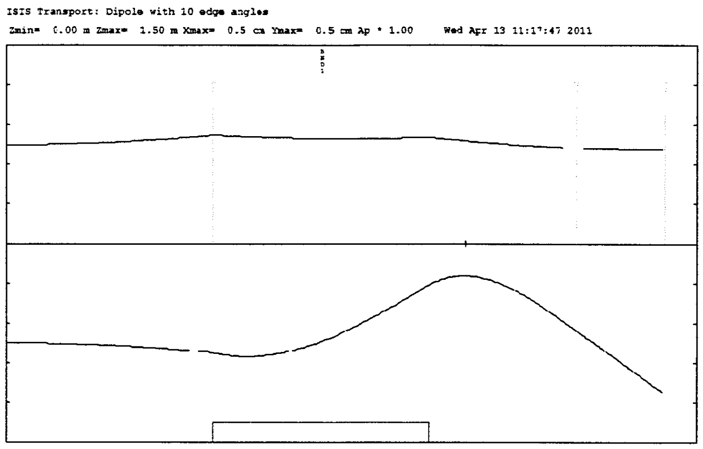
Figure 3-3: TRANSPORT print out for beam label *bb*. Top curve is the y-envelope, bottom curve is the x-envelope. Simulation ends at label D3 which represents the start of the converter plate assembly.

focus in x for all of the beams - there is a beam waist (see figure 3-3) before the converter plates and the beam is larger than it started. It also provides no focusing in the y direction, something which could present trouble for a high-divergence beam. The next step in the design process was to add edge-angles to the dipole in order to counteract the over-focusing of the beam in the x direction.

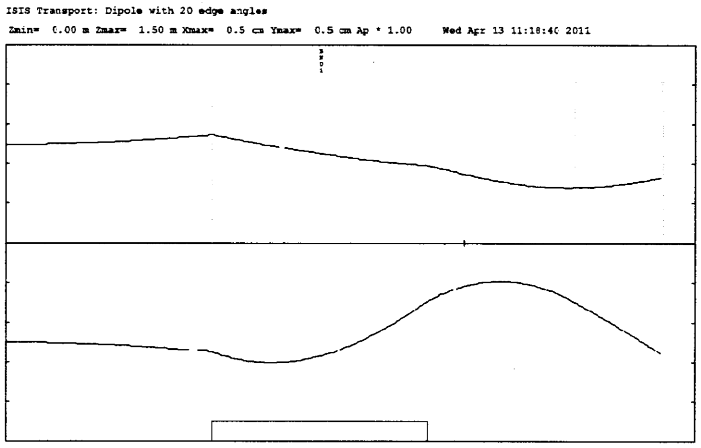
While it is possible to vary the edge angles separately, the decision was made to maintain identical edge angles for both the entrance and exit of the dipole magnet. This kept the dipole design as simple as possible, which in turn made it easier to communicate design specifications to the engineers who created technical documents for each component. Many revisions were created to explore the possible set of edge angles, a few of which are shown in figure 3-4.

In addition to a deterministic beam transport mode, the TRANSPORT code can also run optimization routines to find the best design for a magnet. Holding all parameters fixed except for the edge angles, an optimization was run to obtain equal beam height in x and y at the converter plate. The output from one of such optimizations is shown in figure 3-5. For all of the test cases, the optimal edge angles were between 22 and 30 degrees. The larger beams used in the simulation, which due to details from AES appeared more realistic, required edge angles between 27 and 30 degrees.

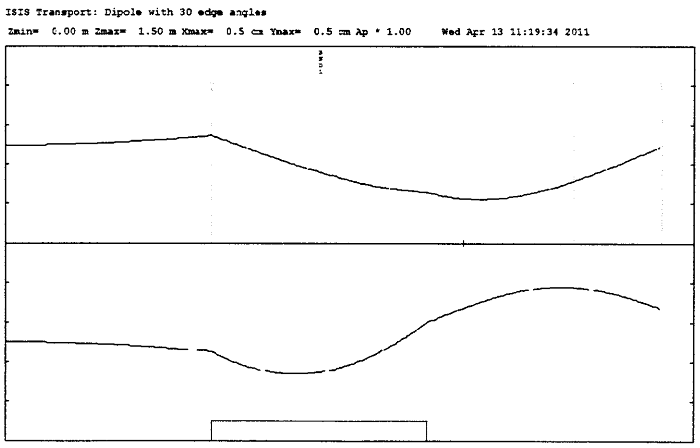
After many simulations and feedback from MIT designers, the edge angles for the dipole were set to 30° for both the entrance and exit angles. While this number would not be optimal for all beam configurations, it was close enough and a 'round' number which could be easily relayed to engineering. It is also the case that for the final dipole parameters, edge angles of 27.5 degrees correspond to a focal distance of 20.5 cm, which is the location of the image slit. It was important that the edge angles be chosen such that the focus was pushed beyond the image slit. This resolved the 1D beam design for the BBGS dipole. However, as more detailed information about the LINAC beam became available, it was necessary to verify the 1D results using more complex codes.



(a) 10 degrees



(b) 20 degrees



(c) 30 degrees

Figure 3-4: TRANSPORT print out for beam label *bb* with different edge angles.

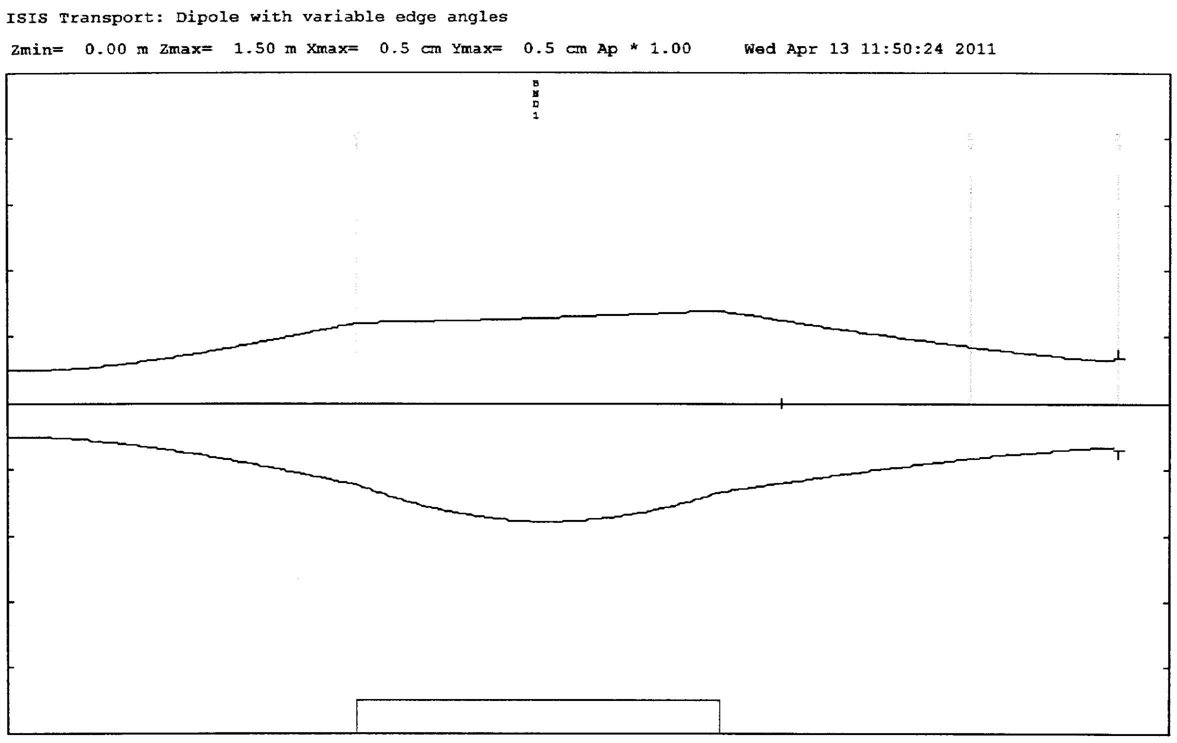


Figure 3-5: TRANSPORT print out for beam label *ab* with edge angle optimization. Edge angles are 26.7°.

3.3 pybeam1d Transport Code

As details about the phase space and energy distribution of the electron beam began to filter in, it became apparent that a more detailed analysis might be necessary to verify the operation of the BBGS transport system. Before moving to full 2D and 3D field models of the magnet system, a 1D transport code was developed by the author which could more accurately represent the ISIS electron beam. The basic idea of this code, *pybeam1d*, was to use matrix transport methods to transport individual particles rather than an idealized beam envelope. This '1D+' particle tracking method made it possible to simulate the transport of non-ideal accelerator beams. The process of writing this code also helped to gain a better understanding of the assumptions used in the TRANSPORT code.

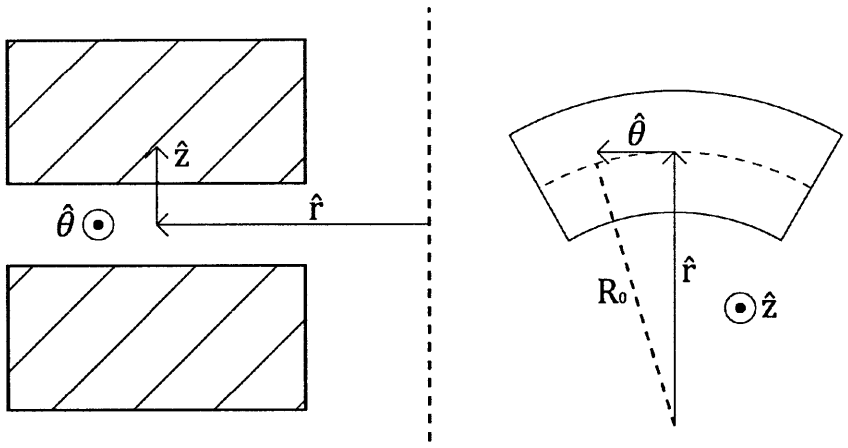


Figure 3-6: Geometry for particle transport calculations. Left: cross section of magnet poles, beam out of page. Right: top view

3.3.1 Code Scope

The scope of the pybeam1d code is not to recreate or enhance all of the functionality of the TRANSPORT code. Instead, it provides a *Python*-based extensible framework with the following built-in functionalities:

1. Matrix transport through dipoles and free space
2. Variable field index and quadrupole edges for dipoles (see chapter 4)
3. Random beam generation based on given phase-space parameters
4. Interface to generate equivalent beams for Opera-3D models
5. Basic space-charge integration (see chapter 6)

3.3.2 Basic Theory

Since many books on the subject matter (Livingood, 1969; Reiser, 1994) do not explicitly derive Matrix-based beam transport equations, a short overview of these derivations is provided here. We start with beam transport in cylindrical geometries (such as that in dipoles) where the beam trajectory travels in the $\hat{\theta}$ direction at a constant equilibrium radius, r_0. This geometry is described in figure 3-6.

Magnetic Field in Axisymmetric Cylindrical Geometry

The theory of beam transport in cylindrical geometries starts with (of course) Maxwell's equations. We can write Ampere's Law for free space:

$$\nabla \times \mathbf{B} = \mu_o \mathbf{j} + \frac{1}{c^2}\frac{\partial E}{\partial t} \tag{3.6}$$

Now, assuming steady state ($\frac{\partial E}{\partial t} = 0$), the equation for the magnetic flux density in a region free of current becomes:

$$\nabla \times \mathbf{B} = 0 \tag{3.7}$$

Expanding this vector expression for cylindrical coordinates, we have:

$$\nabla \times \mathbf{B} = \left(\frac{1}{r}\frac{\partial B_z}{\partial \theta} - \frac{\partial B_\theta}{\partial z}\right)\hat{\mathbf{r}} + \left(\frac{\partial B_r}{\partial z} - \frac{\partial B_z}{\partial r}\right)\hat{\theta} + \frac{1}{r}\left(\frac{\partial (rB_\theta)}{\partial r} - \frac{\partial B_r}{\partial \theta}\right)\hat{\mathbf{z}} \tag{3.8}$$

Since $\nabla \times \mathbf{B} = 0$, each of its vector components must also be equal to zero. This gives:

$$\frac{\partial B_z}{\partial r} - \frac{\partial B_r}{\partial z} = 0 \tag{3.9}$$

Equation 3.9 can be found throughout the scientific literature. It is important because it allows the description of coupling between radial and axial oscillations in cylindrical geometries.

Another consequence of $\nabla \times \mathbf{B} = 0$ is that the magnetic field can be written as the gradient of some scalar potential, ϕ:

$$\mathbf{B} = \nabla \phi \tag{3.10}$$

And hence since $\nabla \cdot \mathbf{B} = 0$:

$$\nabla^2 \phi = 0 \tag{3.11}$$

In cylindrical coordinates, this evaluates to:

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial r^2} + \frac{1}{r}\frac{\partial\phi}{\partial r} + \frac{1}{r^2}\frac{\partial^2\phi}{\partial\theta^2} + \frac{\partial^2\phi}{\partial z^2} = 0 \tag{3.12}$$

Now, since the region of interest for beam transport calculations will be a small region centered about the equilibrium radius r_0, we write $r = r_0 + x$. Substituting for r and eliminating terms which vary in θ gives:

$$\frac{\partial^2\phi}{\partial x^2} + \frac{1}{r_0 + x}\frac{\partial\phi}{\partial x} + \frac{\partial^2\phi}{\partial z^2} = 0 \tag{3.13}$$

Assuming that $\frac{\partial\phi}{\partial x}$ is small compared to r_0, the Laplacian is simplified to the Cartesian result that:

$$\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial z^2} = 0 \tag{3.14}$$

Since x and z are small, we seek solutions for ϕ of order 2 which will result in magnetic fields to the first order. To the second order, the scalar potential can be expanded in polynomial form as:

$$\phi(x, z) = c_1x + c_2z + c_3(x^2 - z^2) + c_4xz \tag{3.15}$$

Here we set $c_3 = 0$, since the geometry of our solutions will be such that the fields in z will either be independent of x or always increasing/ always decreasing in x, as illustrated in figure 3-7. This simplification holds true for most beam transport scenarios and it simplifies the equations of motion by decoupling the x and z equations of motion. This gives:

$$\phi(x, z) = c_1x + c_2z + c_4xz \tag{3.16}$$

and hence

$$\mathbf{B}(x, z) = (c_1 + c_4z)\,\hat{\mathbf{x}} + (c_2 + c_4x)\,\hat{\mathbf{z}} \tag{3.17}$$

We know $B_x(0,0) = 0$ and $B_z(0,0) = B_0$, hence:

$$\mathbf{B}(x, z) = kz\,\hat{\mathbf{x}} + (B_0 + kx)\,\hat{\mathbf{z}} \tag{3.18}$$

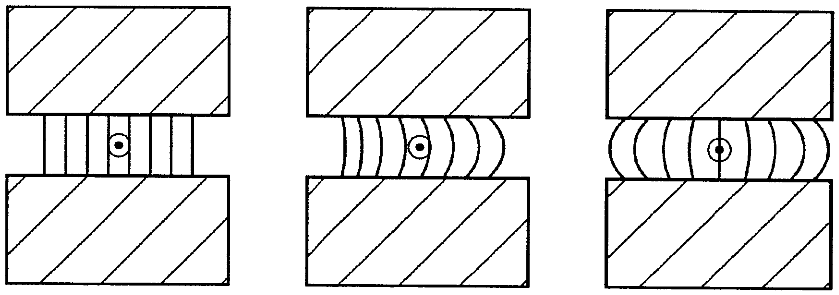


Figure 3-7: Possible field configurations for transport solutions. Left and center are allowed, right is a disallowed configuration in this model.

where k is a constant factor to be defined later.

Equations of Motion About Equilibrium Orbit

In beam dynamics it is useful to plot the trajectory of an 'equilibrium' particle which represents the average properties of a particular beam. This particle moves at the average momentum and travels down the center of the beam line. In this discussion, this particle travels in a circular orbit with bending radius, r_0, momentum, p_0, velocity, v_0, relativistic mass, m, and position with respect to beam center, $(x, z) = (0, 0)$.

The goal of beam transport simulations is to characterize the motion of particles in the beam whose properties differ slightly from the equilibrium orbit properties. Hence, a new variable x is adopted to represent the displacement from equilibrium radius, $x = r - r_0$.

The first step in deriving equations of motion is to find the force exerted on a particle from the magnetic field:

$$\mathbf{F} = q(\mathbf{v} \times \mathbf{B}) \tag{3.19}$$

Here it is assumed that the transverse velocities (x', z') are much smaller than the velocity in the $\hat{\theta}$ direction, so the transverse forces on the non-equilibrium particle become:

$$F_r(r) = -qvB_z(r) \tag{3.20}$$

$$F_z(z) = qvB_r(z) \tag{3.21}$$

Axial Motion To solve for the axial motion, we write the differential equation:

$$ma_z = m\frac{d^2z}{dt^2} = qvB_r(z) \tag{3.22}$$

where here and elsewhere m is the relativistic mass, which is assumed to be approximately constant since the particle momentum p is primarily in the $\hat{\theta}$ direction and the acceleration does not occur in this direction. Now, the field $B_r(z)$ in equation 3.22 can be represented using equation 3.18:

$$B_r(z) = kz \tag{3.23}$$

Here we note that k is in fact equal to $\frac{\partial B_r}{\partial z}$, which by equation 3.9 is also equal to $\frac{\partial B_z}{\partial r}$. This gives:

$$\frac{d^2z}{dt^2} = \frac{qv}{m}\left(\frac{\partial B_z}{\partial r}\right) z \tag{3.24}$$

It is now useful to define the field index, n, a dimensionless parameter which helps to characterize particle motion:

$$n = -\frac{r}{B_z}\frac{\partial B_z}{\partial r} \tag{3.25}$$

Solving for $\frac{\partial B_z}{\partial r}$ near r_0 gives:

$$\frac{\partial B_z}{\partial r} = -n\frac{B_z(r_0)}{r_0} \tag{3.26}$$

plugging this result in to equation 3.24:

$$\frac{d^2z}{dt^2} = \frac{qv}{m}\left(-\frac{n}{r_0}B_z(r_0)\, z\right) \tag{3.27}$$

Using the equation for the equilibrium orbit field, $B_z(r_0) = \frac{mv_0}{qr_0}$, the equation for axial motion becomes:

$$\frac{d^2z}{dt^2} + \frac{v_0^2}{r_0^2} n\, z = 0 \tag{3.28}$$

Substituting for the path length $S = v_0 t$ and $dt = dS/v_0$, the equation used for axial motion through magnets in *pybeam1d* is:

$$\frac{d^2z}{dS^2} + \frac{n}{r_0^2}\, z = 0 \tag{3.29}$$

Radial Motion The solution for radial motion follows the same method. Writing the equation of motion from the force balance:

$$ma_r = m\left(\frac{d^2r}{dt^2} - r\left(\frac{d\theta}{dt}\right)^2\right) = -qvB_z(r) \tag{3.30}$$

Using

$$\frac{d\theta}{dt} = \omega = \frac{v}{r} \tag{3.31}$$

gives:

$$\frac{d^2r}{dt^2} - \frac{v^2}{r} + \frac{qv}{m}B_z(r) = 0 \tag{3.32}$$

Substituting $r_0 + x$ for r, this gives:

$$\frac{d^2(r_0 + x)}{dt^2} - \frac{v^2}{r_0 + x} + \frac{qv}{m}B_z(r) = 0 \tag{3.33}$$

Which can be simplified by taking the following steps:

1. Factor out $\frac{1}{r_0}$ from second term to give $-\frac{1}{r_0}\frac{v^2}{1+\frac{x}{r_0}}$
2. Since $x << r_0$, $\frac{1}{1+\frac{x}{r_0}}$ can be approximated, taking only the first term in the binomial expansion as: $1 + \frac{x}{r_0}$
3. Since r_0 is constant, $\frac{d^2(r_0+x)}{dt^2} = \frac{d^2x}{dt^2}$

These steps give:

$$\frac{d^2x}{dt^2} - \frac{v^2}{r_0}\left(1 + \frac{x}{r_0}\right) + \frac{qv}{m}B_z(r) = 0 \tag{3.34}$$

Now it is possible to use equation 3.18, taking only the z fields:

$$B_z(r) = B_z(r_0) + k\,x \tag{3.35}$$

Again, noting that $k = -n\frac{B_z(r_0)}{r_0}$ and using $B_z(r_0) = \frac{mv_0}{qr_0}$, the equation for radial motion becomes:

$$\frac{d^2x}{dt^2} - \frac{v^2}{r_0} + \frac{vv_0}{r_0} + \left(\frac{v^2}{r_0^2} - \frac{vv_0}{r_0^2}n\right) x = 0 \tag{3.36}$$

Here it is important to draw a distinction between the equilibrium velocity, v_0 and the non-equilibrium particle velocity, v, which may be different. Writing $v = v_0 + \delta v$ where δv is a small perturbation from the equilibrium velocity, we have:

$$\frac{d^2x}{dt^2} - \frac{(v+\delta v)^2}{r_0} + \frac{(v_0+\delta v)v_0}{r_0} + \left(\frac{v^2}{r_0^2} - \frac{vv_0}{r_0^2}n\right) x = 0 \tag{3.37}$$

Expanding and dropping terms in δv^2 as well as terms with $\frac{\delta v x}{r_0^2}$ since they are much smaller than $\frac{\delta v}{r_0}$ terms:

$$\frac{d^2x}{dt^2} + \frac{v_0^2}{r_0^2}\left(1-n\right) x = v_0\frac{\delta v}{r_0} \tag{3.38}$$

Now, substituting for $S = v_0 t$:

$$\frac{d^2x}{dS^2} + \frac{1}{r_0^2}\left(1-n\right) x = \frac{1}{r_0}\frac{\delta v}{v_0} \tag{3.39}$$

Or, in terms of momentum:

$$\frac{d^2x}{dS^2} + \frac{1}{r_0^2}\left(1-n\right) x = \frac{1}{r_0}\frac{\Delta p}{p_0} \tag{3.40}$$

which gives the equation for radial motion used in *pybeam1d*. This equation agrees with those found in the literature, for instance Livingood (1969) eq 2-1.

3.3.3 Matrix Equations

Following the process of Livingood (1969), the *pybeam1d* code treats beam-line components with transfer matrices that take the initial phase space and transform it to the phase space at the exit of the component. This transfer is written in matrix form:

$$\mathbf{x_f} = \mathbf{M}\mathbf{x_o} \tag{3.41}$$

where $\mathbf{x}$ is the vector:

$$\mathbf{x} = \begin{pmatrix} x \\ x' \\ \frac{\Delta p}{p} \end{pmatrix} \tag{3.42}$$

and $\mathbf{M}$ is a 3 by 3 transfer matrix for the given component.

To find the equations for x and x', solutions to equation 3.40 are sought in the form:

$$x = a\sin\frac{\delta S}{r_0} + b\cos\frac{\delta S}{r_0} + \frac{r_0}{\delta^2}\frac{\Delta p}{p} \tag{3.43}$$

$$x' = a\frac{\delta}{r_0}\cos\frac{\delta S}{r_0} - b\frac{\delta}{r_0}\sin\frac{\delta S}{r_0} \tag{3.44}$$

where

$$\delta = \sqrt{1-n} \tag{3.45}$$

Given initial values x_o and x'_o, plus defining the quantity:

$$\phi = \frac{\delta S}{r_0} = \delta\,\theta \tag{3.46}$$

where θ is the angular extent of the dipole path, then

$$a = r_0 x'_o/\delta \quad \text{and} \quad b = x_o - (r_0/\delta^2)\Delta p/p \tag{3.47}$$

The transfer matrix for axial motion in a dipole magnet becomes:

$$M_{D,x} = \begin{pmatrix} \cos\phi & \frac{r_0}{\delta}\sin\phi & \frac{r_0}{\delta^2}(1-\cos\phi) \\ -\frac{\delta}{r_0}\sin\phi & \cos\phi & \frac{1}{\delta}\sin\phi \\ 0 & 0 & 1 \end{pmatrix} \tag{3.48}$$

The transfer matrix given by a dipole edge with angle α is given by the thin-lens approximation (see Livingood (1969)):

$$M_{E,x} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{\tan\alpha}{r_0} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.49}$$

And for a vacuum drift of length S:

$$M_{S,x} = \begin{pmatrix} 1 & S & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.50}$$

For the axial equations, let us replace the variable z in equation 3.29 with y. Now, the matrix equations will be the same as those for motion in x, modified as follows:

1. Replace $\delta = \sqrt{1-n}$ with $\epsilon = \sqrt{n}$
2. For edge angles, change the sign of $\tan\alpha$
3. Ignore variance in momentum

In the equations for y motion, a special case occurs when $n = 0$. We have:

$$\frac{\sin\phi}{\delta} = \frac{\sin\phi}{\epsilon} = \frac{0}{0} \tag{3.51}$$

However, by L'Hospital's rule:

$$\lim_{\epsilon\to 0} \frac{\sin\epsilon\theta}{\epsilon} = \frac{\theta\cos\epsilon\theta}{1} = \theta \tag{3.52}$$

The *pybeam1d* code implements a switch which checks for this special case when applying the matrix equations for axial motion.

3.3.4 pybeam1d Benchmark

Before using the *pybeam1d* code for more detailed beams, it was important to benchmark the code against TRANSPORT calculations. The results of this benchmark are shown in table 3.2. The only differences between the two results are that in *pybeam1d*

Table 3.2: Table of *pybeam1d* results from 90° dipole with $r_o = 30$ cm and no edge angles. Compare to table 3.1 for TRANSPORT results.

x_o/y_o	x'_o/y'_o	x_f	x'_f	y_f	y'_f	Label
0.05	0.5	0.087	1.797	0.086	0.491	aa
	2.5	0.142	4.023	0.356	2.454	ab
	5.0	0.244	7.474	0.707	4.908	ac
0.25	0.5	0.423	8.278	0.254	0.491	ba
	2.5	0.436	8.984	0.429	2.454	bb
	5.0	0.477	11.099	0.747	4.908	bc
0.50	0.5	0.845	16.520	0.492	0.491	ca
	2.5	0.850	16.801	0.608	2.454	cb
	5.0	0.872	17.969	0.857	4.908	cc

the maximum parameters are used as a basis for generating random particles, hence the numbers are not exactly the same since the actual maximum will not be exactly equal to the maximum random radius generated.

In this benchmark, the beam was generated by first picking values for r and r', the distance from the beam axis and angle in the radial direction, from a uniform ellipse. This assumes that since the beam is exiting an accelerator with radial symmetry, the beam should be approximately radially symmetric.

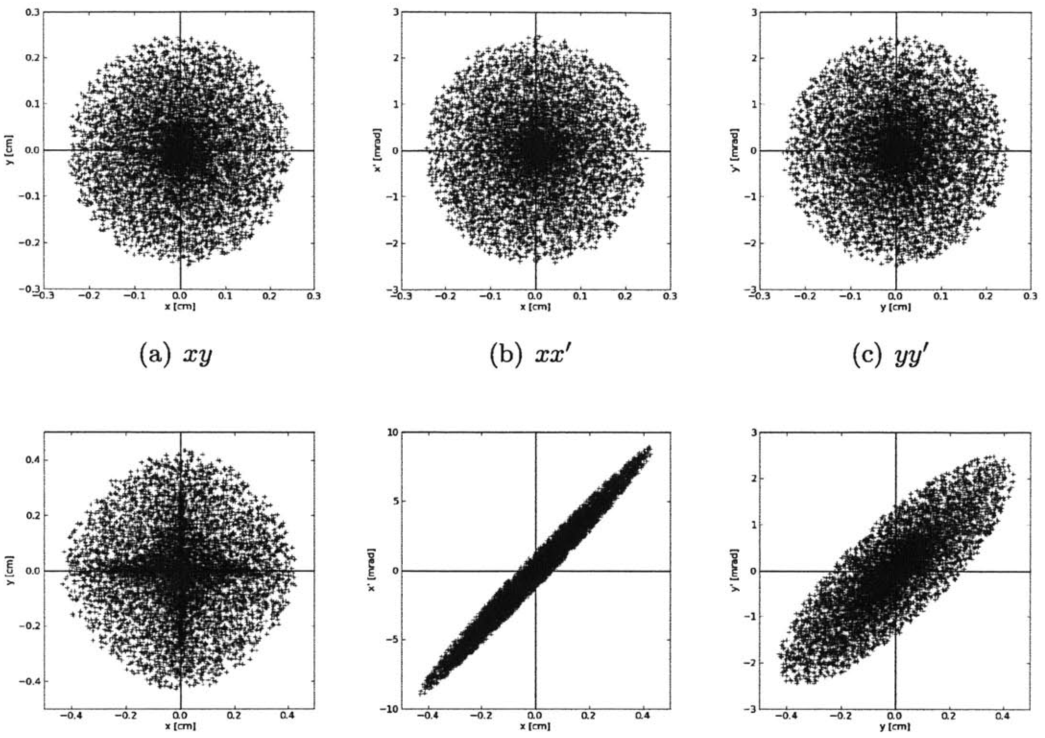
Once the radial beam was generated, values for the x and y directions were extracted by assigning a random angle θ to each point such that:

$$x = r\cos\theta \tag{3.53}$$

$$y = r\sin\theta \tag{3.54}$$

$$x' = r'\cos\theta \tag{3.55}$$

$$y' = r'\sin\theta \tag{3.56}$$



(a) xy (b) xx' (c) yy'

(d) xy (e) xx' (f) yy'

Figure 3-8: *pybeam1d* print out for beam label *bb*. Top row: starting phase space. Bottom row: phase space at converter plate.

Examples of the initial and final beam profiles are shown in figure 3-8.

3.3.5 Beam Transport with Energy Variance

One of the primary reasons for developing the *pybeam1d* transport code was to explore the effects of variations in beam energy on the BBGS electron beam. Even before the LINAC beam simulations were complete, it was known with certainty that there would be some spread in the beam momentum.

The electron beam is accelerated in a pulsed mode, which means the beam is not continuous but actually a series of 'bunches' of particles, each which have a finite length in the beam direction. As the bunches are accelerated across a voltage gap, the voltage is changing sinusoidally in time in order to prepare for the next bunch. This means that particles at the front of the bunch do not experience the same accelerating fields as those in the back of the bunch. Hence, as the beam is accelerated and the momentum increases, the variance in the beam momentum also increases.

Before the electron beam parameters were known, a study was performed to probe the sensitivity of the BBGS transport system to variance in the beam momentum. For this study, a beam of 5000 particles was generated and each particle was assigned a momentum at random from a normalized Gaussian distribution given by:

$$f(p) = \frac{1}{\sqrt{2\pi\sigma_p^2}} \exp - \frac{(p-\bar{p})^2}{2\sigma_p^2} \tag{3.57}$$

where $\bar{p}$ is the average beam momentum and σ_p is the square root of the momentum variance.

For each of the 9 beams simulated in the TRANSPORT study, three different variances in momentum were tested: $\frac{\sigma_p}{\bar{p}} = 0.01, 0.001, 0.0001$. Plots of these momentum distributions are shown in figure 3-9. The results of this set of *pybeam1d* simulations are shown in table 3.3 and figure 3-10. As expected, when the momentum variance is small, the beam behaves nearly identically to that of the monoenergetic electron beam. However, for a standard deviation in momentum of 1%, the beam magnification in the x direction is increased significantly.

At the time these simulations were completed, more clear information on the electron beam began to filter in from AES. The detailed LINAC simulations showed a standard deviation in beam momentum on the order of 0.2%, plus a high skewness in the energy distribution. It became clear that even more detailed calculations would need to be completed in order to estimate the actual beam size at the converter plates.

3.3.6 Transport of AES Beam

The engineers at AES provided three 50,000 particle simulations of the LINAC electron beam as it exits the accelerator. The first step to use these files for transport in the BBGS models was to convert them to distributions we could use in our simulations. The output files generated by AES gave beam data in Snyder-Courant formulation, in the form of 6 variables: $(x, (\beta\gamma)_x, y, (\beta\gamma)_y, z, (\beta\gamma)_z)$, where x, y and z

Table 3.3: Table of *pybeam1d* results from 90° dipole with r_o = 30 cm and no edge angles. Table adds variance in momentum. Units are cm and mrad. Results highlighted in grey are shown in figure 3-10

x_o	x'_o	σ_p/p	x_f	x'_f	Label
0.05	0.5	0.0100	2.901	36.069	aaa
		0.0010	0.335	4.350	aab
		0.0001	0.093	1.880	aac
		0.0000	0.087	1.791	aad
	2.5	0.0100	2.850	35.480	aba
		0.0010	0.416	6.616	abb
		0.0001	0.144	4.040	abc
		0.0000	0.140	4.002	abd
	5.0	0.0100	3.451	43.777	aca
		0.0010	0.389	8.661	acb
		0.0001	0.255	7.435	acc
		0.0000	0.244	7.507	acd
0.25	0.5	0.0100	3.377	43.821	baa
		0.0010	0.592	9.821	bab
		0.0001	0.424	8.228	bac
		0.0000	0.419	8.156	bad
	2.5	0.0100	2.940	38.742	bba
		0.0010	0.613	10.764	bbb
		0.0001	0.445	8.946	bbc
		0.0000	0.433	8.957	bbd
	5.0	0.0100	2.975	39.465	bca
		0.0010	0.655	12.651	bcb
		0.0001	0.478	10.744	bcc
		0.0000	0.473	10.819	bcd
0.50	0.5	0.0100	3.552	48.626	caa
		0.0010	0.943	17.432	cab
		0.0001	0.841	16.385	cac
		0.0000	0.838	16.358	cad
	2.5	0.0100	3.155	42.981	cba
		0.0010	0.956	18.232	cbb
		0.0001	0.849	16.839	cbc
		0.0000	0.838	16.679	cbd
	5.0	0.0100	3.232	42.937	cca
		0.0010	1.055	19.522	ccb
		0.0001	0.881	17.878	ccc
		0.0000	0.866	17.915	ccd

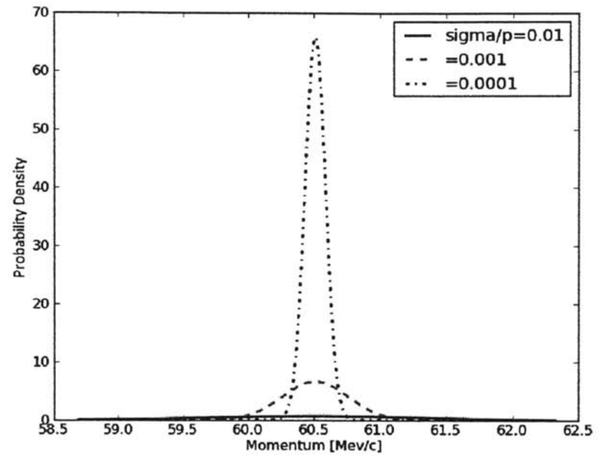
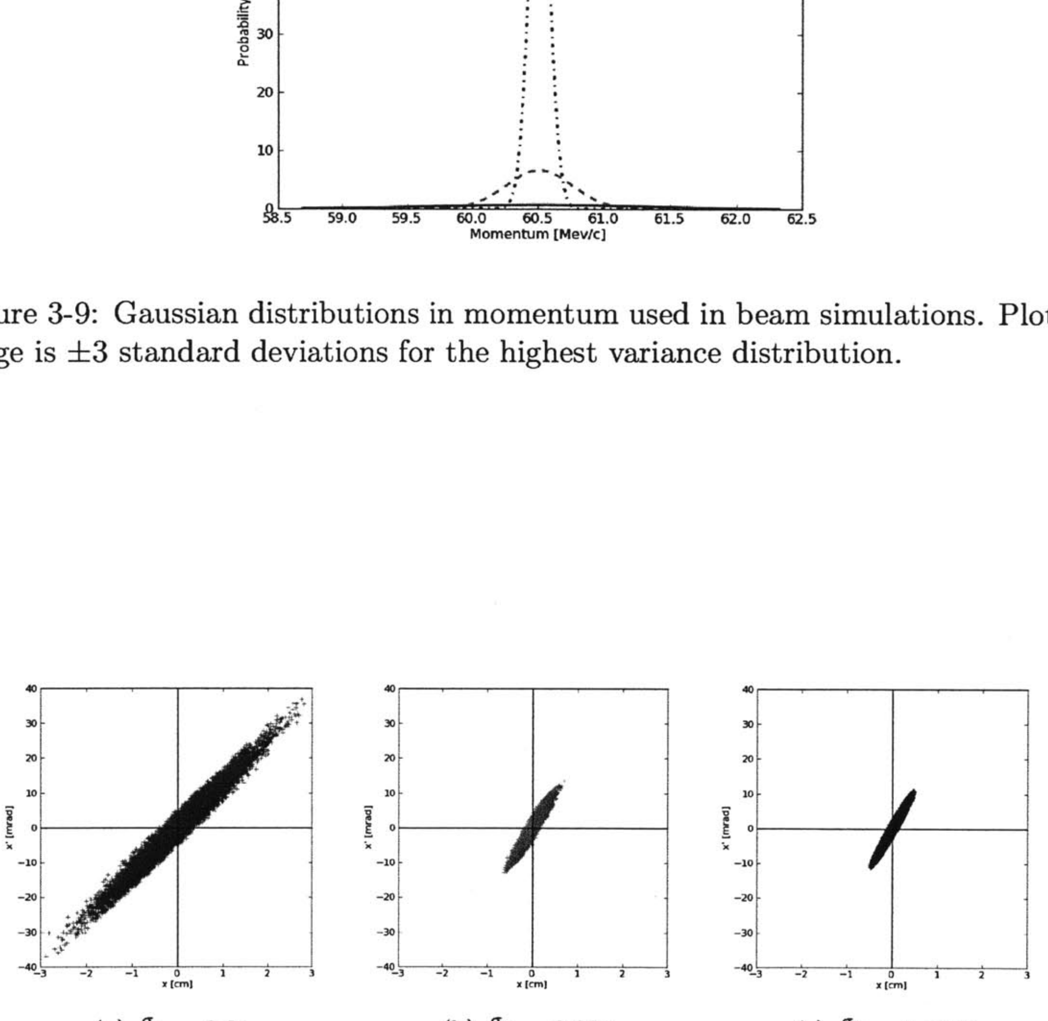


Figure 3-9: Gaussian distributions in momentum used in beam simulations. Plotting range is ±3 standard deviations for the highest variance distribution.



(a) $\frac{\sigma_p}{p} = 0.01$ (b) $\frac{\sigma_p}{p} = 0.001$ (c) $\frac{\sigma_p}{p} = 0.0001$

Figure 3-10: *pybeam1d xx'* phase space for initial beam $x_o = 0.25$ cm, $x'_o = 5.0$ mrad with variance in momentum. From left to right, beam labels are: *bca, bcb, bcc.* As seen, the image phase space was increased significantly when the momentum spread was large.

are the transverse, vertical, and longitudinal locations of the particle compared to a reference particle in the center of the beam, and $(\beta\gamma)_u$ is the dimensionless momentum in the u direction. In this notation, β_u is defined as:

$$\beta_u = \frac{v_u}{c} \tag{3.58}$$

where v_u is the particle velocity in the u direction and c is the speed of light. γ_u is the Lorentz factor, defined as:

$$\gamma_u = \frac{1}{(1-\beta_u^2)^{\frac{1}{2}}} \tag{3.59}$$

The dimensionless momentum data from Parmela were used to generate pairs of (x', y') for each electron, where x' and y' are the angles the particle trajectory makes with the z axis in the x and y directions. Since these angles are small, the approximation $\tan\theta \approx \theta$ was used to find, for direction u:

$$u' = \frac{p_u}{p} = \frac{p_u}{p_z} = \frac{(\beta\gamma)_u}{(\beta\gamma)_z} \tag{3.60}$$

Finally, the particle kinetic energy, T, was found by:

$$T = (\gamma - 1)E_o \tag{3.61}$$

where

$$\gamma = \sqrt{(\beta\gamma)^2 + 1} \tag{3.62}$$

and

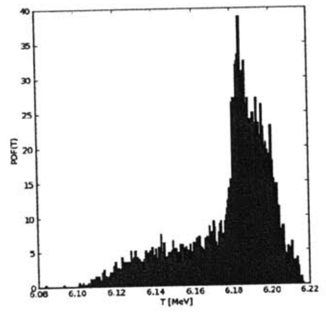
$$(\beta\gamma) = \sqrt{(\beta\gamma)_x^2 + (\beta\gamma)_y^2 + (\beta\gamma)_z^2} \tag{3.63}$$

Before running simulations with the electron beam data from AES, it was important to characterize the basic features of the beam. Three beam simulations were provided by AES. These beams have average energies of approximately 60.7, 30.6, and 6.2 MeV. Hence, they are referenced in this report as beams 607, 306, and 062. A summary table of beam parameters provided by AES is shown in table 3.4.

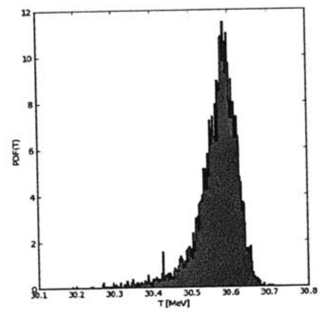
Based on simulations using normally distributed beams of similar momentum

Table 3.4: Beam properties of AES Parmela simulated beam. Dimensions in cm, angles in mrad.

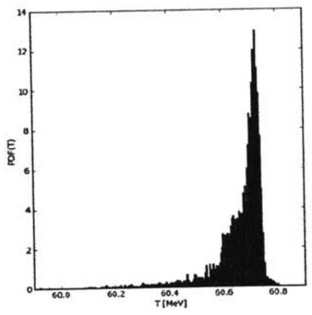
$\langle T \rangle$ MeV	$\sigma_T(\%)$	x_{rms}	x_{max}	y_{rms}	y_{max}	$\% > rms$	x'_{rms}	y'_{rms}
60.66	0.21	0.136	0.489	0.137	0.487	43.3	0.483	0.486
30.57	0.23	0.159	0.510	0.161	0.511	46.9	0.669	0.675
6.178	0.38	0.167	0.509	0.168	0.511	51.7	0.420	0.419



(a) 062



(b) 306

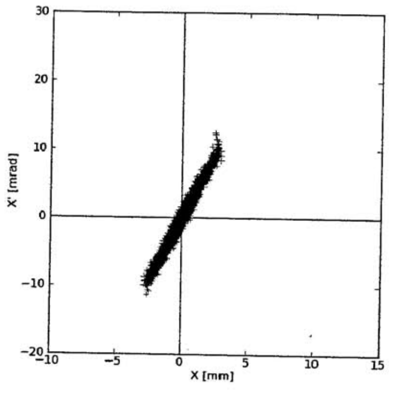


(c) 607

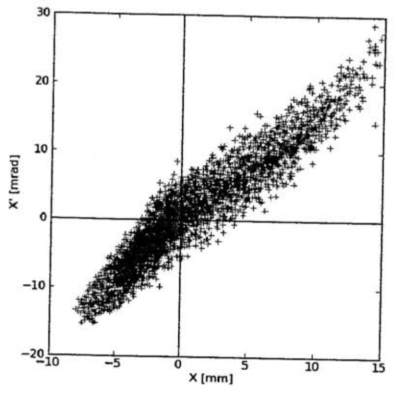
Figure 3-11: Beam energy distributions for all three AES simulations.

variance, the AES beams do not immediately seem troublesome. However, it is clear from observation that the energy distributions of the AES beam have longer 'tails' than a Gaussian distribution fit to the peak (see figure 3-11). This means that extreme values (values far from the mean) are more likely in the AES beam than in the simulated normally distributed beams.

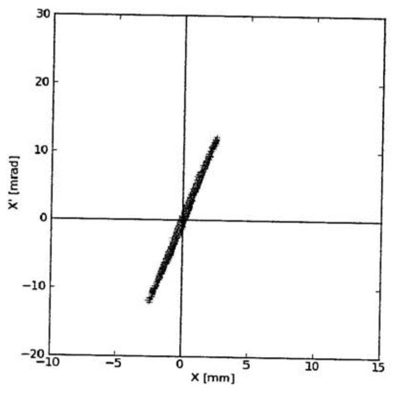
The effect of the low-energy tails of the electron beams from AES was tested by running the *pybeam1d* code with the AES beam as an input file. In all three cases, the low energy beam tail introduces a skewness in the final phase-space ellipse. To showcase this effect, the final phase-space ellipses are presented in figure 3-12 for the AES phase-space with and without variance in energy. The monoenergetic plots were created by using the phase space provided by the AES beam but assigning all particle momentum values equal to the average momentum of the beam. Figure 3-13 shows the correlation between particle kinetic energy and displacement from beam centerline as measured at the converter plate. As expected, there was a clear correlation between the lateral displacement of particles and the kinetic energy. Particles with lower



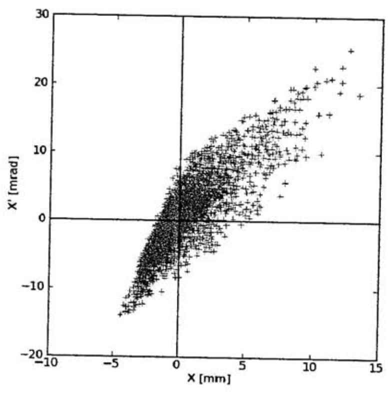
(a) 062 - Monoenergetic



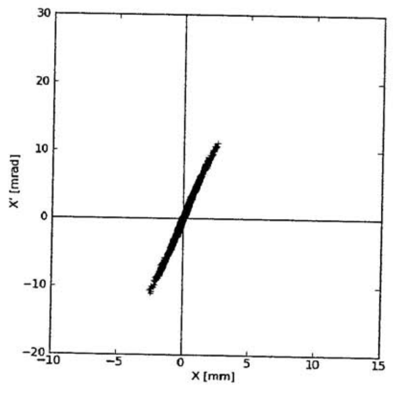
(b) 062



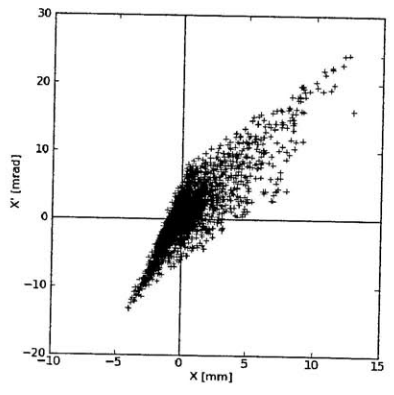
(c) 306 - Monoenergetic



(d) 306

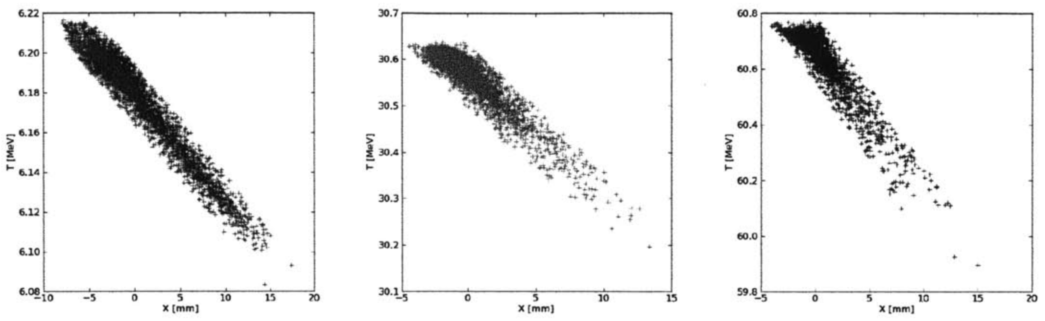


(e) 607 - Monoenergetic



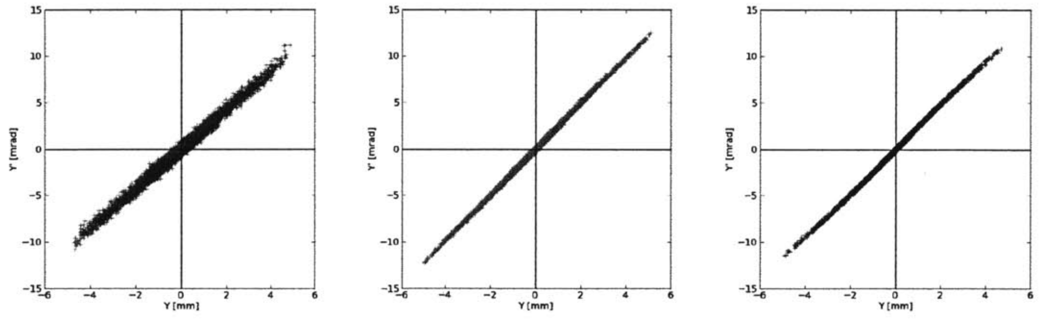
(f) 607

Figure 3-12: Beam xx' phase space at the converter plates for each beam energy both with and without energy variance.



(a) 062 (b) 306 (c) 607

Figure 3-13: Particle kinetic energy vs. distance from beam centerline, as measured at the converter plate.



(a) 062 (b) 306 (c) 607

Figure 3-14: Phase space for yy' evaluated at the converter plates. Since variance in momentum does not effect first order transport in y, 1x magnification was achieved.

kinetic energy tend to be located in positive x relative to the beam centerline, which means on average the lower energy particles were over-focused by the dipole magnet.

These simulations, which were verified with those that follow in section 3.4, presented some important design challenges. While it was originally expected that it would be possible to maintain 1x magnification of the beam at the converter plates, the low energy tail makes this nearly impossible without the use of another focusing component in the beam-line. While simulations were carried out for using a solenoid or quadrupole magnet to focus the beam, it was determined that such a magnet would be too large to fit in the already crowded beam line. This meant that the steering magnets would have to be built with large apertures to accept the low energy beam tail.

3.4 Opera 3D Beam Transport

While the 1D and 1D+ methods of TRANSPORT and *pybeam1d* agreed well, it was still important to verify that higher order effects in the BBGS magnets would not have adverse effects on the beam transport. The inclusion of higher order effects can be approximated in 1D using correlations, but a more thorough alternative is to create a full 3D finite-element solution for the magnetic field of the transport system.

Since the detailed 3D design of the component magnets is covered in chapter 4, the dipole and steering magnet 3D designs are taken as given in this analysis.

After designing the magnets in 3D using *Opera* (Cob, 2009), the next step in creating a beam simulation was to convert the input files provided by AES. For the electron simulations in *Opera 3D*, input files must specify electrons with six variables: $(x, y, z, \theta, \phi, V)$, where θ is the rotation about the z axis, ϕ is the rotation about the *new* x axis, and V is the voltage used to accelerate the electrons. The raw electron data from AES was converted into the correct format for *Opera-3D* using the geometry outlined in figure 3-15. Having calculated the angles x' and y' as described in the

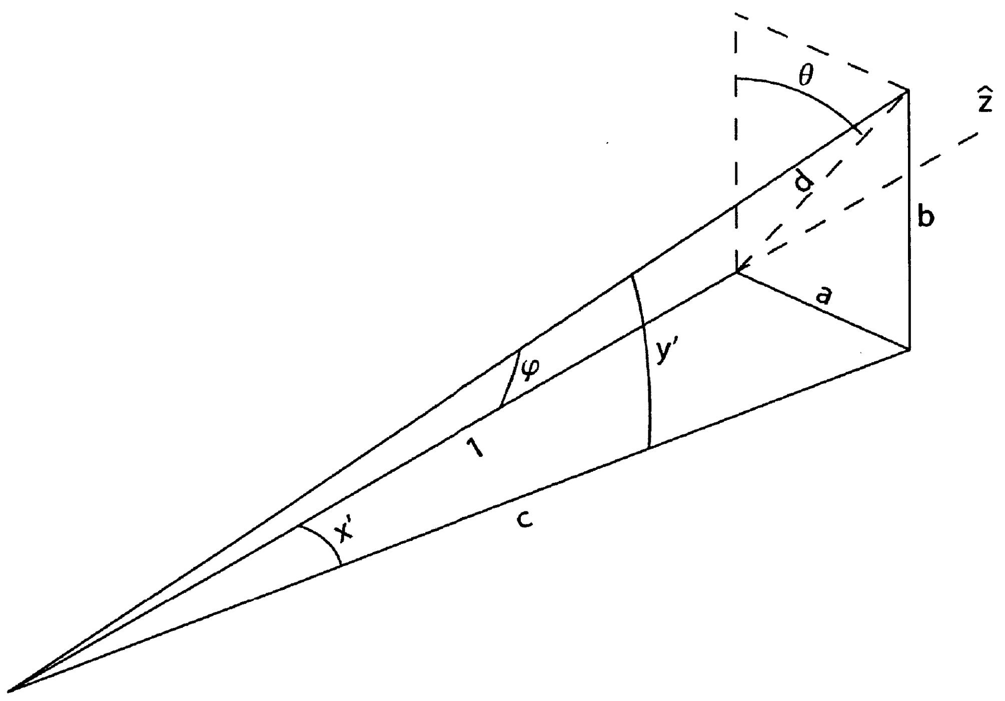


Figure 3-15: Geometry used for conversion from x' and y' to θ and ϕ.

previous section, next the angle θ was calculated. From figure 3-15, it is clear that

$$\theta = \arctan\frac{a}{b} \tag{3.64}$$

and also, from simple trigonometry, that $a = \tan x'$. So, calculating the value of b:

$$b = c\tan y' \tag{3.65}$$

and for c:

$$c = \sqrt{1+a^2} = \sqrt{1+\tan^2 x'} = \frac{1}{\cos x'} \tag{3.66}$$

So, by substituting into equation 3.64:

$$\theta = \arctan\frac{\sin x'}{\tan y'} \tag{3.67}$$

To find ϕ in terms of x' and y', we observe:

$$\phi = \arctan d \tag{3.68}$$

and find:

$$d = \sqrt{a^2 + b^2} = \sqrt{\tan^2 x' + \frac{\tan^2 y'}{\cos^2 x'}} \tag{3.69}$$

substituting into equation 3.68 to find:

$$\phi = \arctan \sqrt{\tan^2 x' + \frac{\tan^2 y'}{\cos^2 x'}} \tag{3.70}$$

Once the input parameters for each particle were calculated, it was necessary to create an *Opera* Command Input (comi) file that would run the beam in the *Opera Post-Processor*. This was an involved process and hence the interfacing *Python* script is included in Appendix A.

The move to 3D computation of fields and trajectories complicates more than just geometry. Before running each 5000 particle beam sample, it was necessary to fine-tune the dipole magnet field such that a particle at average beam energy would intersect with the center of the converter plate. This was an iterative process which involved visual inspection of the trial beam. The most efficient method was to find an electron kinetic energy (T) which intersected the center of the target, then in the next iteration, scale the magnet current (I) by a factor:

$$\frac{I_{\text{new}}}{I_{\text{old}}} \approx 0.8 \frac{T_{\text{desired}}}{T_{\text{measured}}} \tag{3.71}$$

Learned iterative tricks such as this one became very important while working in 3D as the field computations run for minutes rather than milliseconds!

After field calibration, the 5000 particle beam sample was run using a 'comi' file which defined the simulation. To extract the phase space at the converter plate, the *Opera Post-Processor* 'intersect trajectories with patch' function was used. This function provides x, y, z and v_x, v_y, v_z in the coordinate system of the Opera solution file, which for the sake of field symmetry is not the same coordinate system required by the phase space.

The beam coordinate system was defined by first finding the beam direction, $\hat{\mathbf{z}}$:

$$\hat{\mathbf{z}} = \left\langle \frac{\mathbf{v}}{|\mathbf{v}|} \right\rangle \tag{3.72}$$

where $\mathbf{v}$ is the velocity vector in *Opera* coordinates. Now, using the notation that $\hat{\mathbf{Y}}$ is the y direction in *Opera*, the beam $\hat{\mathbf{x}}$ direction was given by:

$$\hat{\mathbf{x}} = -\hat{\mathbf{z}} \times \hat{\mathbf{Y}} \tag{3.73}$$

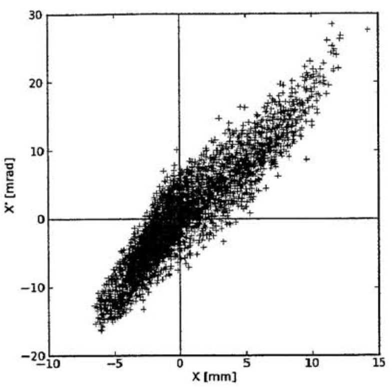
Finally, the beam $\hat{\mathbf{y}}$ direction could be found:

$$\hat{\mathbf{y}} = \hat{\mathbf{z}} \times \hat{\mathbf{x}} \tag{3.74}$$

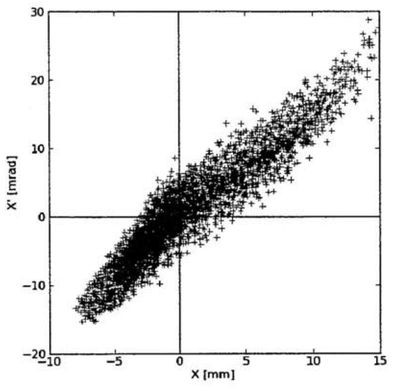
Once the beam coordinate directions were determined, each particle position and momentum was normalized to the average beam position and then dotted into the new coordinate directions to find the final phase space.

The results of the *Opera 3D* beam simulation for beams 062, 306, and 607 are compared to *pybeam1d* results in figures 3-16, 3-17, and 3-18, respectively. The phase space for xx' is remarkably similar between the two models. This means that the dipole was designed well in that higher order effects are not important for the x transport. It also shows the true strength of the 1D method. A 1D solution is possible for 50,000 particles in less than one second, where a 3D solution for only 5000 particles takes a total of approximately 15 minutes, not including the time to design and solve the magnet in 3D, which took many weeks.

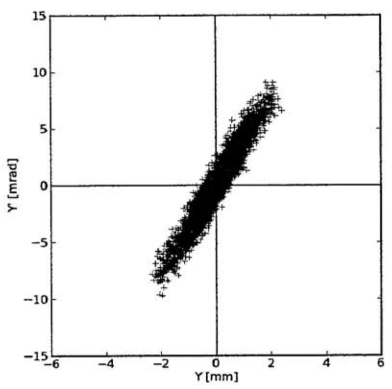
Clearly the first order methods did not predict the same yy' phase space as the 3D trajectory integration. This means that higher order effects dominate the y transport. This is a result of the edge angles which focus the beam in the y direction. First order methods treat this focusing, but not with enough detail to capture the entire effect. The modification of the *pybeam1d* transport with higher order effects is treated in



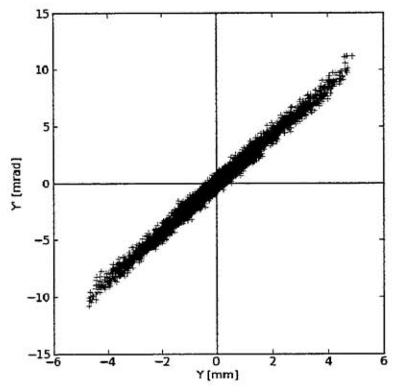
(a) Opera xx'



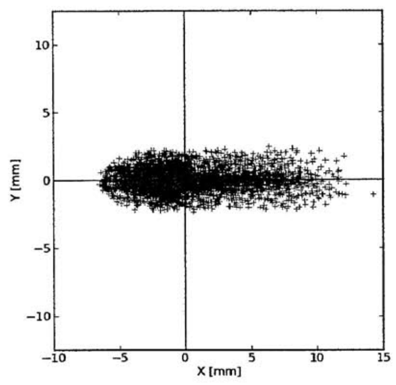
(b) pybeam1d xx'



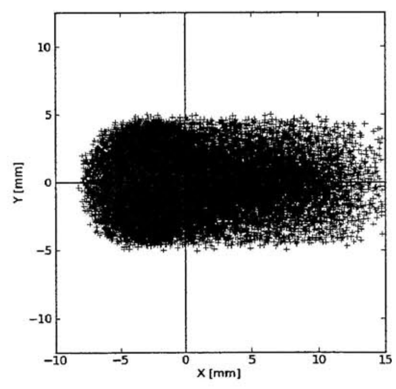
(c) Opera yy'



(d) pybeam1d yy'

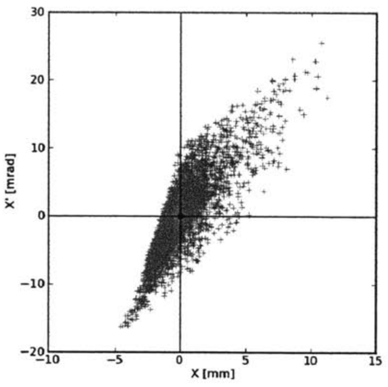


(e) Opera xy

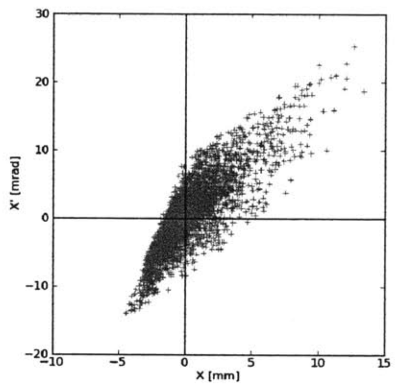


(f) pybeam1d xy

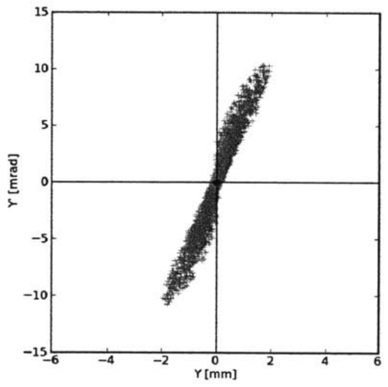
Figure 3-16: Comparison of *Opera* and *pybeam1d* phase space at the converter plates for beam 062.



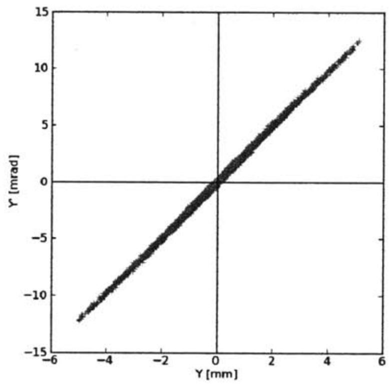
(a) Opera xx'



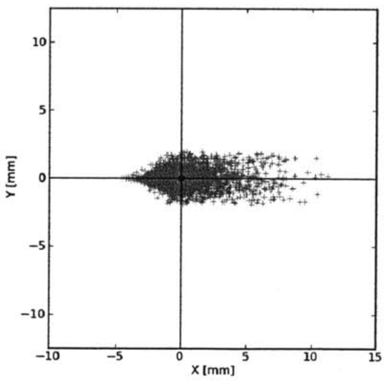
(b) pybeam1d xx'



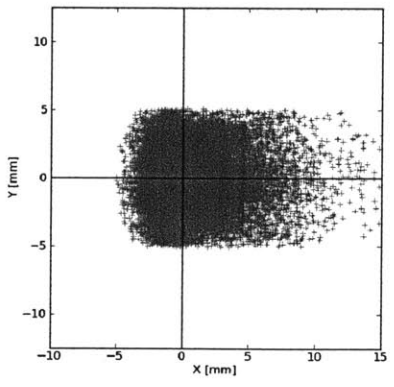
(c) Opera yy'



(d) pybeam1d yy'

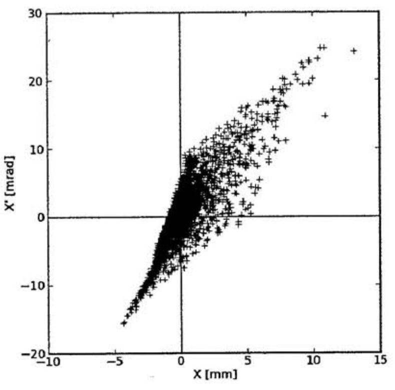


(e) Opera xy

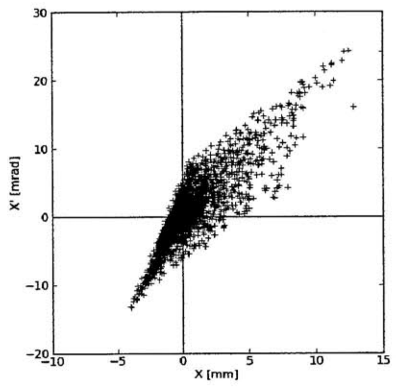


(f) pybeam1d xy

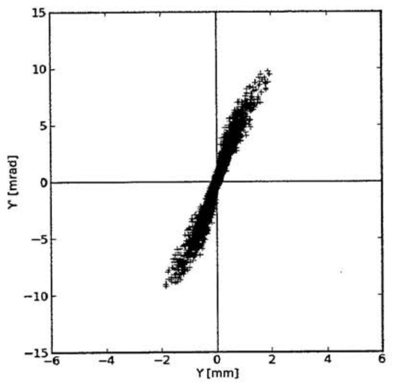
Figure 3-17: Comparison of *Opera* and *pybeam1d* phase space at the converter plates for beam 306.



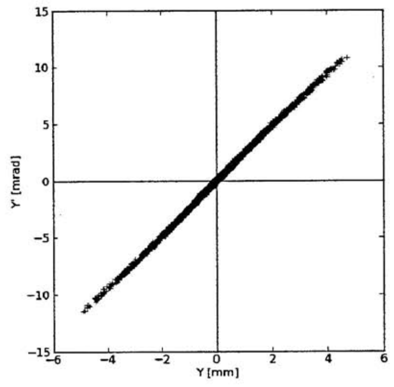
(a) Opera xx'



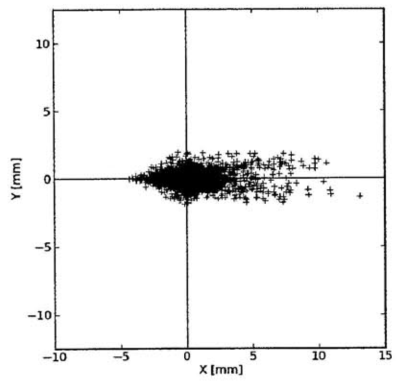
(b) pybeam1d xx'



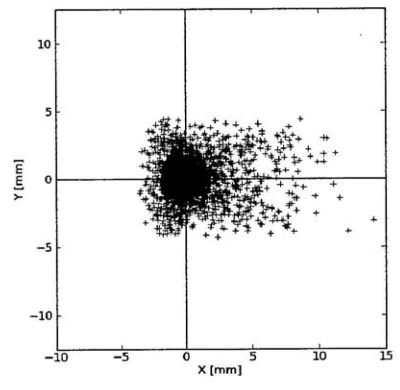
(c) Opera yy'



(d) pybeam1d yy'



(e) Opera xy



(f) pybeam1d xy

Figure 3-18: Comparison of *Opera* and *pybeam1d* phase space at the converter plates for beam 607.

more detail in chapter 4.

The 3D beam transport simulations were important for the BBGS design because they helped to validate previous calculations while providing insight into possible higher order effects in the beam transport. With three methods in reasonable agreement with explainable differences, it was determined that the BBGS magnets could be designed with tight tolerances to the beam simulations.

3.5 Estimation of Radial Space-Charge Effects

Throughout all of the BBGS beam transport simulations, the effects of space-charge on the beam dynamics were ignored. Space-charge effects are the effects of the electric fields generated by the beam on the beam. For this reason they are sometimes referred to as 'self-fields.'

A simple set of calculations was performed to validate the assumption that self-fields are not important in the BBGS beam transport. For these calculations, a cylindrical coordinate system with the beam direction in positive $\hat{\mathbf{z}}$ is used. Assuming an axially symmetric and uniform beam exiting the LINAC with beam envelope, R, the governing equation for the motion of the radial envelope is derived by Humphries (1990) to be:

$$\frac{d^2R}{dz^2} = \frac{K}{R} \tag{3.75}$$

where K is the generalized perveance, given by:

$$K = \frac{eI}{2\pi\epsilon_o m_o(\beta\gamma c)^3} \tag{3.76}$$

where e is the electron charge, I is the beam current, and m_o is the rest mass. This calculation would underestimate the space-charge effects in the BBGS, since the total beam current is actually generated by several short pulses, not an infinitely long continuous cylindrical distribution of charge. To make this an overestimate of space-charge effects, the BBGS current was multiplied by the reciprocal of its duty factor. This calculates the space charge as if the electron beam was actually a continuous beam with a charge density of that found in the beam bunches.

Rather than taking the time to integrate the equation of motion, a worst-case calculation was performed. This assumed a constant $\frac{d^2R}{dz^2}$ given by the initial value of $R = R_o$, applied over the entire beam path. In this worst-case, the final beam radius is given by:

$$R_f = \frac{1}{2}\frac{d^2R_o}{dz^2}z^2 + R_o = \frac{1}{2}\frac{K}{R_o}z^2 + R_o \tag{3.77}$$

This gives the fractional change in beam envelope:

$$1 - \frac{R_f}{R_o} = (2.0 \times 10^{-4}, 1.7 \times 10^{-6}, 1.7 \times 10^{-7}) \tag{3.78}$$

for the 6.2, 30.6, and 60.7 MeV beams, respectively.

This calculation showed that the space-charge effects in the BBGS electron beam could be reasonably neglected. The calculation was an overestimate of the effects as it overestimated both the current density in the beam and the accelerating forces.

3.6 Longitudinal Space Charge Effects

A calculation of space charge effects in the beam direction was not performed.

Chapter 4

Magnet Design

The design of the component magnets for the ISIS BBGS was an iterative process which could not and did not happen in isolation from the beam transport design. The process taken was to first generate an acceptable beam transport solution in 1D. Next, magnets were designed to replicate this solution and verify it in 3D. Finally, based on the results of the 3D analysis, changes were made to the 1D transport solution to begin a new design iteration. For clarity, the component designs are presented separately in this chapter as a series of independent design iterations.

In addition to general space constraints imposed on the BBGS, there were other requirements placed specifically on all magnetic components. All magnetic components were required to be either air or water cooled and hence non-superconducting. It was also a requirement that whenever possible, the magnetic components should be constructed from industry standard parts and materials. These constraints led to the development of iron core electromagnets with hollow copper conductor windings.

4.1 Dipole Magnet

Once the basic design parameters for the BBGS dipole were verified through TRANSPORT and *pybeam1d* calculations, the next task was to create a physical design which would meet those parameters.

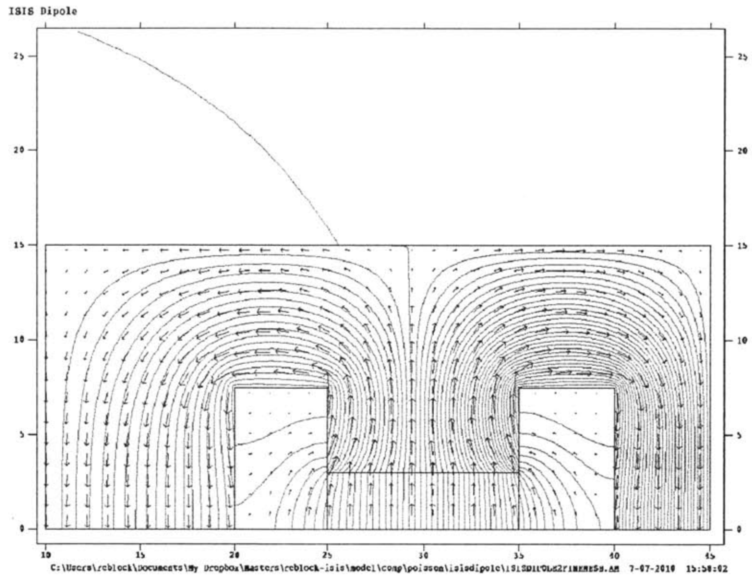


Figure 4-1: Cross section of dipole magnet version 1. Solved using Poisson 2D.

2D Magnet Design The first step in the physical magnet design was to create a simplified solution for the magnetic fields using *Poisson* (Los, 2010). The *Poisson* 2D code solves a finite-element version of the magnetostatics Poisson equation for both x, y and r, z 2D symmetry. For the dipole magnet, r, z symmetry was used with the z direction in the direction of the bending field and r direction in the negative direction of acceleration. This corresponds to a cylindrical coordinate system in which the charged particle moves along the θ direction inside the magnet.

In order to begin iterating upon a magnetic field design, a starting point was needed. Before details of the beam were known, it was determined that a safe design would use a relatively large aperture in order to accept many different sized beams. A cross-sectional view of the initial dipole magnet design is shown in figure 4-1. The dipole aperture was set to ± 3 cm vertical and ± 5 cm horizontal. While this large aperture allowed for potentially large beams from the LINAC, the drawback came with the required total current and current density in the coils. The dimensions of the dipole yoke (the iron surrounding the coils) were created by attempting to balance the magnetic flux through the inner and outer return. The purpose of this balance was to maintain a central field which was as symmetric and uniform as possible about the beam centerline. The yoke was also designed such that the iron would be below saturation near the outside edges of the magnet, and hence the magnetic field lines would be well contained by the yoke.

The goal of the first 2D dipole design, 'BND1', was to achieve a central field that could bend the 60.7 MeV electron beam at a radius of 30 cm. To calculate the required field, we combine the Lorentz force with centripetal acceleration:

$$F_r = ma_r \tag{4.1}$$
$$qvB = \gamma m_o \frac{v^2}{r} \tag{4.2}$$
$$B = \frac{1}{0.300} \frac{p}{qr} \tag{4.3}$$

which gives the required magnetic field in kG for momentum in units of MeV/c, radius in units cm, and charge in units relative to one electron charge. For the 60.7 MeV electron beam, a magnetic field of 6.8 kG is required in the dipole magnet.

To achieve this central field, the BND1 design had a total current per coil equal to 16.2 kiloAmp-turns. The unit of Amp-turns denotes the total current in Amps required for the coil. Since the coil is wound using small conductor, the current in the conductor times the number of turns will give the total current in Amp-turns (A-t). With this first solution for the magnetic field complete, it was important to estimate how practical the design would be. One of the most important considerations for design feasibility is the current density required. Current density drives conductor choice, and conductors were a limiting constraint on the magnet design.

Choice of Conductors Some simple yet important calculations were performed in order to size the conductors for the BBGS dipole magnet. This process began with assumptions regarding the maximum allowable current for different conductor designs. These assumptions were drawn using both industry sources as well as the organizational knowledge of PSFC engineers. For solid copper conductor with air-cooling, the American Wire Gauge (AWG) tables for limiting currents in packed motor windings were used (McGahee, 1998).

Estimates for maximum current densities for water-cooled hollow copper conduc-

Table 4.1: Maximum current for several choices of conductor. American Wire Gauge (AWG) numbers are derived from reference values for creating packed motor windings (McGahee, 1998). Hollow conductor is square with round hole.

Type	OD [mm]	ID [mm]	Area [mm^2]	I_{max} [A]	Packing	J_{coil} [A/mm^2]
24 AWG	0.51	n/a	0.20	0.81	0.7	2.83
20 AWG	0.81	n/a	0.52	2.0	0.7	2.69
18 AWG	1.02	n/a	0.82	3.25	0.7	2.77
10 AWG	2.59	n/a	5.27	20.8	0.7	2.76
Hollow	4.0	2.5	11.1	88.7	0.69	3.84
Hollow	6.35	3.15	30.0	240	0.74	4.43

tor are hard to find in the literature. This is mostly because the maximum current density is highly dependent on the water flow rate that can be achieved. However, an approximate value for a reasonable current density is given by Tanabe (2005) to be $j = 10$ A/mm^2. Based on the experience of Dr. Antaya, this was reduced to approximately 8 A/mm^2.

These estimates were combined to create a reference table, shown in table 4.1, which guided the design of all BBGS magnet coils. For example, the BND1 coil is 50 mm wide and 75mm tall. Based on this coil area of 3750 mm^2, the current density, J_{coil}, is equal to 4.32 A/mm^2. This led to the selection of the 6.35 mm (0.25 in) hollow copper conductor for the first revision of the BBGS dipole. While this simplified analysis did not complete the dipole coil design, it was conservative enough such that a more detailed analysis could be delayed until the design was more complete.

3D Field Verification After developing a basic field design in 2D, each magnet solution was verified using the *Opera 3D TOSCA Magnetostatic* code. This move to three dimensions was important for several reasons. First, it created an extra check to ensure errors were not made in developing the 2D field solution. Second, it allowed for the characterization of coil and magnet end effects. Finally, 3D field models allowed for beam transport in 3D, the benefits of which are described in section 3.4.

As always, the first step to creating a new and more complex model was to verify

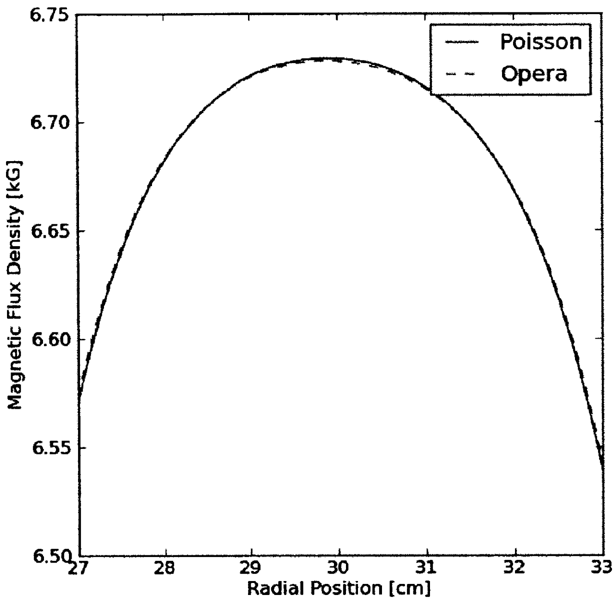


Figure 4-2: Comparison of central field region for *Poisson* and equivalent *Opera* model.

that the complex model agreed with the original model in the limiting case. This meant creating an axially symmetric *Opera* 3D model of the dipole magnet and comparing the field results to those found using *Poisson.* The results of this analysis for the BND1 geometry are shown in figure 4-2. The calculations for the central field of the BND1 model agreed within 0.01% for peak fields and 0.05% point-wise. Based on the element size and field gradients, these differences are within the expected errors of the finite element solution. However, it is important to note a marginal difference between the definition of the *Poisson* and *Opera* models. In *Poisson*, the conductors can be specified to lie exactly adjacent to the yoke. In *Opera*, conductors must be defined such that they are completely within a region which contains no ferromagnetic material. In practice this means that conductors in *Opera* must be defined with a small (1 mm) gap between the conductor and the yoke. Since the magnet is an iron-dominated magnet, these small differences in coil definition were not expected or observed to have a significant effect on the calculated fields.

Once it was determined that the Opera solver was working correctly, a full 3D model of the BND1 geometry was created.

Field Analysis in 3D The ability to generate magnetic fields in 3D allowed for a more detailed study of the coil design as well as higher order effects near the magnet edges. For each design change, 3D beam transport results were compared with 1D results. Discrepancies between the results were then investigated by probing the 3D fields more carefully.

The progression of the BBGS dipole 3D magnet design is shown in figure 4-3. There were 5 major design revisions for the BBGS dipole magnet, referred to as BND1, BND2, ... BND5. Each design achieved the basic requirement for delivering the electron beam to the converter plate, but as the design progressed efforts were made to minimize both the power requirements and mass of the yoke and coils.

The primary design changes for each model are summarized below:

- BND1-BND2: Optimize yoke for field uniformity, minimize length of saddle coils
- BND2-BND3: Minor revisions to coils and yoke
- BND3-BND4: Redesign with small aperture and racetrack coils. Performed once beam parameters were known with more certainty.
- BND4-BND5: Coil adjustment to simplify winding and minimize power

In designs 1-3, the dipole had a wide aperture and saddle-shaped coils. The reason for this was that the size of the electron beam was very uncertain in the early stages of design. This very large aperture coupled with the requirement for non-superconducting coils made it impossible to achieve the required fields without using saddle-shaped coils. A downside to this design was that it was not possible to include a straight-through beam port in the dipole, which would have made LINAC diagnostics much more difficult.

Designs 4 and 5 created a more compact dipole which was designed to the exact specifications of the LINAC electron beam. The final design uses racetrack coils which allow for a straight-through beam port. It was possible to reduce the cross section of the coils due to the small pole gap, which was reduced to ±1.5 cm from ±3.0 cm, as well as the reduced aperture width (down to ±3.75 cm from ±5.0 cm).

BND1 Field Analysis The results of the 1D and 3D beam transport for BND1 are shown in figure 4-4. From the beam transport, it was clear that higher order effects were important in the BBGS transport. The x beam envelope was more uniform than expected from 1D simulations, and the y beam envelope was about 2x the width of 1D transport results.

The first step was to inspect the electron trajectories in order to understand how they may differ from idealized results. As shown in figure 4-4 c, the fringing field of BND1 acts at a significant distance from the magnet edge, resulting in a beam which travels near the inside edge of the magnet aperture. To estimate how this affects the beam phase space, it was important to examine the difference between the magnetic field near the edge of the aperture and the theoretical field index $n = 0$ field assumed in 1D calculations.

A plot of the magnetic field across the magnet aperture evaluated at the center of the magnet is shown in figure 4-5. Clearly, in the range $r = [25, 28]$ cm the magnetic field is not a uniform field in r. One way to measure the non-uniformity is the field index, introduced in chapter 3:

$$n = -\frac{r}{B_z}\frac{\partial B_z}{\partial r} \tag{4.4}$$

Evaluating the field index at $x = 26$ cm gives $n \approx -2.0$. Recall the equation of motion for the z (or y in Cartesian coordinates) direction:

$$\frac{d^2y}{dS^2} + n\frac{y}{r_0^2} = 0 \tag{4.5}$$

Where S is the path length traveled in the magnet and r_0 is the bending radius. Since here $n < 0$, rather than being sinusoidal the solutions to the equation will be

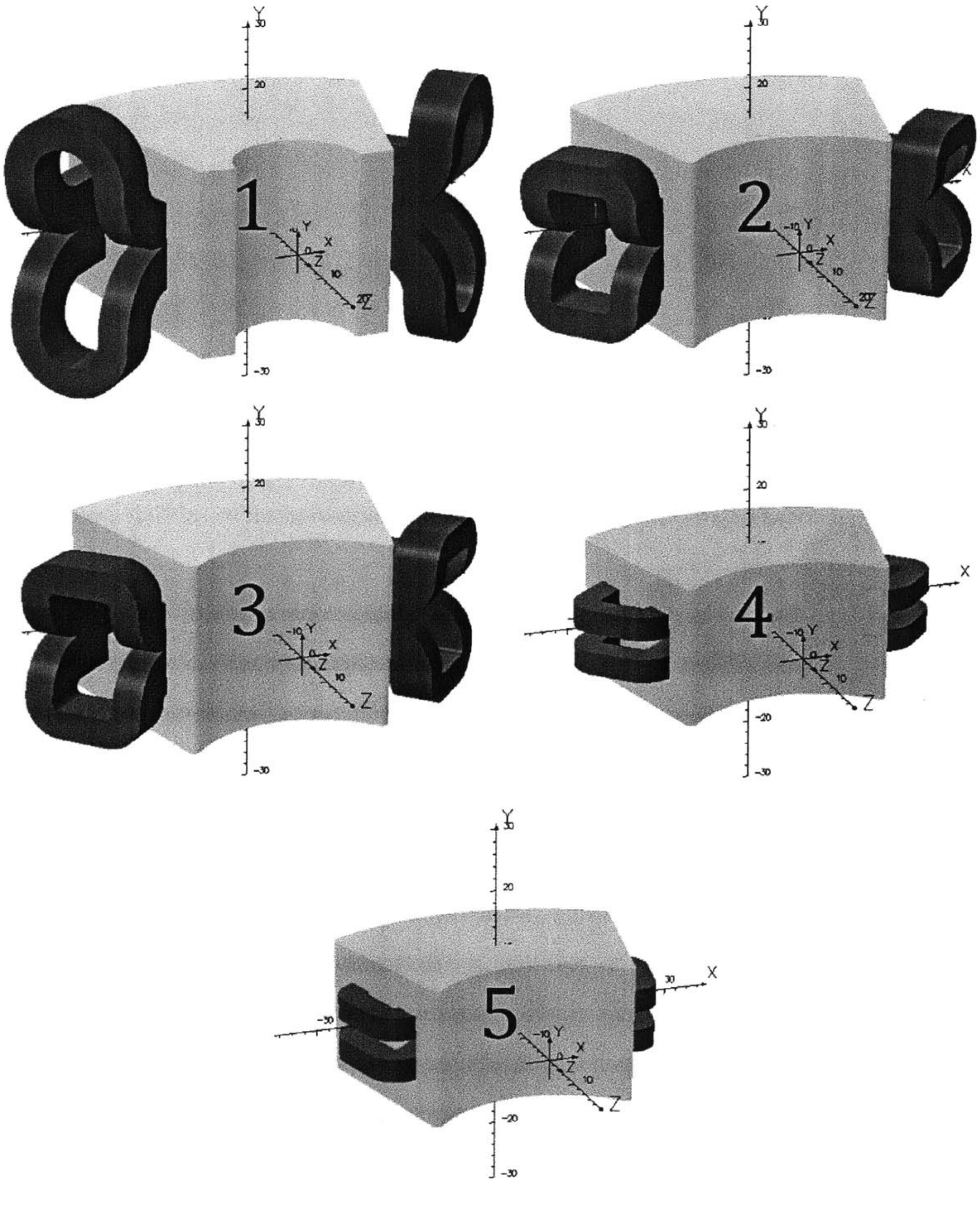
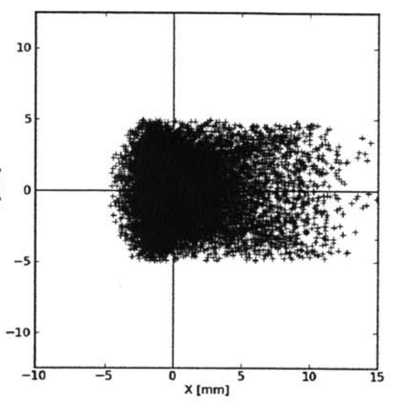
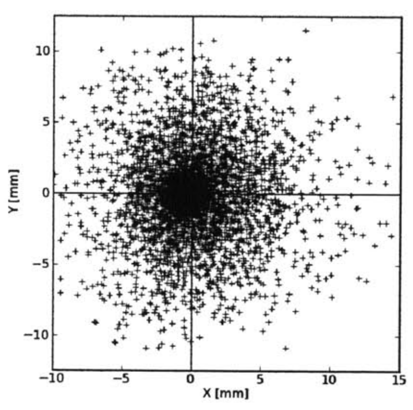


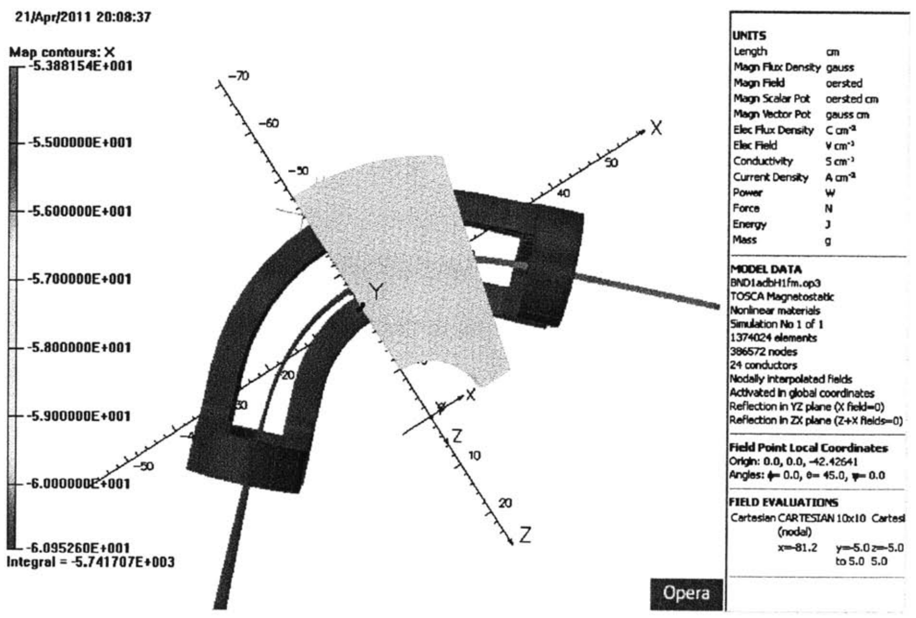
Figure 4-3: Evolution of dipole magnet design.



(a) *pybeam1d*



(b) *Opera 3D*



UNITS

Length	cm
Magn Flux Density	gauss
Magn Field	oersted
Magn Scalar Pot	oersted cm
Magn Vector Pot	gauss cm
Elec Flux Density	C cm^{-2}
Elec Field	V cm^{-1}
Conductivity	S cm^{-1}
Current Density	A cm^{-2}
Power	W
Force	N
Energy	J
Mass	g

MODEL DATA
BND1adbH1fm.op3
TOSCA Magnetostatic
Nonlinear materials
Simulation No 1 of 1
1374024 elements
386572 nodes
24 conductors
Nodally interpolated fields
Activated in global coordinates
Reflection in YZ plane (X field=0)
Reflection in ZX plane (Z+X fields=0)

Field Point Local Coordinates
Origin: 0.0, 0.0, -42.42641
Angles: ϕ= 0.0, θ= 45.0, ψ= 0.0

FIELD EVALUATIONS
Cartesian CARTESIAN 10x10 Cartesi
(nodal)
x=-81.2 y=-5.0 z=-5.0
to 5.0 5.0

(c) *Opera* beam simulation

Figure 4-4: Comparison of beam envelope for BND1. Differences hypothesized to be a result of the non-ideal beam path.

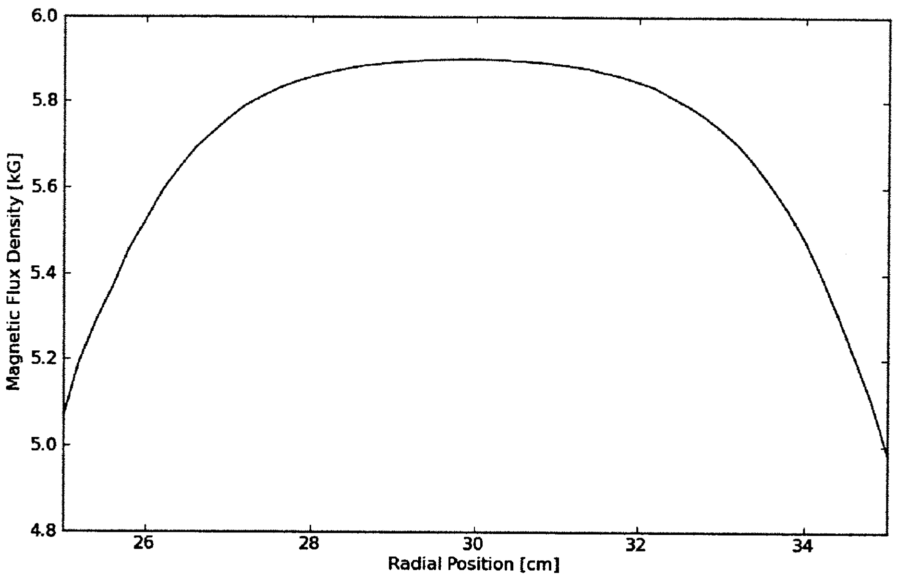


Figure 4-5: Magnetic field across dipole aperture evaluated at the magnet center.

exponential in the form:

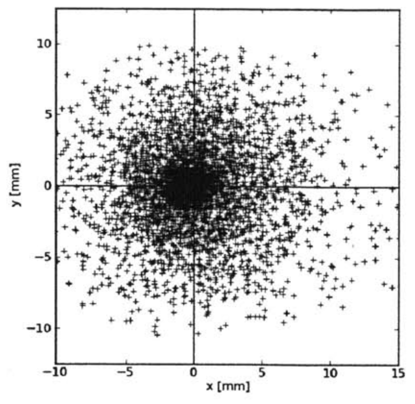
$$y(S) = a \exp \sqrt{-n}\frac{S}{r_0} + b \exp -\sqrt{-n}\frac{S}{r_0} \tag{4.6}$$

Solving for $y(0) = y_0$ and $y'(0) = y'_0$ and substituting $S = r_0\theta$, this gives $a = \frac{1}{2}(y_0 + \frac{r_0}{\epsilon}y'_0)$ and $b = \frac{1}{2}(y_0 - \frac{r_0}{\epsilon}y'_0)$. The new transfer matrix for the y direction for a dipole with negative field index is given by:

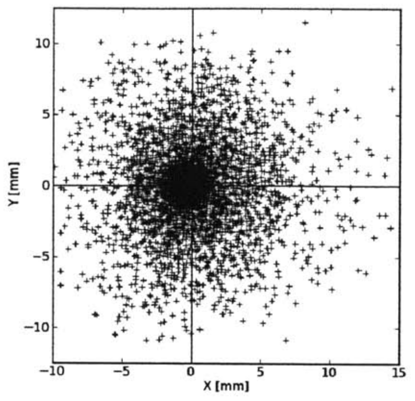
$$M_{D,y} = \begin{pmatrix} \cosh \epsilon\theta & \frac{r_0}{\epsilon} \sinh \epsilon\theta & 0 \\ \frac{\epsilon}{r_0} \sinh \epsilon\theta & \cosh \epsilon\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.7}$$

where here $\epsilon = \sqrt{-n}$ and sinh, cosh are the hyperbolic sine and hyperbolic cosine functions.

The transfer matrix for a negative field index was implemented in the *pybeam1d* matrix code and the results were compared to those from Opera. Since the electron trajectory is in a high field gradient only in the central region of the dipole, the $n = -2$ index is only used for a section $\theta = \pi/8$ within the *pybeam1d* code. A



(a) *pybeam1d* with n



(b) *Opera 3D*

Figure 4-6: Phase space for 3D transport and 1D transport with adjusted field index.

comparison of the phase space for each case is shown in figure 4-6. While there were still differences between the 1D and 3D phase space, this analysis showed that much of the non-ideality in the 3D transport resulted from the field gradient near the inside edge of the magnet aperture. This raised some concerns about how confident we could be with the results, since the magnetic field errors in a finite element solution are highest where the field gradients are also high. This meant that it would be ideal in future iterations to shift the beam transport as close to the center of the magnet aperture as possible.

BND2 and BND3 Field Analysis The primary driver for the changes between BND1 and BND2 was to shift the electron beam transport such that the beam would remain closer to the center of the magnet aperture where the field is known with better accuracy. Once more information was learned about magnet windings and the minimum bending radius of the conductors, the coil geometry was modified to be as compact as possible. It was predicted that this modification in coil geometry would reduce the extent of the fringing fields and hence shift the electron transport towards the center of the aperture.

However, the results of the analysis for BND2 were nearly identical to BND1 (and hence are not repeated here). In order to investigate why this may be, the fringing

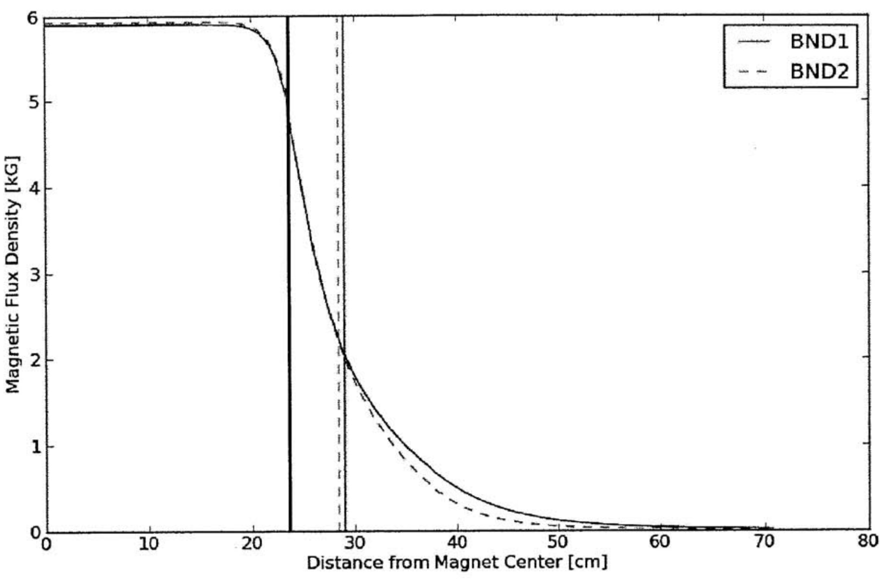


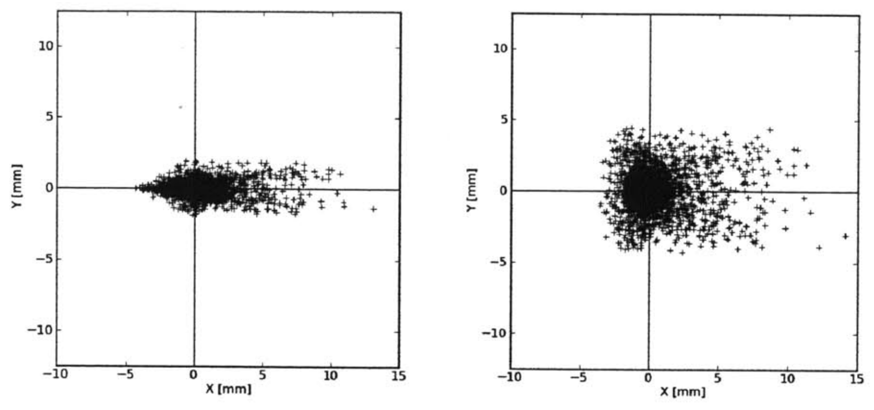
Figure 4-7: Effective length calculation for BND1 and BND2. Thick black line represents magnet edge.

fields were compared between the two models. As shown in figure 4-7, a calculation of the effective length of the bending magnets gives nearly identical results. The effective length is given by:

$$S_{\text{eff}} = \frac{1}{B_0} \int B_y \, dl \tag{4.8}$$

The effective length of the magnet extended approximately 4.8 cm beyond the magnet edge for BND2 and 5.5 cm beyond for BND1. This gave a total effective length of the magnets which was approximately 20% longer than the yoke.

One proposed solution to shifting the beam transport was to shift the dipole magnet in the beam-line such that the electron beam enters the fringing field approximately 1.5 cm to the right of the aperture center. This was explored using BND3, and it was found that this could indeed bring the 3D transport results back in agreement with 1D transport. There were a few reasons why this did not resolve the dipole design. First, detailed knowledge of the beam size gave the opportunity to reduce the total size and power of the dipole magnet. Another reason for opting not to shift the beam was that the fringing field could be significantly altered by field errors introduced in the yoke manufacturing and coil winding process. It was determined



(a) Opera xy (b) pybeam1d xy

Figure 4-8: Comparison of *Opera* and *pybeam1d* beam cross section at the converter plates for beam 607, model BND5.

that in order to be robust, the design should not rely heavily on higher order field effects.

BND4 and BND5 Field Analysis Since the only difference between BND4 and BND5 was a small change in coil geometry, the field analysis and beam transport results were nearly identical. Hence, only the results for BND5 are presented here.

The 3D beam transport results for BND5 were presented in section 3.4. As a reminder, the beam cross section at the converter plate for 1D and 3D simulations are shown in figure 4-8. Here the beam trajectory remained directly in the center of the magnet. The reason for this is that the effective length of the BND5 was only 1.8 cm beyond the magnet edge, where for the BND1 and BND2 design it was of order 5 cm. Since the beam trajectory was close to ideal and the field index was $n \approx 0$ at the aperture center, the x phase space average parameters agreed within $\pm 10\%$. However, the y phase space parameters were quite different in the 1D and 3D models.

The maximum extent of the 1D beam at the converter plate was approximately 5 mm, while the extent of the 3D beam was about 2.5 mm. While this difference was not particularly important with respect to the generation of photons, it was still important to understand what caused this difference.

Since the primary driver for y transport is the field as a result of the magnet edge, the 3D fields were explored to investigate how they differ from the simple assumptions of the 1D code. The approach to exploring the fringe fields was to perform a multipole analysis of the y focusing fields.

The idea of a multipole analysis is a popular tool for field design (Reiser, 1994). By ignoring field variations along the direction of the trajectory, the vector potential in z (and hence fields in x and y) can be written in the form:

$$A_z = \sum_{m=0}^{\infty} r^m \sin m\phi \tag{4.9}$$

It can be shown that this form must satisfy Maxwell's equations. By performing a multipole analysis of the focusing fields along the magnet trajectory, it was possible to extract the quadrupole ($m = 2$) component of the fields and use this component to approximate the y focusing in 1D.

As shown in figure 4-9, the B_x fields were evaluated along a circular path with constant radius for $\phi = [0, 2\pi]$. The goal of this analysis was to extract the multipole components as a function of distance from the magnet edge, then create an equivalent 1D quadrupole focusing magnet that could represent the dipole edge. The B_x fields were evaluated from -2.0 to 7.5 cm with respect to the edge and are shown in figure 4-10.

A Fourier transform of the magnet fields gave the multipole components of the field. It can be shown that for a pure quadrupole ($m = 2$), the transverse magnetic field holding r constant will vary as $\sin n\phi$ where $n = m - 1$. Hence, the Fourier component which represents the quadrupole moment is $n = 1$. The magnitude of the quadrupole component of the field (normalized to B_0) is plotted as a function of distance from the magnet edge in figure 4-11.

The effective quadrupole of the dipole edge was represented in *pybeam1d* using

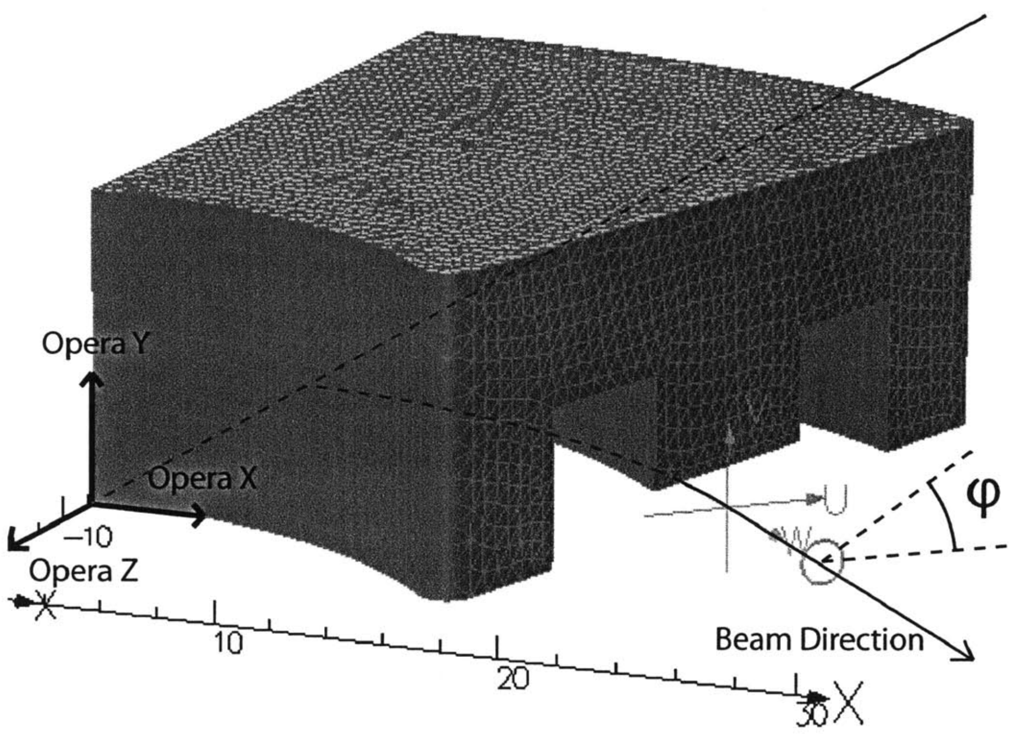


Figure 4-9: Geometry used for multipole analysis. Circle for evaluation has radius a and angle ϕ varies from 0 to 2π. Field of interest lies in the *Opera* XZ plane and is perpendicular to both Y and the beam direction.

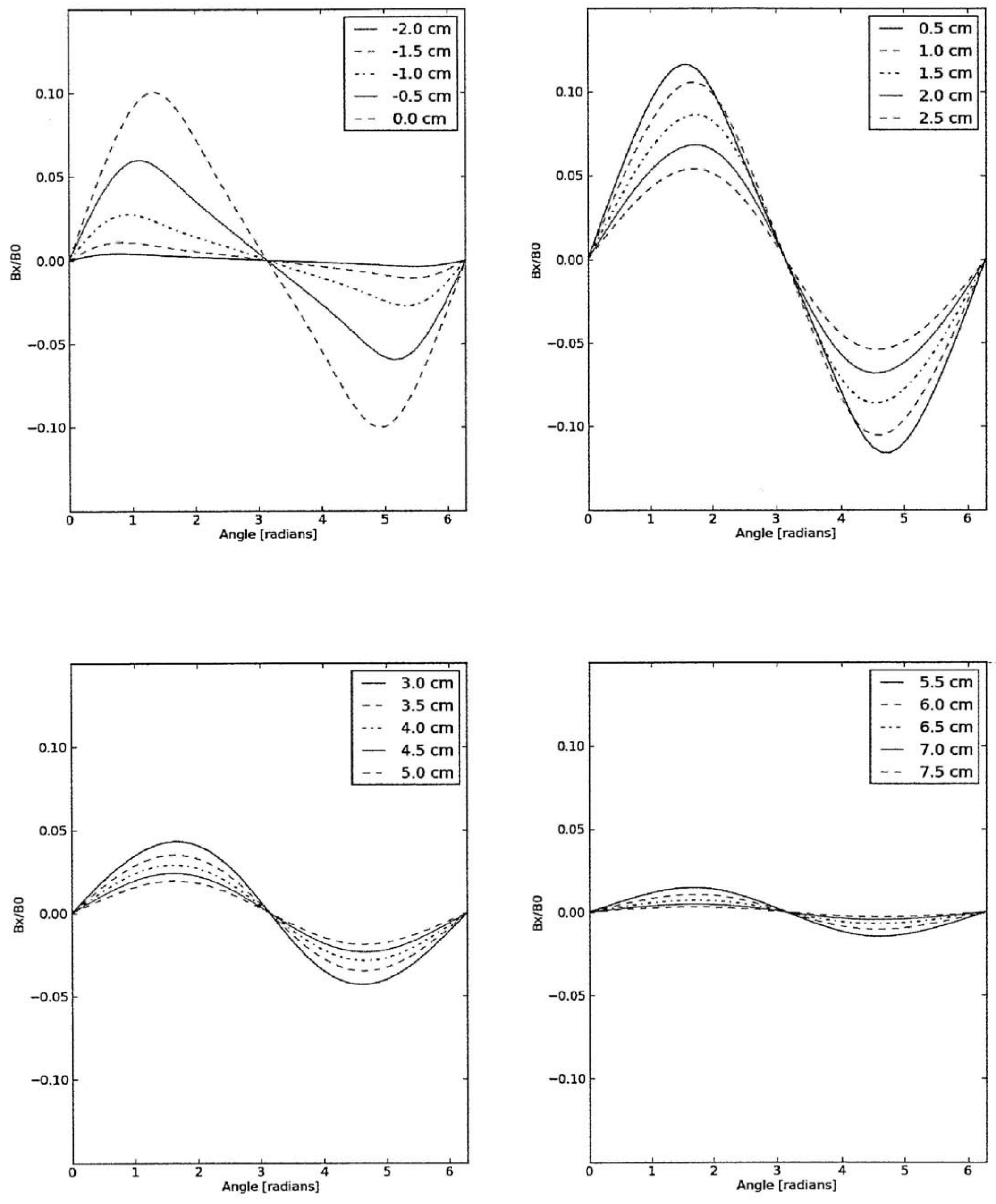


Figure 4-10: Field B_x normalized to the dipole central field B_0, evaluated at a constant beam radius $a = 0.75$ cm, shown for different distances from the magnet edge. Positive distance from edge is outside magnet.

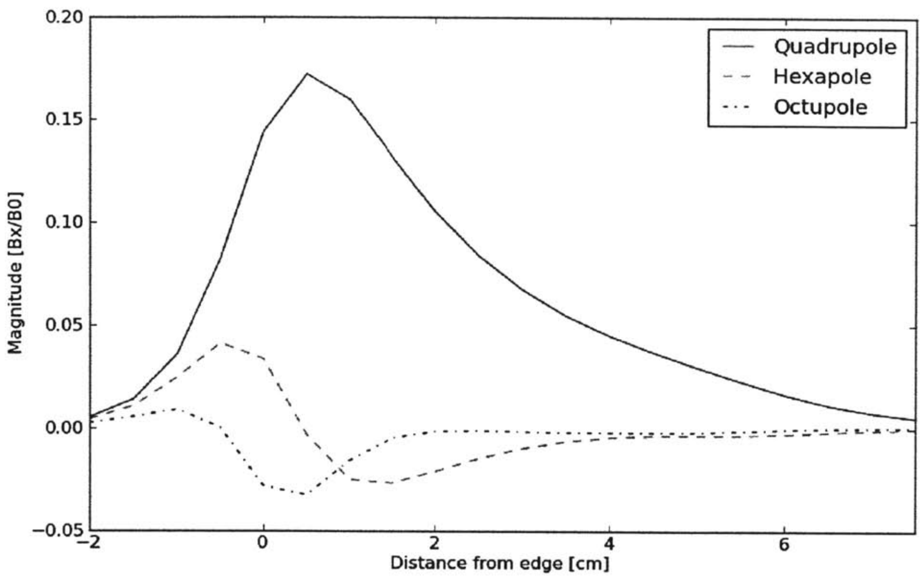


Figure 4-11: Multipole components of fields at $a = 0.75$ cm as a function of distance from magnet edge.

the transfer matrix given by the TRANSPORT manual (Sta, 1972):

$$\begin{pmatrix} x \\ x' \\ y \\ y' \end{pmatrix} = \begin{pmatrix} \cosh k_q S & \frac{1}{k_q}\sinh k_q S & 0 & 0 \\ k_q \sinh k_q S & \cosh k_q S & 0 & 0 \\ 0 & 0 & \cos k_q S & \frac{1}{k_q}\sin k_q S \\ 0 & 0 & -k_q \sin k_q S & \cos k_q S \end{pmatrix} \begin{pmatrix} x_0 \\ x'_0 \\ y_0 \\ y'_0 \end{pmatrix} \tag{4.10}$$

where S is the effective length of the quadrupole and

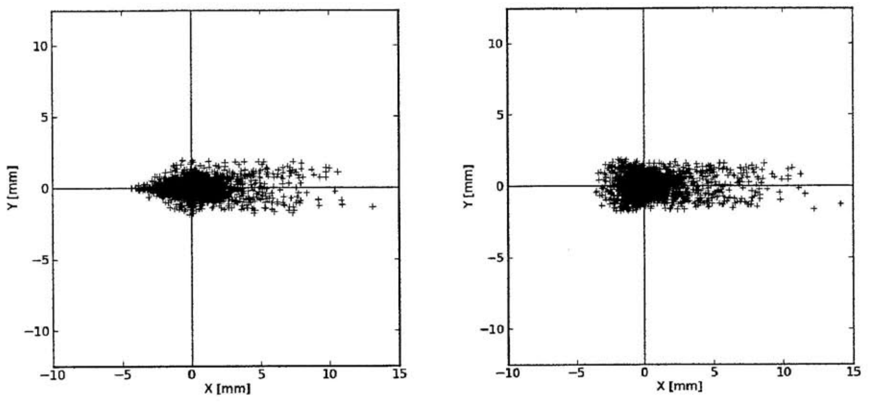
$$k_q^2 = \frac{B_a}{a}\frac{1}{B_0 r_0} \tag{4.11}$$

where B_a is the peak field of the quad evaluated at its radius a, and $r_0 B_0$ is the particle rigidity.

Using the results for BND5 gives a quadrupole magnet defined by the following parameters:

$$a = 0.75 \text{ cm}, \quad B_a = 1138 \text{ G}, \quad S = 3.55 \text{ cm}$$

which gives $k_q = 0.089$. This quadrupole was implemented in *pybeam1d* by assuming



(a) Opera xy (b) pybeam1d xy

Figure 4-12: Phase space comparison at the converter plate using the approximate edge quadrupole in 1D.

the edge fields act just outside the effective edge of the dipole magnet. The results for the effective quadrupole in 1D are compared to those from Opera in figure 4-12.

The *pybeam1d* analysis including quadrupole edges gave a maximum beam envelope in x and y within $\pm 5\%$ of the *Opera* results. While there were still higher order effects in play that the 1D transport did not capture, it was determined that the answers in 1D and 3D had converged enough to confirm the design and move forward with the procurement of the dipole magnet components.

4.2 Steering Magnets

The design of the BBGS steering magnets was one of the most challenging parts of the entire system design. The primary constraint on the steering magnets was that they be contained within the 22 cm region between the image slit system and the converter plates. Since this is a relatively small area to work with, it was first assumed that a combined function steering magnet would be the most efficient solution. After several failed attempts to design such a magnet, a design for two independent magnets was settled upon.

The progression of steering magnet design concepts is shown in figure 4-13. For clarity, magnets labeled with a number in this figure are referred to as 'STR' for steering followed by the number, eg. STR2. The goal of this discussion is to walk through each step in the design progression, showing how each design was eliminated. For all design iterations, the geometry outlined in figure 4-14 is referenced, where L is the effective magnet length, R is the bending radius of the magnet, and θ is the required deflection angle. This gives a minimum aperture width, δ:

$$\delta = L\sqrt{\frac{2(1-\cos\theta)}{\sin^2\theta} - 1} + w \tag{4.12}$$

where w is a term added to account for the width of the beam and vacuum pipe.

STR1 Field Analysis One of the simplest ways to make two-axis steering magnet is to wind resistive coils around a rectangular frame, as shown in figure 4-15. Assuming that such a magnet could take $L = 16$ cm of the available 22 cm (accounting for coil returns and flanges), the minimum STR1 half-aperture would be $(\delta_x, \delta_y) = (2.4, 0.8)$, plus approximately 1.5 cm for beam width and vacuum pipe. This gives aperture dimensions $w_a = \pm 4$ cm and $h_a = \pm 2.5$ cm.

Figure 4-16 shows a scoping calculation for a steering magnet design with these dimensions. The goal was to achieve fields $B_y = 3.26$ kG for x steering and $B_x = 1.10$ kG for y steering. For the coils shown, the current density reached 36 A/mm^2 in the x coils and 11.3 A/mm^2 in the y coils: well above the limits for any of the conductors available for the BBGS. Reducing the current densities to those achievable by simple resistive windings would require an increase in coil width of approximately 900% in the x coils and 300% in the y coils. This would in-turn require a wider total aperture and more current, which would then push the coils out further. It was determined through several iterations that the steering magnet would grow to impractical proportions before it would be possible to achieve the required fields for steering. Hence, this design was put aside before moving to 3D calculations.

STR2 Field Analysis The second steering magnet design adopted similar geometry to that of the BBGS BND1. The idea was to use water-cooled resistive coils similar to those in the dipole, but to incorporate both x and y steering into one mag-

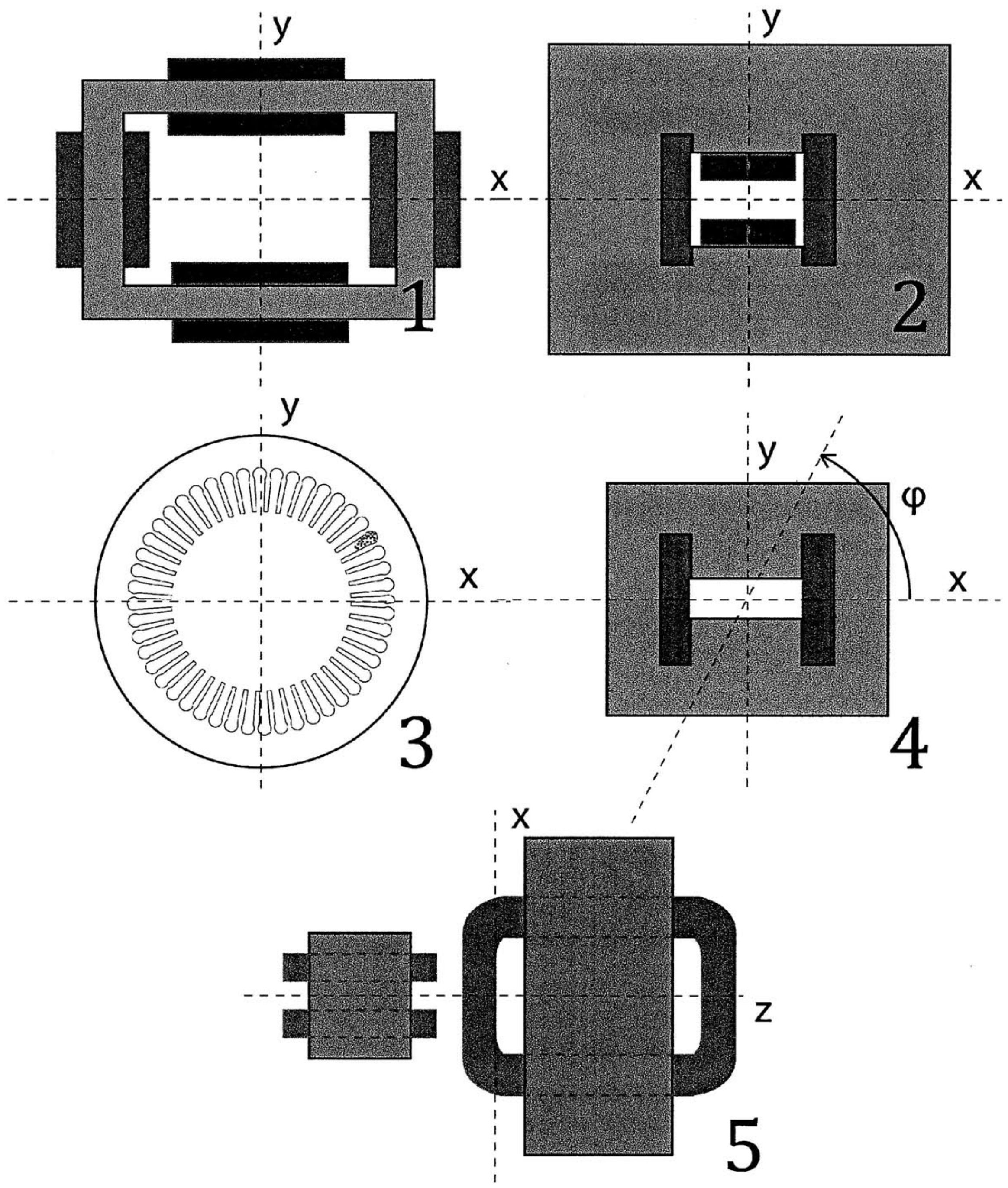


Figure 4-13: Evolution of steering magnet design. Magnets 1 through 4 are shown with cross-sectional view in xy plane. Magnet system 5 shows two independent magnets, cross section of the xz plane.

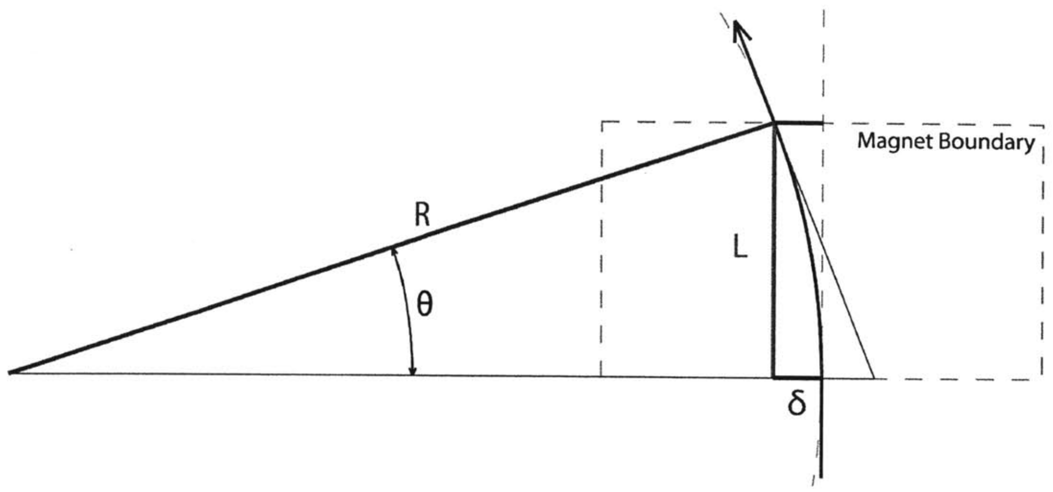


Figure 4-14: Basic steering magnet geometry used in the design process.

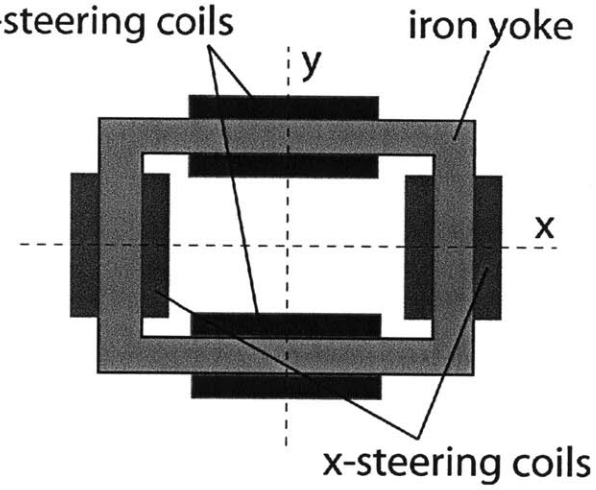
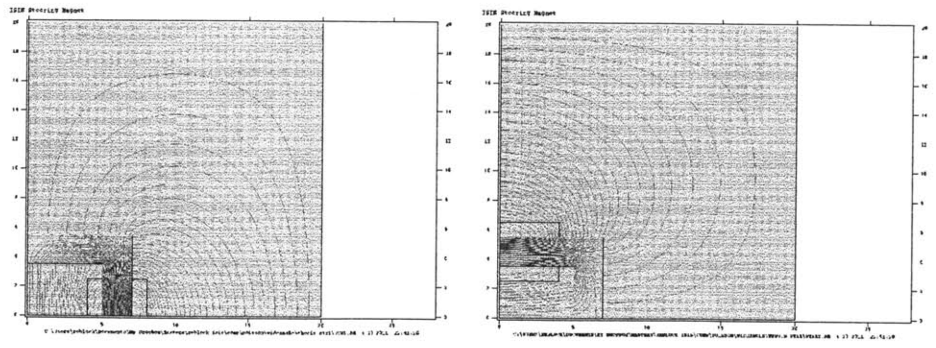


Figure 4-15: Cross-sectional view of version 1 dual-axis steering magnet design. Coils are wound into the page (beam direction) and wrap around yoke.



(a) x steering (b) y steering

Figure 4-16: Version 1 steering magnet field calculations from *Poisson* for x and y steering independently.

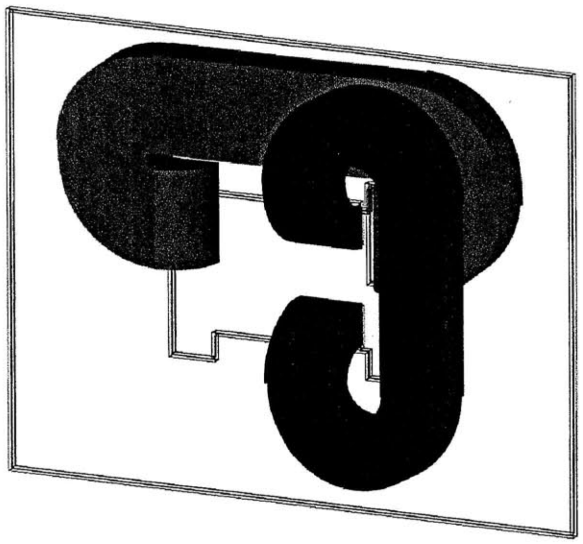
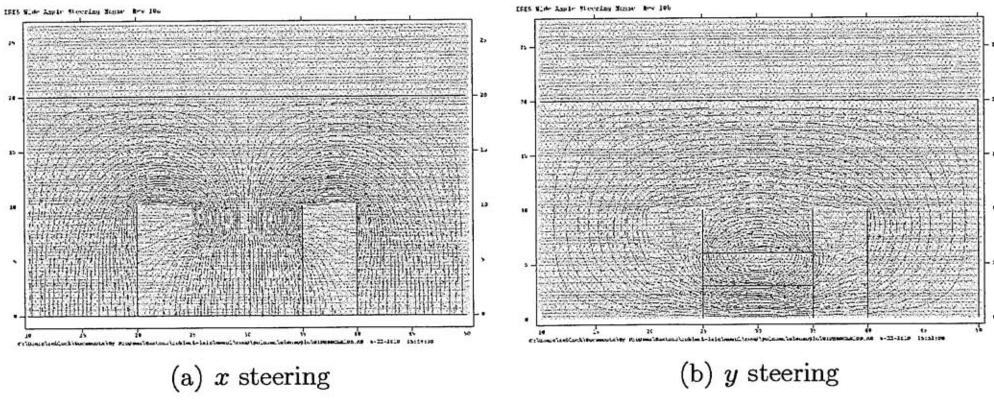


Figure 4-17: Version 2 steering magnet face with 3D coil geometry. Coil returns use much of the magnet length, similar to the BND1 coil design. Only one of two x and one of two y coil ends shown.

net. A schematic of this design layout and the proposed coil geometry is shown in figure 4-17. As before, the design of this magnet was an iterative process in which the coil area was incrementally increased until the desired maximum fields in x and y could be achieved. This involved separate calculations for each steering direction, as depicted in figure 4-18.

As the magnet grew through iterations, it became clear that such a design would not be feasible in the space allowed for the steering magnet. This was primarily due



(a) x steering (b) y steering

Figure 4-18: Version 2 steering magnet field calculations.

to the complicated coil return geometry, which would require approximately 15 cm of the 22 cm allocated for the steering magnets. This number only increases as the effective magnet length decreases, since with short length came higher current and larger coils. Hence, the STR2 design was put to rest.

STR3 and STR4 After weeks of iteration, it became necessary to begin brainstorming and testing other possible designs for the combined steering magnet. During this brainstorming period, decisions on feasibility needed to be performed quickly in order to keep the project on schedule. Steering magnets 3 and 4 are examples of designs which were rejected early in the process, but they are included here to show the types of alternatives considered before the final design was settled upon.

The idea for the STR3 design came from a literature review which turned up a paper by Benaroya and Ramler (1961) on the design of a cylindrically symmetric motor-stator type steering magnet used for deflecting deuteron beams. This design used a sinusoidal winding scheme in 4 coils to achieve uniform fields for steering. The design was used to adjust a 21.6 MeV deuteron beam up to an angle $\theta = 1°$. The magnet was created from resistive copper wire and water cooled only from the outside.

The momentum of a 21.6 MeV deuteron beam is approximately 285 MeV/c. This is approximately 4.7 times that of the electron beam. This means that for the same current and field, the magnet could deflect the BBGS electron beam at a radius $r_e \approx \frac{r_d}{4.7}$. Using the geometry from figure 4-14, $\sin\theta = L/R$, and hence:

$$\frac{4.7\, L_e}{\sin\theta_e} = \frac{L_d}{\sin\theta_d} \tag{4.13}$$

Since the magnet in Benaroya and Ramler (1961) had an effective length $L_d = 29$ cm, this gives a deflection angle for a hypothetical BBGS copy of the magnet $\theta_e \approx 2.9°$. This means that the current would need to be increased or the basic design would need to be modified significantly in order to achieve the steering required for the BBGS.

While there were many benefits to this design, the downside to the design was that it used a fairly complex winding scheme, and that the already-engineered solution was too far from BBGS specifications to take for granted. It would have taken a significant effort in both field modeling and 3D design to complete the motor-stator type design, so the design was abandoned and left as a possible alternative for future iterations of the ISIS system.

Another steering magnet design which was investigated but quickly abandoned was that of STR4. The idea was to create a straightened version of the BBGS dipole magnet, and then to physically rotate that magnet around the beam axis in order to achieve steering in both x and y. Although the magnet design would have been quite simple, the STR4 design was rejected due to the complexity involved with rotating a heavy magnet that is under vacuum. Nevertheless, it may be useful to revisit such a design option in the future.

STR5 Field Design After nearly 10 rejected design iterations for a combined-function steering magnet, it was clear that a new strategy was necessary for packing the BBGS steering function into the 22 cm region available. Since the primary difficulty with achieving both x and y steering in a combined magnet was that the gap sizes grew too quickly, it was determined that separating the magnets could help to reduce required gap sizes and make the design feasible.

The design of two independent magnets proceeded by first determining which magnet should come first in the beam line. To do this, the geometry outlined in figure 4-19 was used. The goal was to estimate whether power could be saved by placing one magnet first instead of the other. It was assumed that power scales approximately linearly with the field required and the pole gap, G, of the magnet. Of the 22 cm available, the x magnet was assigned 10 cm and the y magnet 8 cm. This allowed 4 cm of extra room to account for parts of the coils which would not be included in the effective magnet length.

Using these numbers, the parameters in table 4.2 were calculated. The last number in the table represents the relative power increase required to place the steering

Table 4.2: Sizing of the independent steering magnets. G_1/G_2 denotes the pole gap if the magnet is placed first/second. P denotes estimated power. w is the tolerance added, which is the beam width (in x or y) plus 0.5 cm.

	θ	L [cm]	w [cm]	δ [cm]	G_1 [cm]	G_2 [cm]	$1/R$ [cm^{-1}]	P_2/P_1
X	15°	10	1.5	1.45	1.0	2.45	0.026	2.22
Y	5°	8	1.0	0.5	1.5	5.90	0.011	4.88

magnet second in the beam line. For the x magnet, the power was approximately doubled. For the y magnet, close to 5 times the power was estimated to be required. The $1/R$ term compares the relative power required for each magnet. From this calculation, the x magnet power was approximately double that of the y magnet. To understand what this means for magnet placement, take the x magnet power when it comes first to be P_x. This means that placing the x magnet second will cost about 1.2 P_x, but will save approximately $(0.011/0.026)3.88P_x \approx 1.6P_x$. This would suggest that placing the y magnet first would save some power. However, since this estimate was so rough, it was determined that the total power requirements for the steering magnets would be approximately the same no matter which came first.

Since power requirements alone could not determine which magnet to place first, other factors were considered. First, the mechanical rotation of the collimator and target were considered. Since in the x direction there needs to be 3 times the rotation, it would be beneficial to have the x pivot point as close to the target as possible to minimize lateral travel of the collimator tip. Second, the effects of magnetic field non-uniformity in the direction of the pole gap were considered. Since the second steering magnet will be steering a beam which is off-center from the magnet axis in the pole gap direction, it would be beneficial to minimize any non-uniformity. This leads to a conclusion that a smaller gap is preferred, and hence the x steering magnet should be placed second. From these considerations coupled with the power requirement calculations, it was determined that the beam would encounter the y steering magnet first, and then the x steering second.

The final designs for the x and y steering magnets are shown in figure 4-20. The coils for STR5 were originally designed with room for only 3 conductors horizontally.

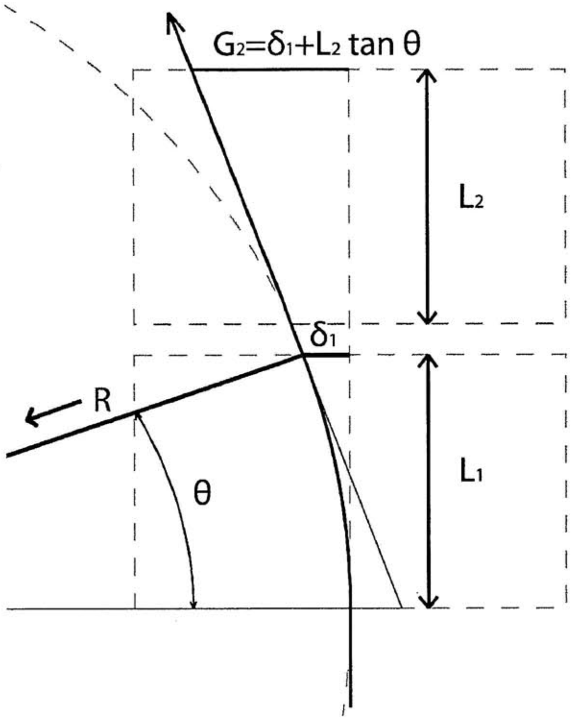


Figure 4-19: Geometry used to determine the minimum gap height (G_2) of the second steering magnet.

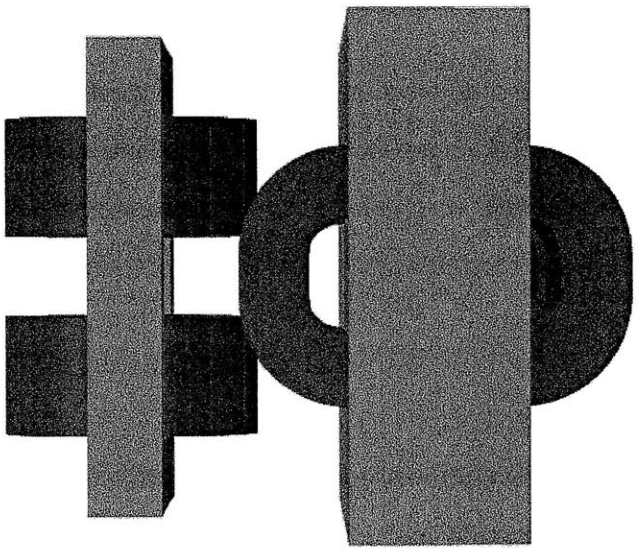


Figure 4-20: STR5x and STR5y final designs. y magnet is on the left, x on the right: beam travels left to right.

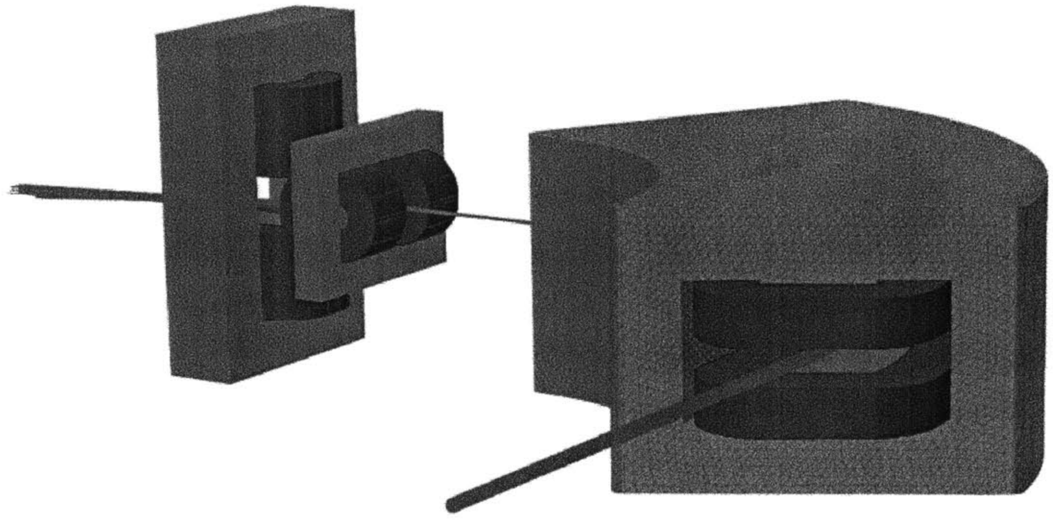


Figure 4-21: *Opera* model of the entire beam-line.

However, this would have meant that the inlet and outlet current leads could not exit the magnet from the same location. Hence, rather than deal with field errors, the coils where increased to be 4 conductors wide. The steering magnets use the same conductors as the dipole magnet. This allows for water cooling and also made the purchasing of conductors for the BBGS more straight-forward.

Once the 2D magnet design was complete, it was important to verify the fields and electron transport in 3D. While hand-calculations could estimate the effects of beam width and energy variance on the required aperture sizes, only a full 3D calculation could show them for the real beam from AES. For this purpose, an *Opera 3D* model was created which contained all three magnets (BND5, STR5y, STR5x) and the AES Parmela beam was transported through the field solution for all three beam energies. The results of this analysis are depicted in figures 4-21 and 4-22.

The 3D simulations showed that it was possible for the steering magnets to achieve effective steering angles of $\pm15/5°$ horizontal/vertical. Since the magnets operate outside of their central field regions where field gradients are large, the simulated phase space at the converter plates may not be as certain as that from the only the dipole transport. However, since there was convergence of the beam envelope for

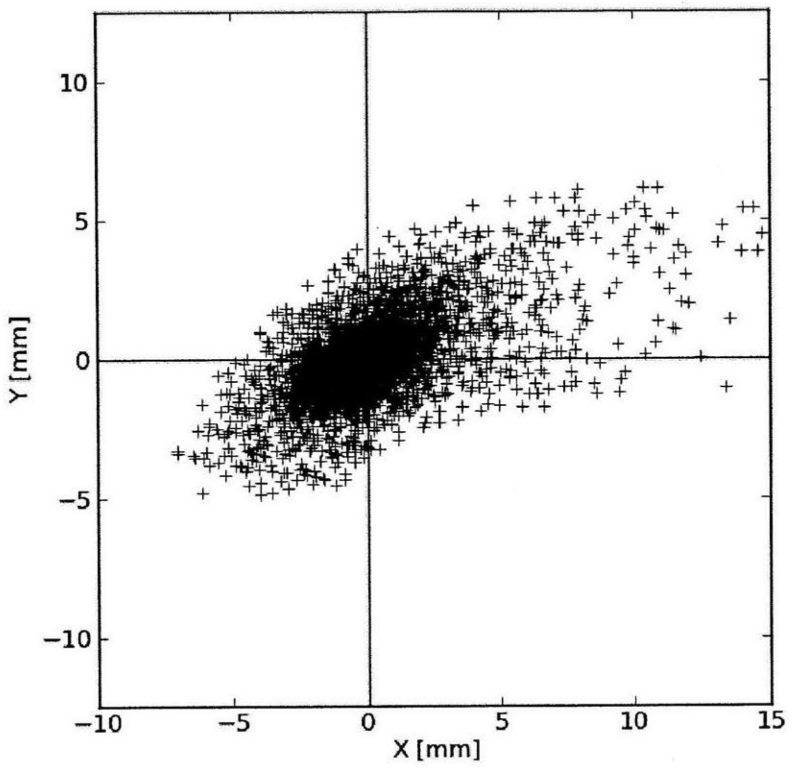


Figure 4-22: Beam cross section for simulation 607 evaluated at the converter plate for maximum bending angle in x and y. Percent of beam within a 2.5 mm radius is 84.0%.

smaller integration steps and mesh size, it was reasonable to base the steering magnet design on only 3D transport simulations.

Chapter 5

Engineering Design

Since the BBGS is a part of a practical engineering system and not just a bench-top experiment, it was important to ensure that the idealized 3D magnet design from field calculations could actually be converted into a practical engineering system. This meant making decisions on power supplies, coil windings, and water cooling patterns which would allow the BBGS to be fully integrated into the ISIS design.

The calculations performed in this chapter may be the most important for the system engineers who must interface and operate the BBGS.

5.1 Power Calculations

Each magnet in the BBGS system requires an independent power supply for operation. To specify each power supply, an equivalent circuit was generated for each magnet as shown in figure 5-1. The magnets were treated as a resistor and inductor in series. For each magnet, the maximum current was known to be approximately 200 A. To

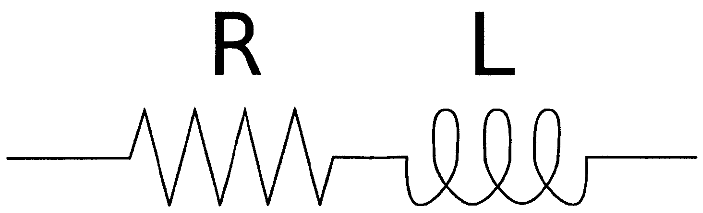


Figure 5-1: Simplified equivalent circuit for an electromagnet in the BBGS: a resistor and inductor in series.

Table 5.1: Conductor resistance calculation

Constant	Value	Units
Resistivity, ρ_{Cu}	1.77×10^{-6}	Ohm-cm
Inner Diameter	3.15	mm
Conductor Width	6.35	mm
Total Area	0.403	cm^2
Flow Area	0.078	cm^2
Copper Area	0.325	cm^2
Resistance/length, r_l	5.44×10^{-6}	Ohm/cm

calculate the resistance of the coils, the following equation was used:

$$R_{\text{coil}} = nL_{\text{ave}}r_l \tag{5.1}$$

where n is the number of turns in the coil, L_{ave} is the average length per turn, and r_l is the resistance per unit length, given by:

$$r_l = \rho_{\text{Cu}}/A \tag{5.2}$$

where ρ_{Cu} is the resistivity of Copper and A is the cross-sectional area of the conductor. For these calculations, a resistivity equal to 1.77×10^{-6} Ohm-cm was used. A calculation of the conductor resistance per unit length is shown in table 5.1

For each magnet, it was assumed that the power source would supply 200 A to the coils and that for one magnet both coils would be wired in series. This means that the total power for each magnet is given by:

$$P = I^2 R_{\text{total}} \tag{5.3}$$

and the voltage of the power supply should be:

$$V = P/I \tag{5.4}$$

Table 5.2: Final magnet specifications

Constant	Units	BND5	STR5x	STR5y
Conductors Wide	-	7	4	4
Conductors Tall	-	6	12	7
n turns/coil	-	42	48	28
Number coils	-	2	2	2
L_{ave} coil	cm	119.4	38.6	21.5
Total Length	m	100.3	37.1	12.0
Resistance	Ohm	5.46×10^{-2}	2.02×10^{-2}	6.55×10^{-3}
Current	Amp	200	200	200
Voltage	Volt	10.9	4.0	1.31
Power	kW	2.18	0.807	0.262
Stored Energy	Joule	271.8	64.5	6.70
Total Current	Amp-turns	16800	19200	11200
Inductance	Henry	1.93×10^{-6}	3.5×10^{-7}	1.07×10^{-7}

Finally, the inductance (L) of each magnet was calculated using the formula:

$$L = 2\frac{E_{\text{stored}}}{I_t^2} \tag{5.5}$$

where E_{stored} is the total stored energy in the magnet and I_t is the total current of the magnet in Amp-turns (note: total, not per coil). The energy stored in each magnet was found by performing a the volume integral:

$$E_{\text{stored}} = \int_V \frac{1}{2}\mathbf{H} \cdot \mathbf{B}\, dV \tag{5.6}$$

This integral was performed in *Opera 3D* using the volume integral tool.

A summary of these calculations for the three final BBGS magnets is shown in table 5.2.

5.2 Magnet Cooling

The ISIS BBGS system has several components which generate enough heat to require cooling beyond the built-in environmental cooling provided within the ISO container. In order to ensure that the BBGS cooling requirements are met, an independent cooling system which provides chilled water to all BBGS components was designed. This cooling system may either be run independently, or incorporated into a larger cooling system for the entire ISIS system.

5.2.1 Basic Design

Designing the cooling system was an iterative process which began with the selection of a reasonable operating pressure drop and temperature rise across individual components. To begin, a pressure drop of 20 psi and maximum temperature rise of 20°C were selected as reasonable operating conditions.

Next, scoping calculations were performed for each component to determine how many cooling loops per coil would achieve the stated pressure and temperature goals. For these calculations, the properties of 20°C water were used and assumed to be constant with changes in temperature. For water flow through smooth pipes of diameter D, the relationship used for the pressure drop, ΔP, was:

$$\Delta P = \frac{8\rho f L Q^2}{\pi^2 D^5} \tag{5.7}$$

where L is the length of the pipe, Q is the volumetric flow rate, ρ is the density and f is the friction factor, given by:

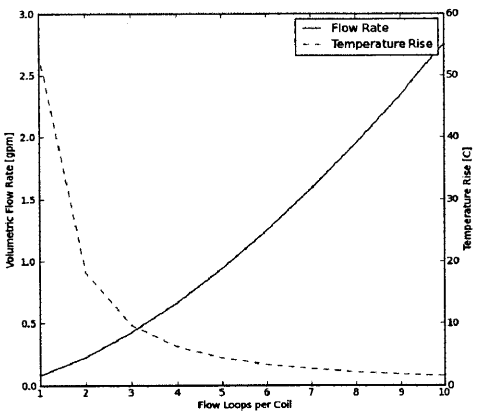
$$f = 0.184 Re^{-0.2} \tag{5.8}$$

where Re is the Reynold's number,

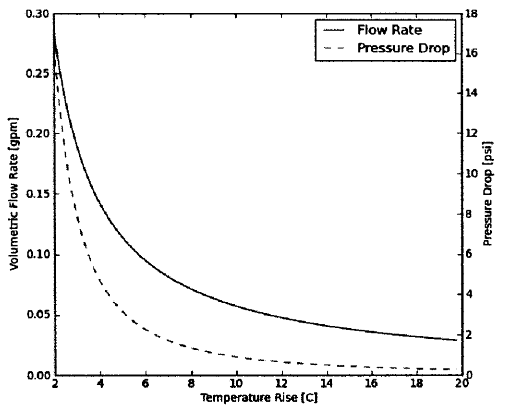
$$Re = \frac{4\rho Q}{\pi \mu D} \tag{5.9}$$

and the flow is assumed to be turbulent. For a derivation of these equations, see Todreas and Kazimi (1990).

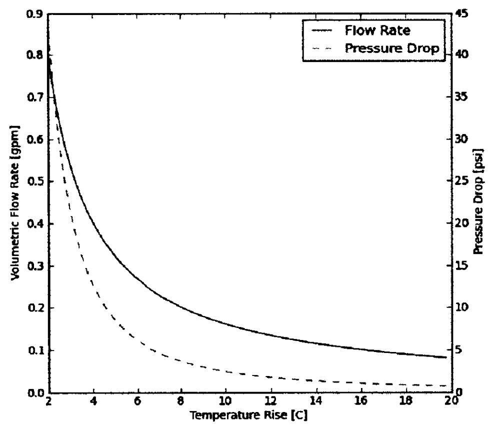
The results of a preliminary model of the flow are shown in figure 5-2. The conclusions drawn from these figures are that the dipole should operate at about 3 cooling loops per coil, the X-steering magnet at one loop per coil, and the Y-steering



(a) Dipole, $\Delta P = 20$ psi



(b) Y Steer, 1 loop/coil



(c) X Steer, 2 loop/coil

Figure 5-2: Cooling parameters based on individual components

magnet on only one loop for the entire magnet. Reducing the number of loops in each of the smaller magnets gives a higher outlet temperature of the water, but it also reduces the flow rate required for the same pressure drop, so it makes sense to do so as much as possible.

As a simple approximation of the BBGS cooling system, each component was placed into a simple parallel circuit as shown in figure 5-3. This allowed for a computation of flow rate and outlet temperatures without non-linear iterations. The results of this calculation are shown in table 5.3.

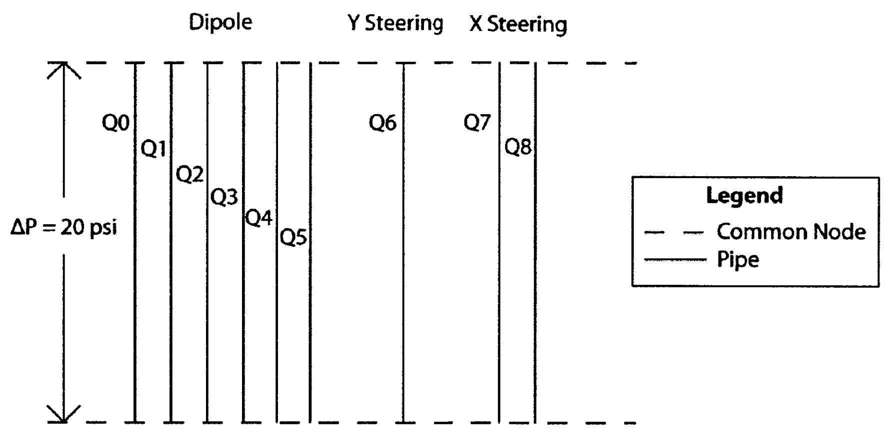


Figure 5-3: Simplified cooling system layout.

Table 5.3: Cooling parameters for components in parallel, 20 psi pressure drop.

Component	n loops	ΔP [psi]	Q [gpm/loop]	ΔT [°C]
Dipole	6	20.0	0.141	9.65
Y Steer	1	20.0	0.17	6.39
X Steer	2	20.0	0.137	11.8
Total		20.0	1.29	9.67

5.2.2 BBGS Non-linear System Model

The issue with the model described above is that it does not accurately represent the flow path of a practical system, and hence may give errors in actual flow rates and pressure drops. Such a system requires serial distribution lines which may effect the flow rates provided to each individual component. A more realistic model for the BBGS cooling distribution is shown in figure 5-4. In order to solve this and potentially more complex systems in the future, a non-linear solving code was developed.

The solver uses a system of linearized equations for the volumetric flow rate. A matrix equation in the form:

$$AQ = b \tag{5.10}$$

is solved, where Q is a vector of flow rates along each individual pipe in the system. The matrix A and vector b are built from a combination of flow conservation equations at each node as well as pressure conservation equations around each loop within the

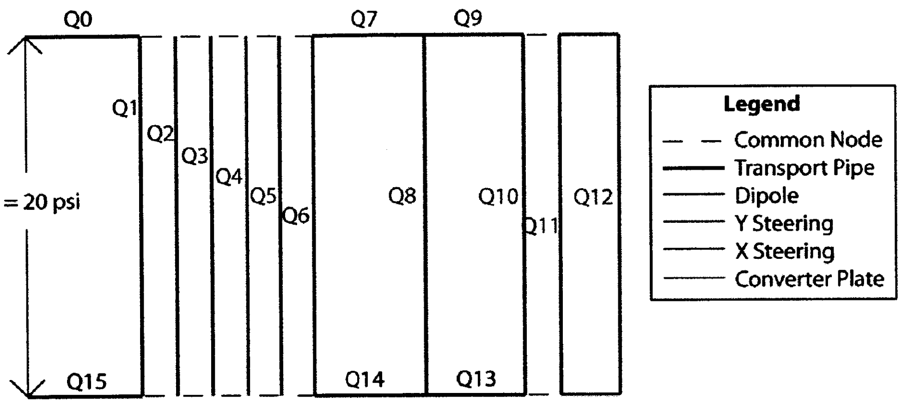


Figure 5-4: Modified cooling system layout.

system. In these equations, the pressure drop across each pipe is related to the flow rate by:

$$\Delta P = KQ \tag{5.11}$$

where K is given by the sum of two components: K_1, the resistance from lengths of pipe, and K_2, the combined resistance from turns, diameter reduction, and other components in the loop. To find K_1, equations 5.7 through 5.9 were combined, which gives:

$$K_1 = \frac{0.184 * 8}{4^{0.2}} \frac{\rho^{0.8} \mu^{0.2} L Q^{0.8}}{\pi^{1.8} D^{4.8}} \tag{5.12}$$

and for a pipe with n_t resistive components,

$$K_2 = \sum_{i=1}^{n_t} k_i \frac{\rho}{2} \frac{Q}{\pi \left(\frac{D}{2}\right)^2} \tag{5.13}$$

where k_i is the form loss coefficent for the component, a factor determined by Todreas and Kazimi (1990) to be of order 0.5 for 90° bends in pipe.

To solve for the flow rate, the matrix equation is solved and iterated upon using an updating strategy which incorporates both the old and new values for Q:1

$$Q_{\text{new}} = Q_{\text{new}}^{0.45} \times Q_{\text{old}}^{0.55} \tag{5.14}$$

Following this procedure, the matrices corresponding to the system described in figure 5-4 are shown below:

```
                                    [[ A ]]                                      [ Q ]  =    [b ]
[ 1. -1. -1. -1. -1. -1. -1. -1.  0.  0.  0.  0.  0.  0.  0.  0.] [Q0 ]        [0 ]
[ 0.  0.  0.  0.  0.  0.  0.  1. -1. -1.  0.  0.  0.  0.  0.  0.] [Q1 ]        [0 ]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1. -1. -1. -1.  0.  0.  0.] [Q2 ]        [0 ]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  1.  1. -1.  0.  0.] [Q3 ]        [0 ]
[ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.  1. -1.  0.] [Q4 ]        [0 ]
[ 0.  1.  1.  1.  1.  1.  1.  0.  0.  0.  0.  0.  0.  0.  1. -1.] [Q5 ]        [0 ]
[ 9.  9.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  9.] [Q6 ]        [dP]
[ 9.  0.  9.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  9.] [Q7 ]   =    [dP]
[ 9.  0.  0.  9.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  9.] [Q8 ]        [dP]
[ 9.  0.  0.  0.  9.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  9.] [Q9 ]        [dP]
[ 9.  0.  0.  0.  0.  9.  0.  0.  0.  0.  0.  0.  0.  0.  0.  9.] [Q10]        [dP]
[ 9.  0.  0.  0.  0.  0.  9.  0.  0.  0.  0.  0.  0.  0.  0.  9.] [Q11]        [dP]
[ 9.  0.  0.  0.  0.  0.  0.  9.  9.  0.  0.  0.  0.  0.  9.  9.] [Q12]        [dP]
[ 9.  0.  0.  0.  0.  0.  0.  9.  0.  9.  9.  0.  0.  9.  9.  9.] [Q13]        [dP]
[ 9.  0.  0.  0.  0.  0.  0.  9.  0.  9.  0.  9.  0.  9.  9.  9.] [Q14]        [dP]
[ 9.  0.  0.  0.  0.  0.  0.  9.  0.  9.  0.  0.  9.  9.  9.  9.] [Q15]        [dP]
```

where in the matrix A above, each 9 represents a placeholder for the calculated value of K for the pipe corresponding to the column number. These values are updated during each time step using the flow rates obtained in the previous time step.

A first calculation was performed without the converter plates in order to see the difference between the simplified and more complex model. The results of this calculation are shown in table 5.4 and figure 5-5.

Table 5.4: Cooling parameters for components within the complex system layout, 20 psi system pressure drop.

Component	n loops	ΔP [psi]	Q [gpm/loop]	ΔT [°C]
Dipole	6	18.56	0.126	10.66
Y Steer	1	18.52	0.121	9.00
X Steer	2	18.51	0.109	14.89
Total		20.0	1.10	11.27

Combined Magnet and Converter Plate Cooling Since the converter plate assembly and collimator were estimated to require close to 5 kW of cooling, it was decided that the magnets and the converter should be cooled in parallel systems rather

[1]This acceleration strategy was adopted from Gupta and Prasad (2000). Without acceleration, solutions are highly dependent on the initial guess for Q and often oscillate without convergence.

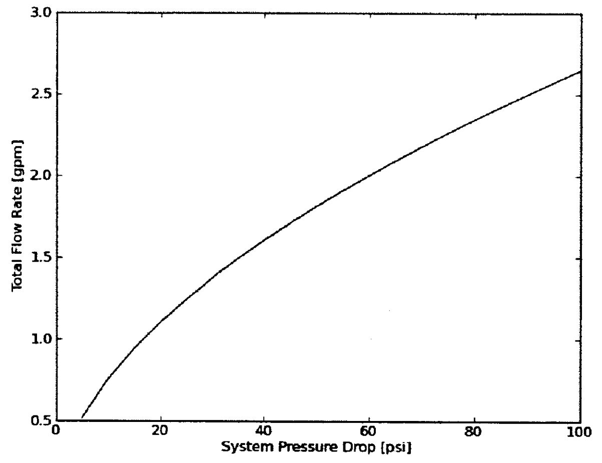
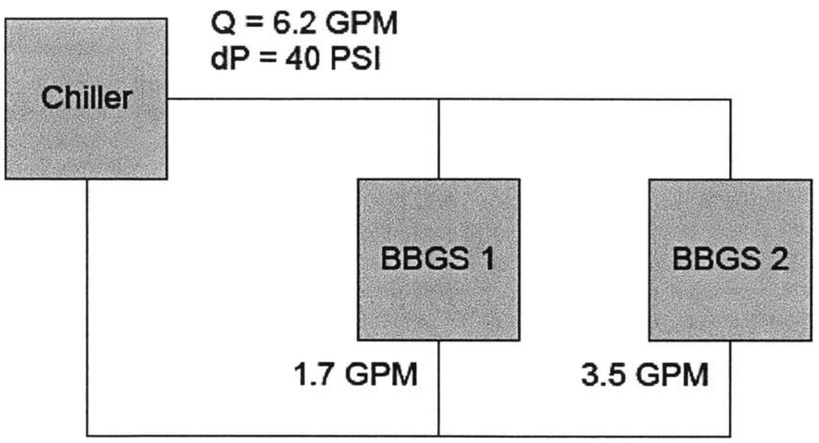


Figure 5-5: Flow rate vs applied pressure drop for total system, excluding converter plates.

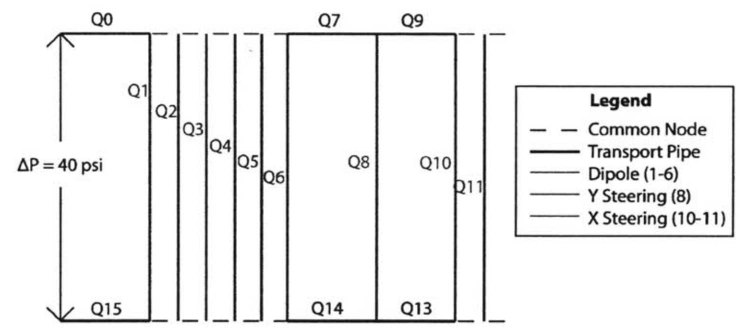
than attaching the converter cooling to the end of the magnet cooling loop. This new system layout is depicted in figure 5-6, where BBGS 1 represents the subsystem of all magnets and BBGS 2 represents the converter assembly.

The design goal for this new system layout was to achieve a rise in water temperature of no more than 10°C over any component while also balancing the flow such that no component has a significantly higher flow rate than required. For the BBGS 2 subsystem, it was assumed that the components would be cooled using lengths of the same hollow copper conductor used by the BBGS magnets. The subsystems were assumed to be supplied by a large diameter (2 in) pipe such that the pressure drop along the distribution lines was negligible.

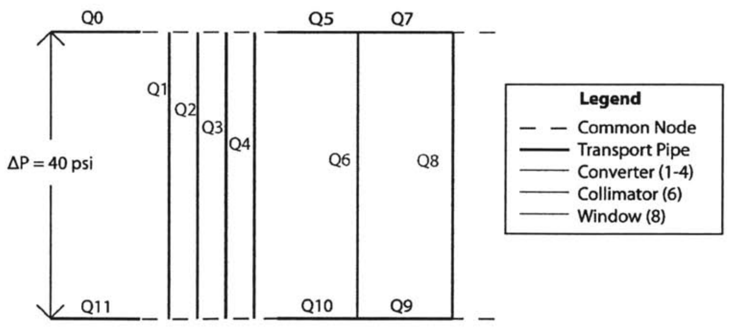
The results of this analysis are shown in table 5.5. The total system requires a pressure drop of approximately 40 psi and flow rate of 6.2 gpm from the chiller. In addition to providing this number as a requirement, the system curves for both the BBGS 1 and BBGS 2 subsystems are provided in figure 5-7. These curves should be used by Raytheon systems engineers to incorporate the BBGS into the ISIS cooling system.



(a) Total System

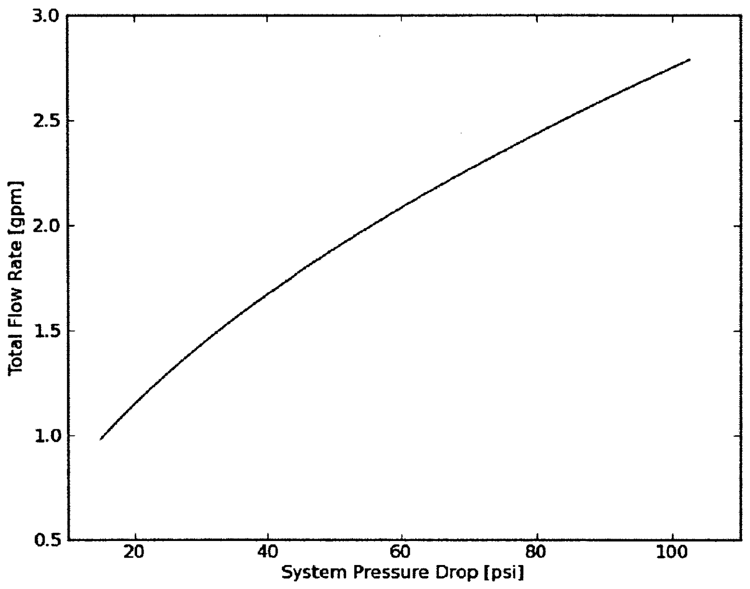


(b) BBGS 1

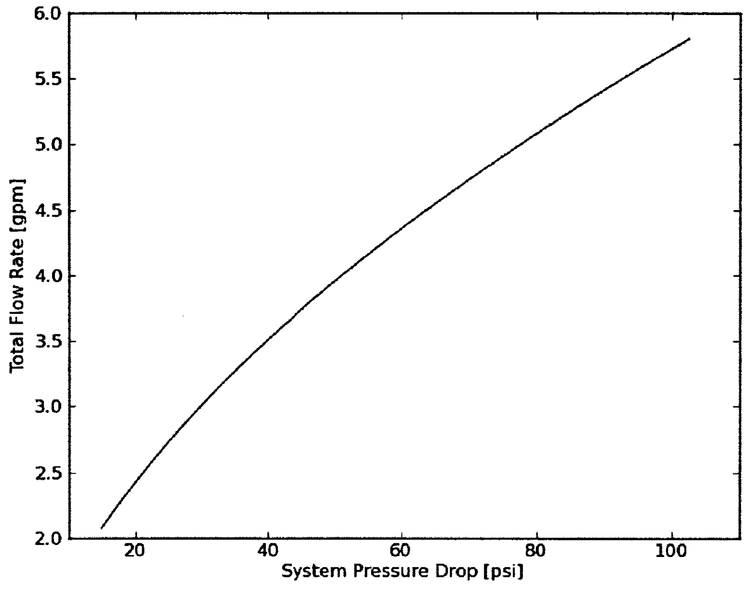


(c) BBGS 2

Figure 5-6: Simplified diagram of cooling system and subsystems. BBGS 1 includes all magnets, BBGS 2 includes converter plate, beam window, and collimator.



(a) BBGS 1



(b) BBGS 2

Figure 5-7: System curves for the BBGS 1 and BBGS 2 cooling subsystems.

Table 5.5: Cooling parameters for components within the final system layout, 40 psi system pressure drop.

Component	n loops	Flow [gpm/loop]	Q_{total} [gpm]	Power [kW]	ΔT [°C]
Dipole	6	0.192	1.15	2.150	7.00
Y Steer	1	0.182	0.182	0.290	5.99
X Steer	2	0.160	0.328	0.850	9.85
Beam Window	1	0.5	0.5	0.1	0.72
Converter	4	0.614	2.46	4.50	6.94
Collimator	1	0.5	0.5	0.3	2.17

5.3 Converter Plate Water Cooling

Before settling upon a final design for the water cooling system, it was necessary to study the water cooling of the converter plate assembly in more detail. The reason for this is that unlike in the magnets, the heat is generated with high density far from the water cooled region. Hence, even with water cooling established, the temperature within the converter plates could reach unsuitable levels.

A basic thermal analysis of the BBGS photon converter plate assembly was conducted using the *Opera 3D* Poisson equation solver. The mathematical equivalence of electrostatic problems to steady-state heat conduction was used to complete a 3D heat analysis without employing new software.

5.3.1 Steady-State Heat Diffusion in Opera 3D

To save the time and hassle of building or learning a new code for this important but simple task, the thermal analysis of the converter plates was completed using the *Opera 3D Electrostatics* solver. The form of the differential equation for heat conduction is the same as that for electrostatic fields. Apart from constants, the two problems have the same mathematical solution, as will be demonstrated in the following section.

Theory The *Opera 3D Electrostatics* solver is a finite-element code designed to solve Maxwell's equations for the electric field in the limit where $\frac{\partial}{\partial t} = 0$. This code solves

Poisson's equation for the electric potential, which can be derived from Maxwell's equations. Starting from Gauss' law:

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon} \tag{5.15}$$

Now we write E as the gradient of some potential:

$$\mathbf{E} = -\nabla \phi \tag{5.16}$$

We can see immediately that this formulation satisfies Faraday's Law of Induction with the steady state assumption:

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} = 0 \tag{5.17}$$

since, by definition:

$$\nabla \times (-\nabla \phi) = 0 \tag{5.18}$$

Plugging in to equation 5.15, we find:

$$-\nabla^2 \phi = \frac{\rho}{\epsilon} \tag{5.19}$$

which is Poisson's equation for the electric potential.

The derivation of Poisson's equation for heat conduction follows from the conservation of thermal energy. Starting with an integral balance of thermal energy, heat flux, and volumetric sources, we write:

$$\frac{\partial}{\partial t} \int_{\Omega} \epsilon T \, dV = \int_{\partial \Omega} \mathbf{q} \cdot \hat{\mathbf{n}} \, dS + \int_{\Omega} s \, dV \tag{5.20}$$

where ϵ is the volumetric specific heat, T is temperature, $\mathbf{q}$ is the heat flux, $\hat{\mathbf{n}}$ is the normal vector to the surface, and s is the volumetric heat generation rate. Assuming steady state ($\frac{\partial}{\partial t} = 0$) and applying Green's theorem to the surface integral of the heat flux:

$$\int_\Omega \nabla \cdot \mathbf{q}\, dV = -\int_\Omega s\, dV \tag{5.21}$$

$$\nabla \cdot \mathbf{q} = -s \tag{5.22}$$

Finally, given a linear relationship between the heat flux and the gradient in temperature, where we assume the thermal conductivity is constant with temperature, we find:

$$\nabla \cdot k\nabla T = -s \tag{5.23}$$

$$-\nabla^2 T = \frac{s}{k} \tag{5.24}$$

Which is Poisson's equation for steady-state heat conduction.

The final step in relating electrostatics to steady state heat conduction is choosing suitable units for comparison. In this analysis, the electric potential in Volts is treated as the temperature in Celcius. The electric charge density in coulombs/cm^3 is treated as the heat generation rate in Watts/cm^3. Finally, since *Opera* requires that ϵ be specified in units of relative permittivity, the thermal conductivity k is specified in Watts/(cm C) and then multiplied by the factor $\frac{1}{8.854E-12}$. The *Opera* solution is solved using the MKS system.

Basic Verification Before generating a detailed analysis of the BBGS converter plates, it was important to benchmark the Opera solver using geometry for which Poisson's equation for heat conduction could be solved analytically.

For this task, an infinite slab geometry was assumed. The slab was uniform and infinite in the y and z directions, and extends to $x = \pm 5$ m. Within the region $x = \pm 0.5$ m, a uniform heat generation rate of 10W/m^3 was applied. Throughout the slab, a thermal conductivity $k = 1.0$ W/(m K) was assumed. The boundary conditions of the problem specify the temperature at $x = \pm 5$ m to be 100°C.

Using symmetry, the analytical solution for positive x gives:

$$T(x) = -5x^2 + 123.75 \quad , \quad 0 \leq x \leq 0.5\text{m}$$
$$T(x) = -5x + 125 \quad , \quad 0.5 \leq x \leq 5\text{m}$$

This results in a maximum temperature of 123.75°C at $x = 0$ and a constant heat flux equal to 5 W/m^2 for $x \geq 0.5$ m, numbers which can be compared with the *Opera* solution.

Implemented in *Opera*, the solution gives $T_{\text{max}} = 123.747$°C and $q = 5 \pm 10^{-6}$ W/m^2 for all $x \geq 0.5$ m. This agrees well with the hand-calculated analytical model.

5.3.2 Data Sources and Limitations

In order to generate a useful model of the converter plate heat transfer, it was important to first determine the material properties and volumetric heat generation in the material.

The determination of the thermal conductivity of graphite was not straightforward, as graphite comes in many different forms and the heat transfer is highly dependent on not only temperature but also the direction of heat flow. As a result, several models were generated which present best and worst cases for the thermal conductivity. The upper and lower bounds on the thermal conductivity were determined using two papers, Lutcov et al. (1970) and Albers (2009).

To model the heat generation rate within the converter plates it was necessary to make several assumptions about the electron transport through the plates. First, it was important to approximate the proportion of the beam energy which is not deposited in the graphite plates, as well as the maximum power density for each beam. This was estimated using the NIST E-STAR database (NIST, 2010) for the electron range and the radiation yield. Table 5.6 shows the numbers used to estimate which beam would result in the highest power density in the converter plates. The assumption of this calculation was that the energy would be deposited approximately uniformly through the plate. The linear power density will not actually be uniform.

Table 5.6: Estimation of linear power density for each beam energy

Energy [MeV]	60.7	30.6	6.2
Current [μA]	80.0	72.7	55.2
Power [W]	4900	2200	350
CSDA Range [g/cm^2]	26	15	3.5
Average Range [cm]	15.3	8.8	2.06
Radiative Yield	20%	12%	2.2%
Power Density [W/cm]	250	220	162

However, since the CSDA range is a concave down function of energy, the power density for the BBGS will always be highest in the higher energy beam.

Since the distribution of power deposition inside the converter assembly was not known with certainty, several models were used to estimate best and worst-case scenarios for beam power distribution. They are described in section which follows.

5.3.3 Models

A view of the basic model geometry used for calculations is shown in figure 5-8. In this model, the green (1) outer region represents the 1020 steel frame, the purple (2) region represents air, and the blue (3) region represents the high density graphite converter plates. In all versions of the converter plate model, the $\pm\hat{\mathbf{x}}$ boundaries are

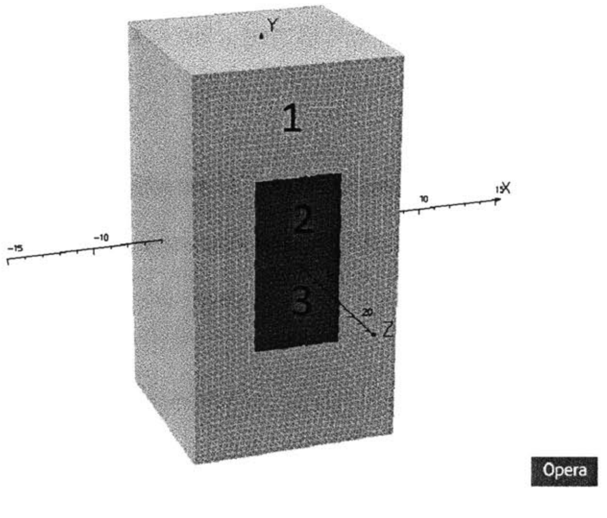


Figure 5-8: Isometric view of simplified converter assembly.

assumed to be held at a constant temperature of 40°C. This number was obtained by assuming that all of the heat generated is transferred through the side walls since they will have water cooling plates attached. This also assumes that the walls are cooled using 3.15 mm diameter tubes with a heat transfer coefficient determined by the Dittus-Boelter correlation:

$$h = \frac{k}{D_H} 0.023 \mathrm{Re}^{0.8} \mathrm{Pr}^{0.4} \tag{5.25}$$

where k is the thermal conductivity of water, D_H is the diameter of the tubing, Re is the Reynold's number, and Pr is the Prandtl number, given by:

$$\mathrm{Pr} = \frac{\mu c_p}{k} \tag{5.26}$$

where μ is the viscosity and c_p is the specific heat of water. For the other boundary conditions in the model, a zero heat flux boundary was enforced. This is a worst-case assumption which accounts for the fact that the magnitude of heat transfer through the air-cooled boundaries will be much smaller than that through the water cooled channels.

Since the material properties and energy deposition rates were uncertain, several models were created for testing. In every model, the power in each converter plate was distributed in a cylindrical region along the beam line with a radius of 0.5 cm. The linear power distribution in the beam direction was varied between models to test different scenarios. Model descriptions and the results for maximum temperature in the graphite are shown in table 5.7.

Table 5.7: Model descriptions and results

Model	Power Distribution	k W/(cm K)	$T_{\max}$ [°C]
A	Uniform	50.0	159
B	Uniform	5.0	268
C	Uniform	1.0	323
D	Distributed[2]	1.0	373
E	Uniform 150% power density	1.0	384

[2]Heat distributed in the beam direction using the results of Tabata et al. (1994) for electron

The interpretation of the above results was dependent on what material temperature limits would be important. The reaction of primary concern was that of the oxidation of graphite in air, which through the results of Shemlet et al. (1994) becomes important for temperatures greater than 500°C.

In the worst case modeled, a margin of 115°C was obtained. This margin may not be sufficient to prevent rapid oxidation of the converter plates since the models did not properly account for the non-linear behavior of thermal conductivity, and the energy deposition rate within the sample plates is not well known. In addition, the assumption of a constant temperature boundary of 40°C is highly dependent on the simple heat transfer model assumed between the coolant and the converter assembly wall.

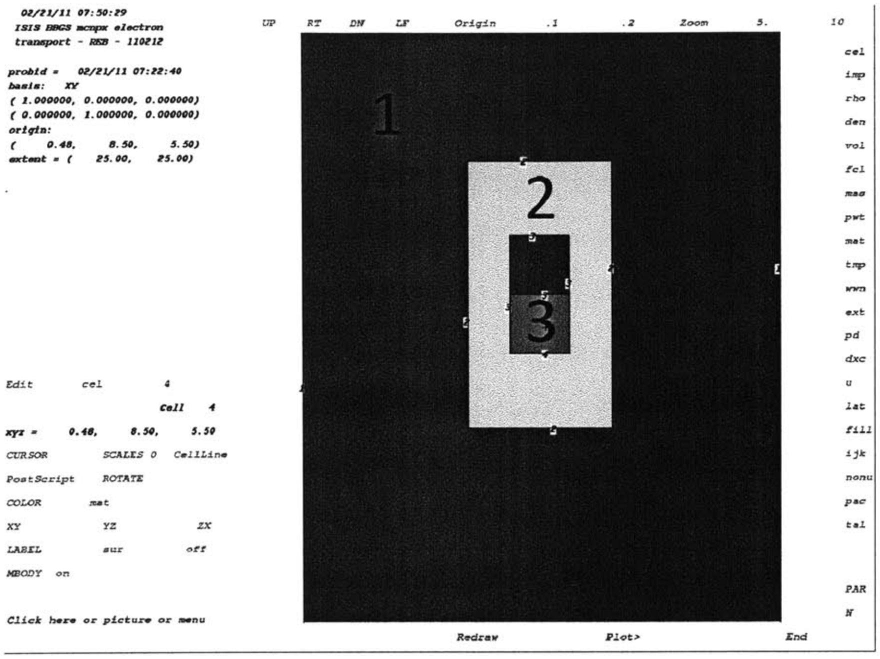
In order to build a better model for the heat transfer in the BBGS converter plates, it was important to determine more accurate estimates the energy deposition rate in the material. This required the use of MCNPX for energy deposition calculations in order to verify that the estimates for energy deposition rates were conservative enough to model the worst case for heat transfer.

5.3.4 MCNPX Calculations

Although the MCPNX modeling of the converter plates and photon beam was tasked to Raytheon, it was useful to create a simplified model which could be used as an extra verification tool. A cross-sectional view of the MCNPX model for the BBGS converter plates is shown in figure 5-9. The model is composed of four 'cells,' which are used to specify unique regions in MCNPX. It has a geometry identical to that used in the *Opera 3d* analysis, except there is a bounding box of air which extends beyond the model by at least 20 cm in all directions.

In order to define an electron beam, the model used the MCNPX 'SDEF' card. The SDEF card allows for the creation of a beam with one energy but a truncated Gaussian distribution in space. The truncation is performed by defining a cell outside which no particles may be generated. In the BBGS model, the source is limited to a

energy deposition in materials.

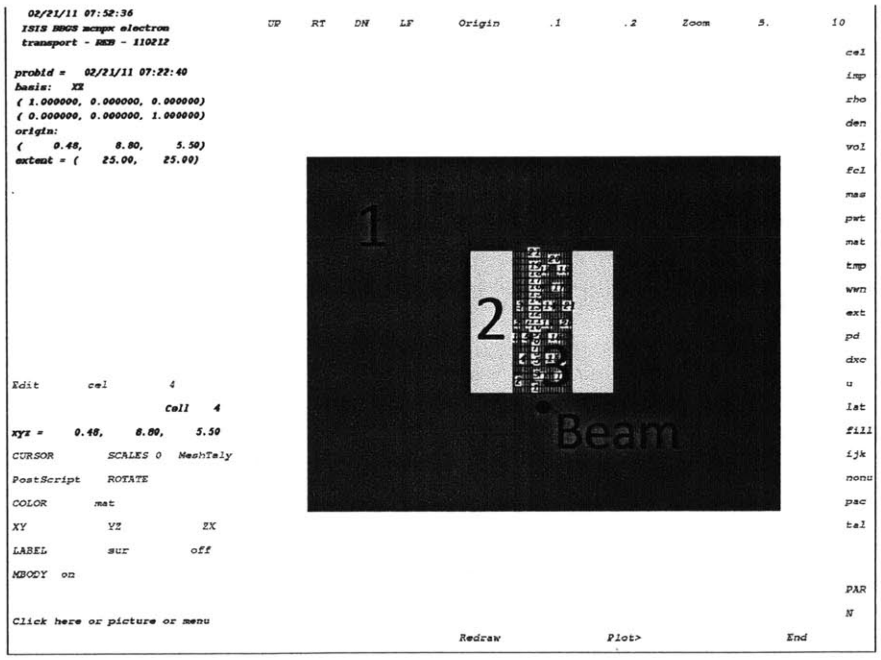


```
02/21/11 07:50:29
ISIS BBGS mcnpx electron
transport - RSB - 110212

probid =   02/21/11 07:22:40
basis:   XY
( 1.000000, 0.000000, 0.000000)
( 0.000000, 1.000000, 0.000000)
origin:
(     0.48,      8.50,      5.50)
extent = (   25.00,    25.00)

Edit        cel        4
                   Cell   4
xyz =    0.48,     8.50,     5.50
```

(a) xy plane



```
02/21/11 07:52:36
ISIS BBGS mcnpx electron
transport - RSB - 110212

probid =   02/21/11 07:22:40
basis:   XZ
( 1.000000, 0.000000, 0.000000)
( 0.000000, 0.000000, 1.000000)
origin:
(     0.48,      8.80,      5.50)
extent = (   25.00,    25.00)

Edit        cel        4
                   Cell   4
xyz =    0.48,     8.80,     5.50
```

(b) xz plane with mesh tally displayed

Figure 5-9: Cross sections of the MCNPX model, created using the built-in MCNPX plot function. Regions labeled are: (1, blue) Air, (2, green) Stainless Steel, (3, red) Graphite

0.5 cm wide by 0.2 cm tall box which is centered on the ideal beam centerline. This represents an approximate *rms* beam envelope from the electron beam simulations. All electrons generated travel in the forward beam direction since MCNPX does not have built-in capability for creating beam divergence.

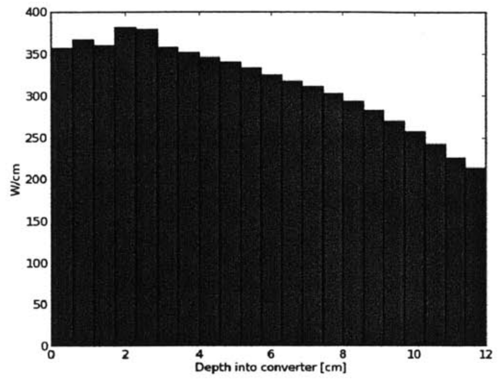
The primary function of this MCNPX model was to determine whether or not the estimates for heat deposition rate in section 5.3.3 were overestimates or underestimates. If the heat deposition rate from MCNPX modeling was higher than the worst-case previously modeled, then it would be necessary to run more detailed heat simulations. Hence, the only tally performed in the MCNPX model was a mesh tally which counted energy deposition rates throughout the carbon converter plates. The mesh tally was performed over a $20 \times 20 \times 20$ mesh which covered the $5.0 \times 5.0 \times 12.0$ cm length of the carbon targets in the converter assembly. For simplicity, the 1mm air regions between each converter plate were ignored. Simulations with 100000 particles were run for an electron energy of 60.67 MeV and a graphite density of both 2.1 g/cm^3 and 1.7 g/cm^3. By running with different densities, it was possible to estimate a maximum and minimum case for heat deposition on the converter plates.

The results of the MCNPX simulations are shown in table 5.8 and figure 5-10. The MCNPX simulations gave insights into the true nature of the electron trans-

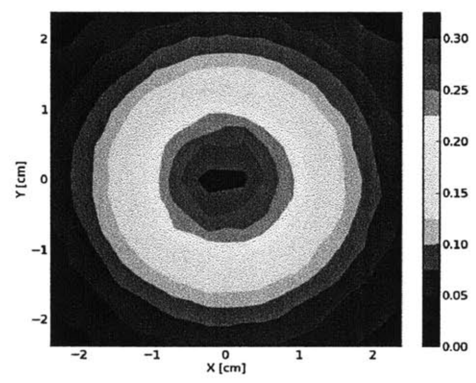
Table 5.8: MCNPX Energy Deposition Results

Model	Density	Total Power	Peak Linear Power	Est. Collimator Power
-	[g/cm^3]	[W]	[W/cm]	[W]
A	1.7	3800	370	500
B	2.1	4300	450	100

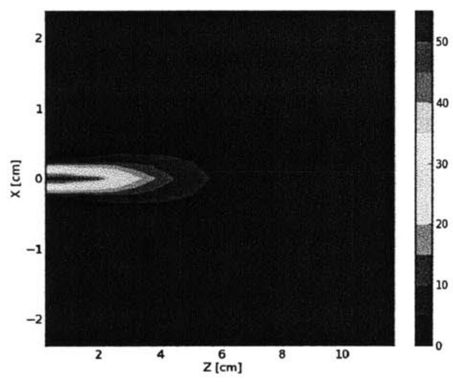
port in the converter plates. Rather than being contained within a 1 cm diameter tube, the energy is deposited in a cone shape which expands as the distance from the first converter plate is increased. The maximum linear energy deposition rate for 1.7 g/cm^3 graphite was 370 W/cm, and it occurs approximately 3 cm into the converter assembly. This maximum energy deposition was about the same as those energy deposition rates modeled in *Opera*, and hence it was determined that further simulations would not be necessary to validate the cooling design.



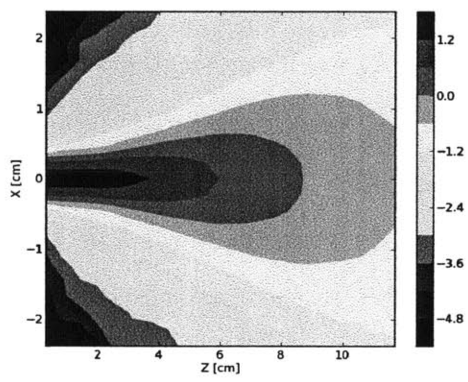
(a) Linear Power Density



(b) Energy Deposition (W/cm^3) evaluated in xy plane at $z = 12$ cm



(c) Energy Deposition evaluated in xz plane at $y = 0$ cm



(d) Same as (c), Logarithmic scale

Figure 5-10: Energy deposition rates evaluated over several regions for model A, graphite density equal to 1.7 g/cm^3.

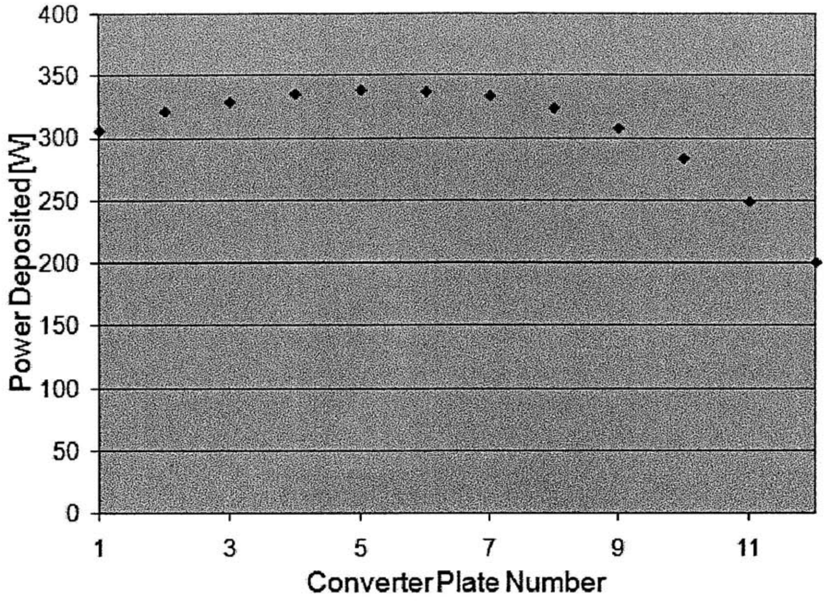


Figure 5-11: Energy deposition rates evaluated for electrons in Graphite from ISIS project technical lead, B. Blackburn. Graphite density used was 2.1 g/cm^3.

In order to verify the validity of the results from this analysis, the results were compared to those of Brandon Blackburn of Raytheon, shown in figure 5-11. The energy deposition curves did not agree exactly. Even though Blackburn used a converter plate density which was 20% higher, the peak power density was slightly lower. This was a bit concerning, but it was determined that since Blackburn had run several simulations with many more particles and verified each with multiple checks, his simulation was most likely a more accurate depiction of reality.

One reason which may help to account for the discrepancy between the two simulations was that the MCNPX simulations performed by the author may not have handled the photon energy deposition correctly. Since there were no supercomputers available, it was necessary to transport only the electrons and not the secondary photons. This means that the photon energy may have been assumed to be deposited where it was generated - which would help to explain why the models in this thesis were peaked closer to the first plate than those run by B. Blackburn.

Simulation Sensitivity Since the heat conduction simulations relied on several basic assumptions about the heat transfer between the water and the converter assembly, it was important to understand how sensitive the results were to these assumptions. The thermal margin of 115°C is not significant unless its sensitivity to assumptions is clearly defined.

To quantify the sensitivity of the thermal margin to assumptions about heat transfer, the boundary temperature of the BBGS model E was increased from 40°C until the maximum temperature of the solution reached 500°C. This corresponded to a boundary temperature of 150°C. This means that the calculated heat transfer coefficient between the water and the copper conductor could be as low as 30% of the calculated value before the thermal limits of the converter plates are reached. The Dittus-Boelter correlation, like any thermal hydraulic correlation, could not be assumed to have more than $\pm 20\%$ accuracy. However, even this level of uncertainty would not result in a temperature beyond the thermal margins.

Even with this analysis complete, it is important to note that the converter plate thermal analysis remains quite uncertain. Every assumption in the analysis was meant to be conservative, but there were many assumptions that could not be validated independently. For example, the thermal conductivity of graphite was assumed to be a worst-case based on data available in the literature and from the vendor, but it is not clear if the graphite which arrives will have the same properties. This, coupled with the uncertainties in the heat transfer correlations and water flow rate calculations, could lead to undesirable results. Hence, it will be important to test the BBGS converter assembly not only for its photon generation but also for its heat transfer properties. It will also be prudent to plan for extra water cooling for the converter plates in case it is needed.

Chapter 6

Proposed Design Verification

The timing of the BBGS design and fabrication did not allow for the BBGS to be tested as a part of this thesis project. However, a series of simulations and a testing plan was created in order to facilitate future testing.

6.1 MIT Electron-Cyclotron Resonance Ion Source

Since the ISIS LINAC will not be immediately available once the BBGS is assembled, the BBGS will be verified using an Electron-Cyclotron Resonance Ion Source (ECRIS) which is being commissioned at MIT in Summer 2011. The ECR ion source was designed by Armero (2010) as a part of a Master's thesis project and will be commissioned and tested by Mark Artz as a part of his Master's work. A schematic diagram of the ECRIS and its planned interface with the BBGS is shown in figure 6-1.

For testing the BBGS, the ion source will be used to generate H^+ ions. Now, even though the ISIS LINAC will use electrons, it will be possible to test the magnet at similar operating conditions to those in ISIS. The reason for this is that the particle momentum is given by:

$$p = \sqrt{T^2 + 2TE_o} \tag{6.1}$$

where T is the particle kinetic energy and E_o is the rest mass energy. The ion source

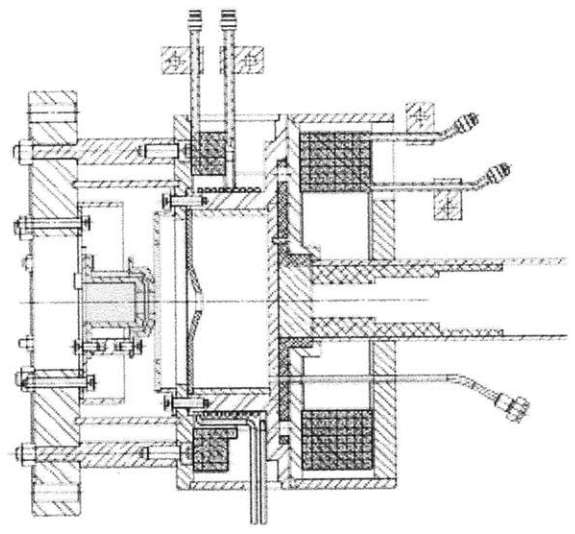
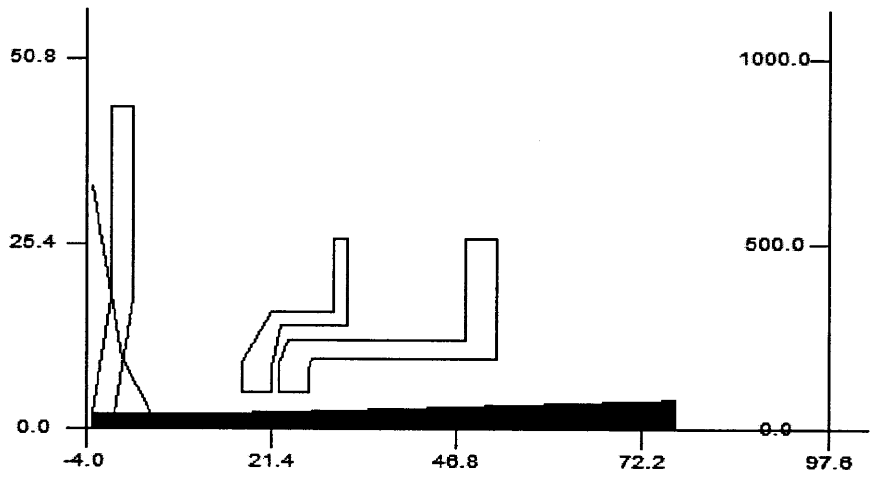


Figure 6-1: Schematic of the MIT ECRIS adapted from Armero (2010). The vacuum flange will interface directly with the BBGS 6-way valve using a zero-length coupling. Beam travels from the shaded region and exits left.

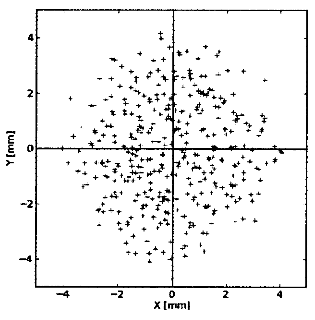
can accelerate protons to energies up to 30 keV, giving them a momentum of 7.5 Mev/c. Due to the mass difference between electrons and protons, an equivalent momentum electron beam would have a kinetic energy of 7.01 MeV. Hence, even though the ECR ion source emits protons, at high voltage it will produce particles with momentum equivalent to electrons in the ISIS operating range of 6 to 60 MeV. While the ion source will not allow for testing at full power, agreement between low energy simulations and measurements should be sufficient to verify the transport system operation.

6.2 ECRIS Beam Simulations

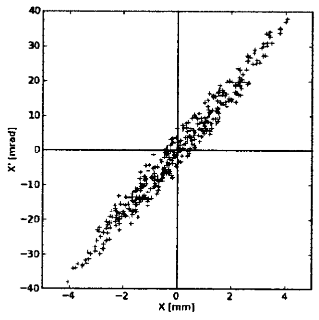
Beam simulations were performed in both 1D and 3D in order to prepare for the testing of the BBGS with the ECRIS. To generate an initial beam, Mark Artz performed a calculation of the ECRIS using the *BEAM3D* code (Antaya and Xie, 1987). The results of this simulation are shown in figure 6-2. The *BEAM3D* output files specify particle positions and velocities in the transverse and beam direction, (x, v_x, y, v_y, z, v_z). Using the same methods outlined in chapter 3, these parameters were converted into



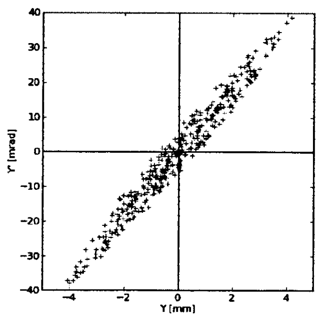
(a) *BEAM3D* output



(b) xy



(c) xx'



(d) yy'

Figure 6-2: Particle trajectory output from *BEAM3D*, as calculated by M. Artz. Beam ellipses extracted from output file using *pybeam1d*

equivalent parameters in the phase space (x, x', y, y', p).

The transport of the resulting 400 particle phase space was then simulated using both the *pybeam1d* and *Opera 3D* beam codes.

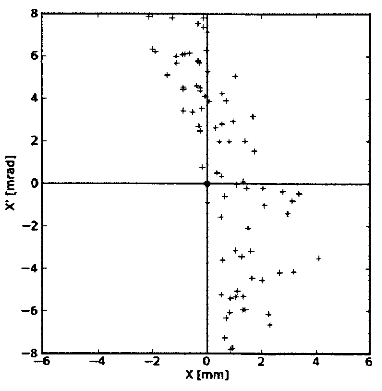
6.2.1 BBGS Without Space Charge

At first, the beam simulation was simplified by ignoring the effects of space charge in both the 1D and 3D simulations. Since the ECRIS would be running at low current (2 mA), it was estimated that space charge would not be important. This assumption will be tested in section 6.2.2.

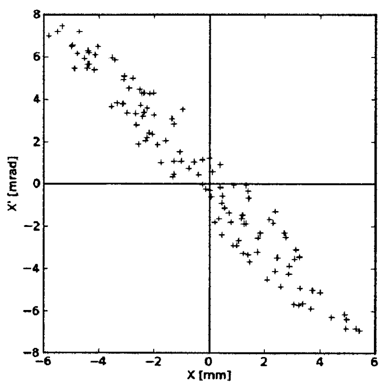
For the 1D calculations, the dipole magnet was treated with a field index $n = 0$ and with edge quadrupole focusing elements. The parameters for the quadrupole were determined using the same methods outlined in section 4.1, but for the new operating current.

The results of the 1D and 3D calculations for the phase space evaluated at the converter plates are shown in figure 6-3. Although it is not certain where emittance measurements will be taken along the beam-line, these results give a reference and could be quickly recalculated for different locations along the beam-line. Due to the large initial divergence of the beam, the beam must be cut by the divergence slits. For the simulations shown, the beam was cut at a radius of 2.5 mm by the first divergence slit.

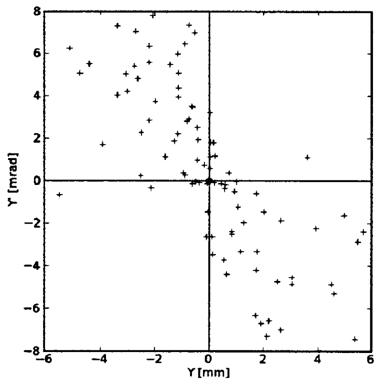
The *Opera* and *pybeam1d* results for the ECRIS beam transport did not agree well, except for maximum envelope parameters. This was because the ECRIS beam takes up a significant portion of the dipole aperture, so *Opera* captures higher-order effects which were not simulated in *pybeam1d*. These effects could be represented using a higher order ($m > 2$) field expansion near the dipole edges, but this calculation was left for future work.



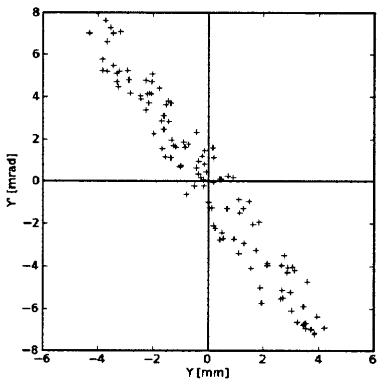
(a) Opera xx'



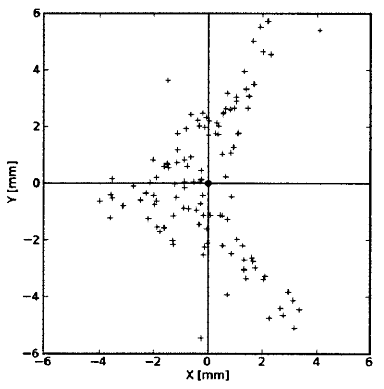
(b) pybeam1d xx'



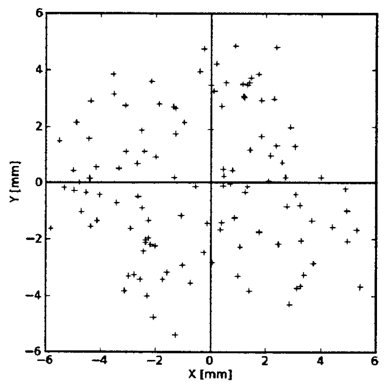
(c) Opera yy'



(d) pybeam1d yy'



(e) Opera xy



(f) pybeam1d xy

Figure 6-3: Comparison of *Opera* and *pybeam1d* phase space at the converter plates for 2 mA ECRIS beam. The triangular beam in *Opera* xy is indicative of a second-order aberration.

6.2.2 BBGS Basic Space Charge Forces

In order to verify the assumption that space charge would not be important for the 2mA beam, as well as to facilitate the future transport of more intense test beams, the *pybeam1d* code was modified to include basic integration of space charge forces. Following from Antaya and Xie (1987), the electric field applied to a particle at beam radius r due to space charge was assumed to be that of a cylindrically symmetric beam with the total current of all the particles enclosed by r:

$$E_r(r) = \frac{1}{2\pi\epsilon_o r}\sum_i \frac{I_i}{v_{zi}} \tag{6.2}$$

where I_r is the current assigned to quasi-particle i and v_{zi} is the velocity of quasi-particle i in the direction of the beam. Since it is not possible to simulate every electron with charge e, each particle in the simulation is assigned an equal portion of the total beam current, I_i.

Assuming a small displacement along the beam direction, ΔS, the extra components to add to r become:

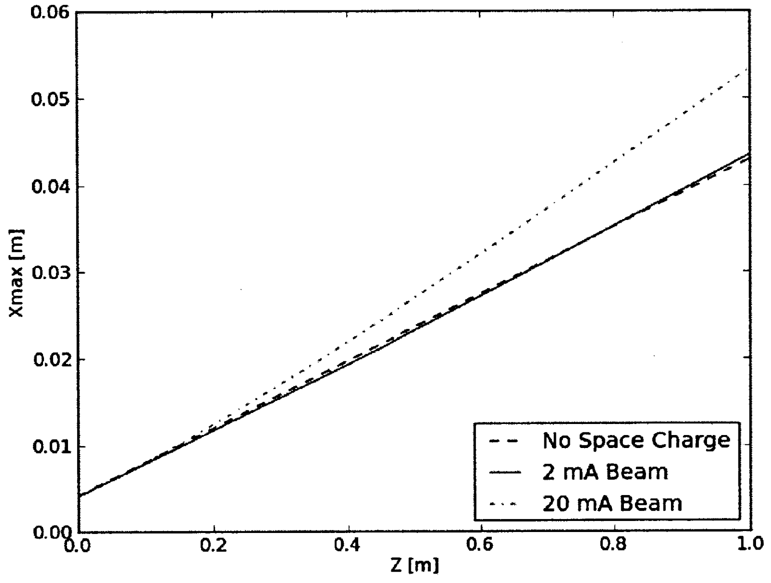
$$\Delta r = \frac{1}{2}\frac{q}{mv_z^2}E_r\,\Delta S^2 \tag{6.3}$$

and for r':

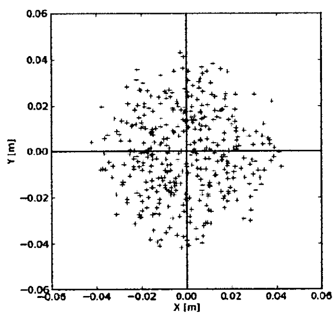
$$\Delta r' = \frac{q}{mv_z^2}E_r\,\Delta S \tag{6.4}$$

While this is not a robust or efficient integration scheme, the step size ΔS was reduced until the results for different step sizes converged. It should be noted that this was meant to be a quick calculation, and hence could certainly be improved in the future.

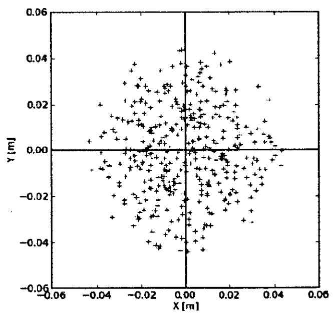
The space charge integration routine was used for the ECRIS 2 mA beam parameters in a 1.0 m drift space in order to understand the importance of space charge for low current ECRIS operation. The beam envelopes and cross sections after 1.0 m of drift are shown in figure 6-4. The 2.0 mA beam uses 400 quasi-particles, each with 5.0 μA of current. The 20.0 mA beam was not simulated at that current using *BEAM3D*; instead, the phase space of the 2.0 mA beam was used and each quasi-particle current was scaled by a factor of 10. This allowed for a simple estimate of the importance of space charge, and can be used as a reference to drive future calculations. Clearly, the results of this analysis showed that for the 2 mA beam space charge could be reasonably neglected.



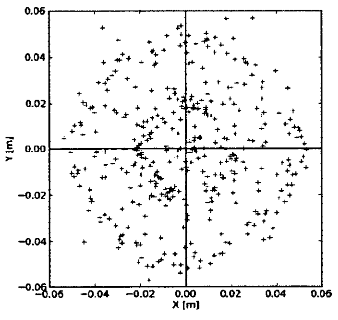
(a) Beam envelope from *pybeam1d*



(b) No Space Charge



(c) 2 mA Beam



(d) 20 mA Beam

Figure 6-4: Beam envelope and final beam cross sections for the ECRIS beam in a 1.0 m drift space. The 20 mA beam has a clear core and halo, an effect which would be expected for such a high current.

6.3 Testing Plan

Once the BBGS and ECRIS are commissioned in late Spring/ early Summer 2011, it will be important to test the actual system performance against the calculations performed for this thesis. The measurements will include, but will not be limited to:

- Measurement of dipole B_0 vs applied current to give operating parameters for different beam energies
- Measurement of fringe field for comparison with simulated fields
- Steering magnet field measurement and analysis, concentrating on possible errors as a result of non-ideal coils
- ECRIS phase space analysis using the techniques described by Xie (1989)

These and other measurements will be carried out by M. Artz and the author.

Chapter 7

Conclusion

We designed an electron beam transport system for the ISIS Bremsstrahlung Beam Generation System using well established methods for the design of iron-dominated electromagnets. The design process involved creating generally applicable components, then specifying those components as the input constraints became more certain. This resulted in a series of magnets which were finely tuned for the electron beam parameters which were provided by the engineers from Advanced Energy Systems.

Throughout the design process, a transport code and several smaller scripts were developed in order to verify the magnet designs with several independent methods. Since these codes may prove to be useful in future beam design at the PSFC Technology and Engineering Division, they were documented and are provided in the Appendices of this thesis.

During the course of this project, many opportunities for future work and improvements to the BBGS design were identified. A selection of ideas for continued work is provided below.

Dipole Field Design for Energy Variance Beam simulations performed using *Opera 3D* and *pybeam1d* showed that a negative field index within the dipole magnet could help to limit the asymmetric radial beam spreading which results from an input beam with high momentum variance. Since the magnet edges provide more than enough axial focusing, future design iterations to the BBGS dipole could include a shaped pole to force a small negative field gradient. This type of re-working could be important if it is found that the LINAC electron beam has a larger momentum variance than that of the simulated electron beams.

Additional Beam Shaping Components While the MCNPX calculations performed at Raytheon do not suggest that precise beam size will be an important factor in achieving a large signal to noise ratio, experiments may prove otherwise. In this case, future revisions of the ISIS BBGS may include additional components such as a solenoid or quadrupole focusing magnet. These components could also allow for on-line adjustments of beam shape and size in the case that the electron beam parameters vary significantly with energy. Such components were not considered in this thesis due to the limited space allocated for the entire BBGS.

Improved Converter Plate Design Based on the thermal calculations performed for this thesis, it may be necessary to redesign the converter plate assembly in order to improve the extraction of heat from the plates. At a minimum, the temperature of the plates should be carefully monitored during testing and additional water cooling should be made available in case the flow rates estimated in this thesis will not be sufficient.

Appendix A

Beam Transport Code

A.1 pybeam1d Source Code

```
## PYBEAM1D ------------------------------------------------------------
## Beam Generation and transport in 1D
## Author: R.E. Block, reblock@alum.mit.edu
## Created: 2/23/11
## Modified: 4/28/11
## Filename: beam1d.py
## ---------------------------------------------------------------------

## Imports, etc --------------------------------------------------------

from numpy import *
from random import random
from random import gauss
from matplotlib.pyplot import *

## ---------------------------------------------------------------------

## Functions -----------------------------------------------------------

def readecr(name):
# For reading BEAM3D output files
    f=open(name,'r')
    A=f.readlines()
    for i in range(0,len(A)):
        A[i] = A[i].split()
        A[i][-1] = A[i][-1][:-1]
        for j in range(0,len(A[i])):
            A[i][j]=float(A[i][j])
    B=[]
    z=A[0][1]
    for i in range(0,len(A)):
        if A[i][1]==z:
            B.append(A[i])
    B=array(B).T
    return B

def rbeam(rmax,rpmax,npoints):
# Beam generation , distributed randomly in r,r'
    theta=zeros(npoints)
    for i in range(0,npoints):
        theta[i]=random()*2*pi
    u=zeros(npoints)
    for i in range(0,npoints):
        u[i]=random()
    r=zeros(npoints)
    r=rmax*sqrt(u)
    return r,theta

def xy(r,theta):
# Conversion from polar coordinates
    x=r*cos(theta)
    y=r*sin(theta)
    return x,y

def xpyp(r,rp):
# Generate x and y ellipses from r,r'
    theta = zeros(len(r))
    for i in range(0,len(r)):
        theta[i] = random()*2*pi
    x = r*cos(theta)
    xp= rp*cos(theta)
    y = r*sin(theta)
    yp= rp*sin(theta)
    return x,xp,y,yp

def plotellipse(x,y,xlab,ylab,color,xlimits=None,ylimits=None):
# For creating standardized plots
    figure(None,figsize=(6,6))
    plot(x,y,color)
    if xlimits != None:
        xlim(xlimits)
    if ylimits != None:
        ylim(ylimits)
    xlabel(xlab)
    ylabel(ylab)
    axhline(0,0,1,color='k')
    axvline(0,0,1,color='k')
    #show()

def drift(x,xp,s):
# Simple drift region (for x or y)
    xnew = x+xp*s
    xpnew=xp
    return xnew,xpnew

def dipole(x,xp,dp,r,t,u1,u2,n=0):
# Dipole transport for radial direction
    d=(1.0-n)**(0.5)
    x1,xp1 = edge(x,xp,r,u1)
    x2 = x1*cos(d*t) + xp1*r/d*sin(d*t) + dp*r/d**2*(1-cos(d*t))
    xp2 = -x1*sin(d*t)*d/r + xp1*cos(d*t) + dp/d*sin(d*t)
    xnew,xpnew = edge(x2,xp2,r,u2)
    return xnew, xpnew

def spacedrift(x,xp,y,yp,s,ds,v,I,q=1.612e-19,m=1.6726e-27,N=25):
# Drift with space charge, default ion is proton
    n=int(round(s/ds))
    ds=s/n
    envelope=zeros((3,n+1))
    envelope[0]=linspace(0,s,n+1)
    envelope[1][0]=max(x)
    envelope[2][0]=max(y)
    for i in range(0,n):
        forces=spaceforce(x,y,q,m,v,I,N)
        dx,dxp,dy,dyp=dxy(x,xp,y,yp,forces,ds)
        x,xp=drift(x,xp,ds)
        y,yp=drift(y,yp,ds)
        x=x+dx
        xp=xp+dxp
        y=y+dy
        yp=yp+dyp
        envelope[1][i+1]=max(x)
        envelope[2][i+1]=max(y)

    return x,xp,y,yp,envelope
```

```
def spaceforce(x,y,q,m,v,I,N=25):
# Calculates force on all particles using cylindrically symmetric fields
    factor=q*I/(m*2*pi*8.85e-12*v**3)
    forces=zeros(len(x))
    r=sqrt(x**2+y**2)
    rmax=max(r)
    rmin=min(r)
    rspace=linspace(rmin-rmin/N,rmax+rmax/N,N+1)
    rcum=zeros(len(rspace))
    for i in range(0,len(rspace)):
        for j in range(0,len(r)):
            if r[j]<rspace[i]:
                rcum[i]=rcum[i]+1
    rcum=rcum*factor
    for i in range(0,len(r)):
        for j in range(1,len(rspace)):
            if r[i]<r[j]:
                forces[i]=rcum[j-1]/r[i]
                break

    return forces

def xyforces(x,y,forces):
# Convert radial force to x and y
    r=sqrt(x**2+y**2)
    xforce=forces*x/r
    yforce=forces*y/r
    return xforce,yforce

def dxy(x,xp,y,yp,forces,ds):
# Calculate displacement resulting from space charge forces
    xforce,yforce=xyforces(x,y,forces)
    dx =xforce*0.5*ds**2
    dxp=xforce*ds
    dy =yforce*0.5*ds**2
    dyp=yforce*ds
    return dx,dxp,dy,dyp

def dipoley(y,yp,r,t,u1,u2,n=0):
# Dipole in axial transverse direction
    y1,yp1 = edgey(y,yp,r,u1)
    if n==0:
        y2 = y1 + r*t*yp1 # since sin(psi)/eps = 0/0 = theta in limit
        yp2 = 0.0 + yp1
    elif n>0:
        d=sqrt(n)
        y2 = y1*cos(d*t) + yp1*r/d*sin(d*t)
        yp2 = -y1*sin(d*t)*d/r + yp1*cos(d*t)
    else:
        d=sqrt(-n)
        y2 = y1*cosh(d*t) + yp1*r/d*sinh(d*t)
        yp2= y1*sinh(d*t)*d/r + yp1*cosh(d*t)
    ynew,ypnew = edgey(y2,yp2,r,u2)
    return ynew, ypnew

def edge(x,xp,r,u):
# Edge focusing in radial direction
    xnew = x
    xpnew = tan(u)/r*x + xp
    return xnew, xpnew

def edgey(y,yp,r,u):
# Edge focusing in axial direction
    ynew = y
    ypnew = -tan(u)/r*y + yp
    return ynew, ypnew

def edgeyquad(y,yp,kq,s):
# Quadrupole in focusing plane
    ynew =      cos(kq*s)*y + 1/kq*sin(kq*s)*yp
    ypnew= -kq*sin(kq*s)*y +       cos(kq*s)*yp
    return ynew, ypnew

def edgexquad(x,xp,kq,s):
# Quadrupole in defocusing plane
    xnew =      cosh(kq*s)*x + 1/kq*sinh(kq*s)*xp
    xpnew=  kq*sinh(kq*s)*x +       cosh(kq*s)*xp
    return xnew,xpnew

def setpvar(pvar,n,mu=0):
# Generate momentum spread from Gaussian distribution
    dp = zeros(n)
    sig=sqrt(pvar)
    for i in range(0,n):
        dp[i] = gauss(mu,sig)
    return dp

def runbbgs(x,xp,y,yp,dp):
# Run entire BBGS magnet system with final parameters, no quad edges
    # Drift distances and transport const.
    d1 = 334.5 # mm
    rbnd5 = 287.5 # mm
    tbnd5 = pi/2
    ubnd5 = pi/6 # 30 degree edge angles
    d2 = 278.5 # mm
    d3 = 72.6
    ry = 0 # need to specify
    ty = pi/36
    d4 = 79
    rx = 0 # specify
    tx = pi/12
    d5 = 92.4
    # Now perform operations in beam-line
    # Start with drift from LINAC to BND5
    x2,xp2 = drift(x,xp,d1)
    y2,yp2 = drift(y,yp,d1)
    # Now insert dipole with 30 degree edges
    x3,xp3 = dipole(x2,xp2,dp,rbnd5,tbnd5,ubnd5,ubnd5)
    y3,yp3 = dipoley(y2,yp2,rbnd5,tbnd5,ubnd5,ubnd5)
    # Drift to BPM
    x4,xp4 = drift(x3,xp3,d2)
    y4,yp4 = drift(y3,yp3,d2)
    # Drift to end
    x5,xp5 = drift(x4,xp4,d3+d4+d5)
    y5,yp5 = drift(y4,yp4,d3+d4+d5)
    return x5,xp5,y5,yp5

#  End pybeam1d code specification
## ------------------------------------------------------------------------
```

A.1.1 Example Input File: Space Charge

```
## SPACECHARGE ——————————————————————————————————————————————
## Example file for using pybeam1d
## Author: R.E. Block, reblock@alum.mit.edu
## Created: 4/12/11
## Modified: 4/28/11
## ——————————————————————————————————————————————————————————

## Imports ——————————————————————————————————————————————————
from numpy import *
from beam1d import *
try:
    import cPickle as pickle
except:
    import pickle
## ——————————————————————————————————————————————————————————

## User inputs ——————————————————————————————————————————————
savecorfile = 'ecris/out-2mA-edit.txt'    # Name of input file
fignames='ecris/ecr-opera'                # Name for output files
Ep=938.272                                # Proton rest mass energy
q=1.612e-19                               # charge in Coulomb
m=1.6726e-27                              # mass in kg
I = 5.0E-6                                # current per particle in Ampere
S1 = 1.0                                  # drift in meters
dS = 0.001                                # integration step size
tometers=1.0E-3
tomm=1.0E3
limitsx=[-0.06,0.06]
## ——————————————————————————————————————————————————————————

## Execution ————————————————————————————————————————————————
# Start by loading file
data=readecr(savecorfile)  # load file
x= data[2]*tometers        # computation must be done in meters
y= data[3]*tometers
vx=data[4]
vy=data[5]
vz=data[6]
et=data[7]
el=data[8]
xp=arctan(vx/vz)
yp=arctan(vy/vz)
T=sqrt(et**2+el**2)*1E-6   # calculate T and convert to MeV
```

```
pcalc=sqrt(T**2+2*T*Ep)          # P in MeV/c
vzave=mean(vz)*1.0E6         # average velocity in m/s
npoints=len(x)
dp=zeros(npoints)
# Now run test with space charge
xs,xps,ys,yps,envelope = spacedrift(x,xp,y,yp,S1,dS,vzave,I)
xs2,xps2,ys2,yps2,envelope2 = spacedrift(x,xp,y,yp,S1,dS,vzave,I*10.0)
x1,xp1 = drift(x,xp,S1)
y1,yp1 = drift(y,yp,S1)
# Plot results
plotellipse(xs,ys,'X [m]','Y [m]','k+',limitsx,limitsx)
plotellipse(xs2,ys2,'X [m]','Y [m]','k+',limitsx,limitsx)
plotellipse(xs,xps,'X [m]',"X' [rad]",'k+')
plotellipse(x1,y1,'X [m]','Y [m]','k+',limitsx,limitsx)
# Plot envelope of beam
figure()
plot(array([0.0,S1]),array([max(y),max(y1)]),'k--')
plot(envelope[0],envelope[1],'b')
plot(envelope2[0],envelope2[1],'r-.')
xlabel('Z [m]')
ylabel('Xmax [m]')
legend(['No Space Charge','2 mA Beam','20 mA Beam'],loc='lower right')
show()
##------------------------------------------------------------------------
```

A.1.2 Example Input File: Parmela input

```
## RUNSAVECOR ------------------------------------------------------------------
## Example usage of pybeam1d to run a Parmela save file (savecor)
## Author: R.E. Block, reblock@alum.mit.edu
## Created: 4/14/11
## Modified: 4/14/11
## -----------------------------------------------------------------------------

## Imports ---------------------------------------------------------------------
from numpy import *
from beam1d import *
import beamgen5 as bg
try:
    import cPickle as pickle
except:
    import pickle
## -----------------------------------------------------------------------------

## User inputs -----------------------------------------------------------------
savecorfile = '../SAVECORtest.txt'
fignames = 'beam1dsavecor607'
npoints = 5000
xlimits=[-10,15]
ylimits=[-20,30]
## -----------------------------------------------------------------------------

## Now problem execution -------------------------------------------------------
# Grab savecor beam
mybeam = bg.beamgen5(savecorfile,npoints,fignames,True)
mybeam.printdata()
x = mybeam.x*10
y = mybeam.y*10
dp= mybeam.T/mean(mybeam.T) - 1.0
dp=dp#*0.0
xf,xpf,yf,ypf = runbbgs(x,mybeam.xp,y,mybeam.yp,dp)
plotellipse(-xf,yf,'X_[mm]','Y_[mm]','k+',[-10,15],[-12.5,12.5])
if 0:
    figure(None, figsize=(6,6))
    plot(-xf[0:5000],mybeam.T[0:5000]/1E6,'r+')
    xlabel('X_[mm]')
    ylabel('T_[MeV]')
show()
## -----------------------------------------------------------------------------
```

A.2 Opera Beam Generation

```
## BEAMGEN ----------------------------------------------------------
## Creates generic beams and uses beam output from Parmela
## Author: R.E. Block, reblock@alum.mit.edu
## Created: 7/27/10
## Modified: 4/28/11
## Filename: beamgen.py
## Notes: see end of file for example usage
## ------------------------------------------------------------------

## Imports ----------------------------------------------------------
from math import pi
from pylab import *
try:
    import cPickle as pickle
except:
    import pickle
## ------------------------------------------------------------------

## User Inputs and Constants ----------------------------------------
TESTING = False
tl1 = 'TRACK X0='
tl2 = ' Y0='
tl3 = ' Z0=' # 75.467
tl3a= ' THETA=' #was 81.2 for Z0, -18.5 error
tl4 = ' PHI='
tl5 = ' PSI=0 VOLTS='
tl6 = ' STEP=1 NSTEP=1500 OPTION=PARTICLE FILE=' #was step=1
tl7 = ' STATUS='
tl8 = ' PRINT=NO DISPLAY=YES\n'
## ------------------------------------------------------------------

## Class and Function Definitions -----------------------------------
class beamgen:
# Class for generating Opera beams from AES save files and pybeam1d
    def __init__(self, savefile, n, fignames, readsave=True):
        self.maxN = n
        self.savecor = savefile
        self.fignames = fignames
        if readsave:
            self.readSAVECOR()

    def readSAVECOR(self):
    # Used to read save file from Parmela
```

```
f = open(self.savecor, 'r')
filedump = f.readlines()
f.close()
del filedump[0]
del filedump[0]
for i in range(0,len(filedump)):
    filedump[i] = filedump[i].replace('D','E')
    filedump[i] = filedump[i].split()
    for j in range(0,len(filedump[i])):
        filedump[i][j] = float(filedump[i][j])

self.P = array(filedump)
self.P = self.P.T

self.xp = -self.P[1]/self.P[5]
self.yp = self.P[3]/self.P[5]

self.x = -self.P[0]
self.y = self.P[2]

self.N = len(self.P[0])

self.rmsx = sqrt(sum(self.x**2)/self.N)
self.rmsy = sqrt(sum(self.y**2)/self.N)

self.rmsxp= sqrt(sum(self.xp**2)/self.N)
self.rmsyp= sqrt(sum(self.yp**2)/self.N)

self.maxx = max(self.x)
self.maxy = max(self.y)

self.maxxp = max(self.xp)
self.maxyp = max(self.yp)

self.z = self.P[4] - mean(self.P[4])
self.rmsz = sqrt(sum(self.z**2)/self.N)

self.bg = sqrt(self.P[1]**2+self.P[3]**2+self.P[5]**2)
self.g  = sqrt(self.bg**2 +1)
self.T  = (self.g-1)*0.511e6

self.r = sqrt(self.x**2+self.y**2)
self.maxr = max(self.r)
self.rmsr = sqrt(sum(self.r**2)/self.N)
```

```
    def alterInit(self,P):
    # Used for reading from pybeam1d generated array
        self.x = P[0]
        self.y = P[2]

        self.xp = P[1]
        self.yp = P[3]

        self.N = len(P[0])

        self.rmsx = sqrt(sum(self.x**2)/self.N)
        self.rmsy = sqrt(sum(self.y**2)/self.N)

        self.rmsxp= sqrt(sum(self.xp**2)/self.N)
        self.rmsyp= sqrt(sum(self.yp**2)/self.N)

        self.maxx = max(self.x)
        self.maxy = max(self.y)

        self.maxxp = max(self.xp)
        self.maxyp = max(self.yp)

        self.z = zeros(self.N)
        self.rmsz = 0.0

        self.T = P[4]

        self.r = sqrt(self.x**2+self.y**2)
        self.maxr = max(self.r)
        self.rmsr = sqrt(sum(self.r**2)/self.N)

        self.maxN = self.N


    def printplots(self):
        Ntext = 'N␣=␣' + str(self.maxN)

        figure(1,figsize=(6,6))
        plot(self.x[0:self.maxN],self.y[0:self.maxN],'r+')
        xlabel('X␣[cm]')
        ylabel('Y␣[cm]')
        axis('scaled')
        annotate(Ntext,(self.maxx*.7,self.maxy*-.7))
        savefig(self.fignames+'xy.png')
        figure(2,figsize=(6,6))
        plot(self.xp[0:self.maxN]*1000,self.yp[0:self.maxN]*1000,'r+')
        xlabel("X'␣[mrad]")
        ylabel("Y'␣[mrad]")
        axis('equal')
        annotate(Ntext,(self.maxxp*1000*.7,self.maxyp*1000*-.7))
        savefig(self.fignames+'xpyp.png')

        figure(3,figsize=(6,6))
        plot(self.x[0:self.maxN],self.xp[0:self.maxN]*1000,'r+')
        xlabel("X␣[cm]")
        ylabel("X'␣[mrad]")
        annotate(Ntext,(self.maxx*.7,self.maxxp*1000*-.7))
        savefig(self.fignames+'xxp.png')

        figure(4,figsize=(6,6))
        plot(self.y[0:self.maxN],self.yp[0:self.maxN]*1000,'r+')
        xlabel("Y␣[cm]")
        ylabel("Y'␣[mrad]")
        annotate(Ntext,(self.maxy*.7,self.maxyp*1000*-.7))
        savefig(self.fignames+'yyp.png')

    def printdata(self):

        count = 0
        for i in range(0,self.maxN):
            if self.r[i] > self.rmsr:
                count = count + 1
        percOut = float(count)/self.maxN*100

        print self.fignames
        print 'x_rms␣=␣' + str(self.rmsx)
        print 'y_rms␣=␣' + str(self.rmsy)
        print 'x_max␣=␣' + str(self.maxx)
        print 'y_max␣=␣' + str(self.maxy)
        print "x'_rms␣=␣"+ str(self.rmsxp*1000)
        print "y'_rms␣=␣"+ str(self.rmsyp*1000)
        print '<E>␣␣␣=␣' + str(mean(self.T))
        print 'Sig_E␣=␣' + str(std(self.T)/mean(self.T)*100)
        print 'r_rms␣=␣' + str(self.rmsr)
        print 'r_max␣=␣' + str(max(self.r))
        print 'PercOUt=␣' + str(percOut)

    def dumpdata(self):
        f = open(self.fignames+'pickle','w')
        datatodump = {'x':self.x*10,'y':self.y*10,'xp':self.xp,'yp':self.yp,'T':self.
            T,'dp':self.T/mean(self.T)}
        pickle.dump(data,f)
        f.close()

    def histogram(self):
        print 'Printing␣Histograms...'

        m = 200
        figure()
        self.pdf, self.bins, self.patches = hist(self.T/10**6,m,None,True)
        axis('tight')
        xlabel('Kinetic␣Energy␣[MeV]')
        ylabel('PDF(T)')
        savefig(self.fignames + 'hist')

        self.density = zeros((1,m))
        for i in range(0,m):
            self.density[0][i] = self.pdf[i]/(pi*(self.bins[i+1]**2 - self.bins[i
                ]**2))

        self.density[0] = self.density[0]/sum(self.density[0])*m
        self.cdf = []
        for i in range(0,m):
            self.cdf.append(float(sum(self.pdf[0:(i+1)])))
        self.cdf = array(self.cdf)
        self.cdf = self.cdf/sum(self.pdf)
        figure()
        bar(self.bins[0:m],self.density[0],self.maxr/m)
        axis('tight')
        xlabel('Radius␣[cm]')
        ylabel('Normalized␣Beam␣Density␣[per␣cm^2]')
        savefig(self.fignames + 'dens')

        figure()
        bar(self.bins[0:m],self.cdf,self.maxr/m)
        axis('tight')
        xlabel('Radius␣[cm]')
        ylabel('CDF(R)')
        savefig(self.fignames + 'cdf')


    def phiVSr(self):
        dR = 0.05
        R = 0.2
        numBins = 50
        U = []

        if self.phi == None:
            beamgen2.convertfullangles(self)

        for i in range(0,self.N):
            if self.r[i] < R:
                if self.r[i] > R-dR:
                    U.append([self.r[i],self.phi[i]])

        U = array(U)
        U = U.T

        y, bins, patches = hist(U[1],numBins,None,True)

    def todegrees(self):
        self.xpdeg = self.xp*180/pi
        self.ypdeg = self.yp*180/pi

    def convertfullangles(self):
        self.phi = arctan((tan(self.xp)**2+tan(self.yp)**2/(cos(self.xp)**2))**(.5))
        self.theta = arctan(sin(self.xp)/tan(self.yp))

    def convertangles(self):
        self.phi = arctan((tan(self.xp[0:self.maxN])**2+tan(self.yp[0:self.maxN])
            **2/(cos(self.xp[0:self.maxN])**2))**(.5))
        self.theta = arctan(sin(self.xp[0:self.maxN])/tan(self.yp[0:self.maxN]))

        for j in range(0, self.maxN):
            if self.yp[j] > 0:  # was xp<0
                self.phi[j] = -self.phi[j]

        self.theta = self.theta*180/pi
        self.phi = self.phi*180/pi

    def gencomi(self,energy,filename,trackname):
        linecount=0
        cut = (self.rmsx+self.rmsy)/2
        #print 'Writing comi file, cutting at r= ' + str(cut)
        comifile = open(filename,'w')
        for i in range(0, self.maxN):
            if 1: # self.r[i] <= cut:
                linecount=linecount+1
                comifile.write(tl1)
                comifile.write(str(self.x[i]))
                comifile.write(tl2)
                comifile.write(str(self.y[i]))
                comifile.write(tl3)
                comifile.write(str(75.467-self.z[i]))
                comifile.write(tl3a)
                comifile.write(str(self.phi[i]+180))
                comifile.write(tl4)
                comifile.write(str(90-self.theta[i]))
                comifile.write(tl5)
                comifile.write(str(self.T[i]))
                comifile.write(tl6)
                comifile.write(trackname)
                comifile.write(tl7)
                if i!=0:
                    comifile.write('OLD')
                else:
                    comifile.write('NEW')
                comifile.write(tl8)
        comifile.close()
        print 'Closing file, wrote N= '+str(linecount)
##------------------------------------------------------------------------

## Example of usage -----------------------------------------------------
if __name__ == '__main__':
    savecorfile = 'SAVECORtest.txt'
    fignames = '607-v5'
    newbeam1 = beamgen2(savecorfile,5000,fignames)
    newbeam1.printdata()
    newbeam1.printplots()
    newbeam1.histogram()
    newbeam1.convertangles()
    newbeam1.gencomi('6.07E7',fignames+'.comi',fignames)
##------------------------------------------------------------------------
```

A.3 3D Beam Analysis

```
## INTERSECTIONS ------------------------------------------------------------
## Interprets Opera intersection files and creates phase space plots
## Author: R.E. Block, reblock@alum.mit.edu
## Created: 7/28/10
## Modified: 4/28/11
## Filename: intersections.py
## --------------------------------------------------------------------------

## Imports ------------------------------------------------------------------
from math import pi
from pylab import *
## --------------------------------------------------------------------------

## User Inputs --------------------------------------------------------------
DEBUG1 = False
DEBUG2 = False
DEBUG3 = True
DEBUG4 = False
DEBUG5 = False
TESTING = False
TESTING1 = True
# ---------------------------------------------------------------------------

## Class and Function Definition --------------------------------------------
class intersection:
# Used for interpreting opera intersection files
    def __init__(self,filename,energy,color):
        self.file = filename
        self.energy = str(energy)
        self.color = color

    def importdata(self):
        vc = 2.9979245800E10

        f = open(self.file+'.txt','r')
        filedump = f.readlines()
        # print filedump[len(filedump)-1]
        f.close()
        del filedump[0]
        #print filedump[0]
        for i in range(0,len(filedump)):
#           filedump[i] = filedump[i].replace('D','E')
            filedump[i] = filedump[i].split()
        for j in range(0,len(filedump[i])):
            filedump[i][j] = float(filedump[i][j])

    self.P = array(filedump)
    self.P = self.P.T
    # [0]: Current, [1]: X, [2]: Y, [3]: Z, [4]: Vx, [5]: Vy, [6]: Vz

    # Import variables from intersections file output
    X = self.P[1]
    Xm = mean(self.P[1])
    Y = self.P[2]
    Ym = mean(self.P[2])
    Z = self.P[3]
    Zm = mean(self.P[3])
    VX = self.P[4]
    VY = self.P[5]
    VZ = self.P[6]
    Vmag = sqrt(VX**2+VY**2+VZ**2)
    Vhat = array([VX/Vmag,VY/Vmag,VZ/Vmag])

    # Calculate new coordinate directions of beam
    self.zhat = mean(Vhat,1)
    self.zhat = self.zhat/sqrt(sum(self.zhat**2))
    self.xhat = array([self.zhat[2],0,-self.zhat[0]])/sqrt(self.zhat[0]**2+self.
        zhat[2]**2)
    self.yhat = -array([self.zhat[1]*self.xhat[2]-self.zhat[2]*self.xhat[1],self.
        zhat[0]*self.xhat[2]-self.zhat[2]*self.xhat[0],self.zhat[0]*self.xhat[1]-
        self.zhat[1]*self.xhat[0]])
    self.yhat = self.yhat/sqrt(sum(self.yhat**2))

    # Calculate new coordinates of beam based on beam center
    Xnorm = X-Xm
    Ynorm = Y-Ym
    Znorm = Z-Zm

    self.x = Xnorm*self.xhat[0]+Ynorm*self.xhat[1]+Znorm*self.xhat[2]
    self.y = Xnorm*self.yhat[0]+Ynorm*self.yhat[1]+Znorm*self.yhat[2]
    self.z = Xnorm*self.zhat[0]+Ynorm*self.zhat[1]+Znorm*self.zhat[2]

    self.vx = VX*self.xhat[0]+VY*self.xhat[1]+VZ*self.xhat[2]
    self.vy = VX*self.yhat[0]+VY*self.yhat[1]+VZ*self.yhat[2]
    self.vz = VX*self.zhat[0]+VY*self.zhat[1]+VZ*self.zhat[2]

    self.xm = mean(self.x)
    self.ym = mean(self.y)
```

```
        self.bx = self.vx/vc
        self.by = self.vy/vc
        self.bz = self.vz/vc

        self.bgx = self.bx/sqrt(1-self.bx**2)
        self.bgy = self.by/sqrt(1-self.by**2)
        self.bgz = self.bz/sqrt(1-self.bz**2)

        self.Vmag2 = sqrt(self.vx**2+self.vy**2+self.vz**2)
        self.beta = self.Vmag2/vc
        self.gamma = 1/(sqrt(1-self.beta**2))
        self.T = (self.gamma-1)*0.511e6

        self.xp = self.vx/self.vz
        self.yp = self.vy/self.vz

        self.xpm = mean(self.xp)
        self.ypm = mean(self.yp)

        self.x = self.x*10
        self.y = self.y*10
        self.xp = self.xp*1000
        self.yp = self.yp*1000

    def plotbeam(self):
        fig = figure(1,figsize=(6,6))
        fig.clf()
        plot(self.x,self.y,self.color)
        plot(0,0,'ko')
        axhline(0,0,1,color='k')
        axvline(0,0,1,color='k')
        xlabel('X [mm]')
        ylabel('Y [mm]')
        if TESTING1:
            axis((-6,6,-6,6))
        else:
            axis('equal')
        ax = axis()
        if TESTING: print ax
        savefig(self.file+'xy.png')

        fig = figure(2,figsize=(6,6))
        fig.clf()
        plot(self.x,self.xp,self.color)
        plot(0,0,'ko')
        axhline(0,0,1,color='k')
        axvline(0,0,1,color='k')
        xlabel('X␣[mm]')
        ylabel("X'␣[mrad]")
        if TESTING1:
            axis((-6,6,-8,8))
        ax = axis()
        if TESTING: print ax
        savefig(self.file+'xxp.png')

        fig = figure(3,figsize=(6,6))
        fig.clf()
        plot(self.y,self.yp,self.color)
        plot(0,0,'ko')
        axhline(0,0,1,color='k')
        axvline(0,0,1,color='k')
        xlabel('Y␣[mm]')
        ylabel("Y'␣[mrad]")
        if TESTING1:
            axis((-6,6,-8,8)) # was 2 and 8
        ax = axis()
        if TESTING: print ax
        savefig(self.file+'yyp.png')

    def fitellipses(self,indicator):
        if indicator == 'xy':
            yy = self.x
            yyp = self.y
            labels=['X␣[mm]','Y␣[mm]']
        elif indicator == 'xxp':
            yy = self.x
            yyp = self.xp
            labels=['X␣[mm]',"X'␣[mrad]"]
        elif indicator == 'yyp':
            yy = self.y
            yyp = self.yp
            labels=['Y␣[mm]',"Y'␣[mrad]"]
        else:
            yy = self.xp
            yyp = self.yp
            labels=["X'␣[mrad]","Y'␣[mrad]"]

        ycm = 0
        ypcm = 0
```

```
i11 = mean(yyp**2)
i22 = mean(yy**2)
i12 = mean(yy*yyp)

Tyyp = array([[i11,i12],[i12,i22]])
w,v = eig(Tyyp)
self.w = w
self.v = v

ang = arctan(v[0][1]/v[0][0])-pi/2
phi = arange(0,2*pi,pi/80)
a = 2*sqrt(w[0])
b = 2*sqrt(w[1])
u = 2*sqrt(w[0]) * cos(phi)
v = 2*sqrt(w[1]) * sin(phi)

up = ycm + (u*cos(ang)) - (v*sin(ang))
vp = ypcm + (u*sin(ang)) + (v*cos(ang))

uint = ycm + (2*sqrt(w[0]) * cos(0))*cos(ang)
A = cos(ang)**2/a**2 + sin(ang)**2/b**2
B = -2*cos(ang)*sin(ang)*(1/a**2-1/b**2)
C = sin(ang)**2/a**2 + cos(ang)**2/b**2

area = pi * a * b
epsu = a*b

ghat = A*epsu
bhat = C*epsu
ahat = B*epsu/2

fig = figure(None,figsize=(6,6))
plot(self.y,self.yp,self.color)
plot(up,vp,'k.-')
plot(0,0,'ko')
axhline(0,0,1,color='k')
axvline(0,0,1,color='k')
xlabel(labels[0])
ylabel(labels[1])
savefig(self.file+indicator+'fit.png')

print 'Twiss Parameters for '+ self.file+':'
print 'alpha = ' + str(ahat)
print 'beta = ' + str(bhat)
```

```
        print 'gamma␣=␣' + str(ghat)
        print 'emmitance␣=␣' + str(epsu)

        return (ahat,bhat,ghat,epsu,up,vp)

    def histogram(self):
        print 'Printing␣Histograms...'
        m = 200

        figure()
        self.xpdf, self.xbins, self.xpatches = hist(self.x,m,None,True)
        axis('tight')
        xlabel('X␣[mm]')
        ylabel('PDF(X)')
        savefig(self.file + 'hist-x')

        figure()
        self.xppdf, self.xpbins, self.xppatches = hist(self.xp,m,None,True)
        axis('tight')
        xlabel("X'␣[mrad]")
        ylabel("PDF(X')")
        savefig(self.file + 'hist-xp')

        figure()
        self.ypdf, self.ybins, self.ypatches = hist(self.y,m,None,True)
        axis('tight')
        xlabel('Y␣[mm]')
        ylabel('PDF(Y)')
        savefig(self.file + 'hist-y')

        figure()
        self.yppdf, self.ypbins, self.yppatches = hist(self.yp,m,None,True)
        axis('tight')
        xlabel("Y'␣[mrad]")
        ylabel("PDF(Y')")
        savefig(self.file + 'hist-yp')

        figure(None,figsize=(6,6))
        self.Tpdf, self.Tbins, self.Tpatches = hist(self.T/10**6,m,None,True,histtype
            ='stepfilled',color=self.color[0])
        #axis('tight')
        xlabel('T␣[MeV]')
        ylabel('PDF(T)')
        #plot(self.Tx,self.BfT,'g-',linewidth='2')
        savefig(self.file + 'hist-T')
```

```
def printdata(self):
    f = open(self.file+'-raw.csv','w')
    f.write('ISIS_BBGS_Electron_Tracking:_version_5\n')
    f.write('Rob_Block,\t10SEP2010,\t'+self.file+'\n')
    f.write('Particle_ID,\tX,\tX_p,\tY,\tY_p,\tT\n')
    f.write('[#],\t[mm],\t[mrad],\t[mm],\t[mrad],\t[MeV]\n')

    for i in range(0,len(self.x)):
        f.write(str(i+1)+',\t')
        f.write(str(self.x[i])+',\t')
        f.write(str(self.xp[i])+',\t')
        f.write(str(self.y[i])+',\t')
        f.write(str(self.yp[i])+',\t')
        f.write(str(self.T[i]/10**6)+'\n')

    f.close()

def printhist(self):
    f = open(self.file+'-pdf.csv','w')
    f.write('ISIS_BBGS_Electron_Tracking:_Dipole_version_4\n')
    f.write('Rob_Block,\t10SEP2010,\t'+self.file+'\n')
    f.write('X_Bins,\tPDF(X),\tX_p_Bins,\tPDF(X_p),\tY_Bins,\tPDF(Y),\tY_p_Bins,\
        tPDF(Y_p),\tT_Bins,\tPDF(T)\n')
    f.write('[mm],\t-,\t[mrad],\t-,\t[mm],\t-,\t[mrad],\t-,\t[MeV],\t-\n')

    for i in range(0,len(self.xbins)-1):
        f.write(str(self.xbins[i])+',\t')
        f.write(str(self.xpdf[i])+',\t')
        f.write(str(self.xpbins[i])+',\t')
        f.write(str(self.xppdf[i])+',\t')
        f.write(str(self.ybins[i])+',\t')
        f.write(str(self.ypdf[i])+',\t')
        f.write(str(self.ypbins[i])+',\t')
        f.write(str(self.yppdf[i])+',\t')
        f.write(str(self.Tbins[i])+',\t')
        f.write(str(self.Tpdf[i])+'\n')

    i = len(self.xbins)-1
    f.write(str(self.xbins[i])+',\t')
    f.write(',\t')
    f.write(str(self.xpbins[i])+',\t')
    f.write(',\t')
    f.write(str(self.ybins[i])+',\t')
    f.write(',\t')
```

```
            f.write(str(self.ypbins[i])+',\t')
            f.write(',\t')
            f.write(str(self.Tbins[i])+',\t')
            f.write('\n')

            f.close()

    def plotspec(self):
            figure(None, figsize=(6,6))
            plot(self.x, self.T/10**6, self.color)
            xlabel('X_[mm]')
            ylabel('T_[MeV]')
            savefig(self.file + 'XT')

            TMeV = self.T/10**6

            Tm = mean(TMeV)
            Tv = var(TMeV)
            l = min(TMeV)
            h = max(TMeV)

            Tm = (Tm-l)/(h-l)
            Tv = Tv/(h-l)**2

            alpha = Tm*((Tm*(1-Tm))/Tv -1)+27
            beta = (1-Tm)*((Tm*(1-Tm))/Tv -1)+3
            print alpha,beta
            self.Tx = arange(l,h,(h-l)/100)
            self.BfT = ((self.Tx-l)/(h-l))**(alpha-1)*(1-(self.Tx-l)/(h-l))**(beta-1)
            self.BfT = self.BfT/(sum(self.BfT)*(h-l)/(len(self.BfT)-1))
            xtest = arange(0,1,0.01)
            Btest = xtest**(alpha-1)*(1-xtest)**(beta-1)

def percentGreaterThan(z_arr,z0):
    count = 0
    for i in range(0,len(z_arr)):
        if z_arr[i]>z0:
            count = count + 1
    PGT = float(count)/len(z_arr)*100
    return PGT
## ---------------------------------------------------------------------------

## Execution Examples -------------------------------------------------------
if DEBUG1:
    filenames = ['607A','607Crms','607Cxy','306A','306Crms','306Cxy','62A','62Crms','
        62Cxy']
    beams = []
    colors = ['r+','b+','g+']
    # filename = '607Crms'
    energy = '60.7'
    ellipsedata = []
    # color = 'b+'
    for i in range(0,len(filenames)):
        beams.append(intersection(filenames[i],energy,colors[i%3]))
        beams[i].importdata()
        print 'Data_#' + str(i) + '_imported'
        ellipsedata.append((beams[i].fitellipses('xxp'),beams[i].fitellipses('yyp'),
            beams[i].fitellipses('xy')))
        print 'Ellipses_fitted_for_data_#' + str(i)

    #beam1 = intersection(filename,energy,color)
    #beam1.importdata()
    # beam1.plotbeam()
    #beam1.fitellipses()
    dummy = 1

if DEBUG2:
    for i in range(0,len(ellipsedata)):
        print '&_{0:2.1f}_&_{1:2.2f}_&_{2:2.2f}_&_{3:.3f}_&_{4:.3f}_\\\\'.format(
            float(len(beams[i].x))/5000*100,sqrt(ellipsedata[i][2][1]*ellipsedata[i
            ][2][3]),sqrt(ellipsedata[i][2][2]*ellipsedata[i][2][3]),ellipsedata[i
            ][0][3],ellipsedata[i][1][3])

if DEBUG3:
    #beamv3 = intersection('607-fullsym-v3-nocut-test','60.7','b+')
    #beamv3 = intersection('607-fullsym-v3-inter','60.7','b+')
    #beamv3 = intersection('306-v3-nonlin-full','30.6','b+')
    #beamv3 = intersection('306-v3-nonlin-cut','30.6','b+')
    #beamv3 = intersection('607-v3-nonlin-cut','30.6','g+')
    #beamv3 = intersection('62-v3-cut-test','6.2','b+')
    #beamv3 = intersection('607-steer-v3-cutxy','6.2','b+')
    #beamv3 = intersection('bnd5-607-intersections','60.7','k+')
    #beamv3 = intersection('306-v5-intersections','30.6','r+')
    #beamv3 = intersection('../opera/BND5/intersections/607-v5a/oaes607','60.7','k+')
    #beamv3 = intersection('../opera/BND1/int/607-bnd1a-int','60.7','k+')
    #beamv3 = intersection('../opera/BND5_comb/int/607-bnd5-str5-int','60.7','k+')
    beamv3 = intersection('beamld/ecris/ecr-opera-int','60.7','k+')
    beamv3.importdata()
    #beamv3.plotspec()
    beamv3.plotbeam()
    beamv3.histogram()
    #beamv3.printdata()
    #beamv3.printhist()
    pgt = percentGreaterThan(abs(beamv3.x),2.5)
    print 'Percent beam within 2.5 mm = ' + str(100-pgt) + ' %'

if DEBUG4:
    pgt = percentGreaterThan(beamv3.x,5.0)
    print str(pgt) + ' %'
##---------------------------------------------------------------------------
```

Appendix B

Miscellaneous Code

B.1 Opera 3D Multipole Analysis

```
## MULTIPOLE ------------------------------------------------------------
## For generation of comi files and analysis of output from Opera
## Author: R.E. Block, reblock@alum.mit.edu
## Created: 4/01/11
## Modified: 4/28/11
## Filename: multipole.py
##-----------------------------------------------------------------------


## Imports --------------------------------------------------------------
from numpy import *
from matplotlib.pyplot import *
from scipy import fftpack
from numpy import concatenate as concat
##-----------------------------------------------------------------------


## User defined inputs --------------------------------------------------
r0 = 28.75              # Bending radius of Dipole Magnet
rb = 0.75               # Radius to evaluate field
b0=6622.0               # Central field of dipole
base = 'bnd5-t1'        # Name of base-file
dz=0.5                  # Length between field evaluations
zmax=7.5                # Length from magnet edge to stop field calcs
dt=dz/r0
t1=arange(-dt*4,0.0,dt)+pi/4
z1=zeros(len(t1))
z2=arange(0,zmax+dz,dz)
t2=ones(len(z2))*pi/4
```

```
t = concat((t1,t2))   # Theta values in Opera coords
z = concat((z1,z2))   # Z values in Opera coords

extraz=zeros(len(z))
extraz[:4]=arange(-dz*4,0.0,dz)

nt=50
pathlen=z+extraz
legendtext=[]
for i in range(0,len(pathlen)):
    legendtext.append(str(pathlen[i])[0:5]+ '_cm')
legendspec=['b-','r--','k-.','g-','m--']*4

if 1: print legendtext

# Program control - pick between generating comi file and reading tables
if 1:
    comi=False
    loadtbl=True
    plotall=False
    calcfft=True
else:
    comi=True
    loadtbl=False
    plotall=False
    calcfft=False
## -----------------------------------------------------------------------

## Functions -------------------------------------------------------------

def readtable(name,nhead):
    f=open(name,'r')
    for i in range(0,nhead):
        f.readline()
    data2d = f.readlines()
    for i in range(0,len(data2d)):
        data2d[i] = data2d[i].split()
        data2d[i][-1] = data2d[i][-1][:-1]
        for j in range(0,len(data2d[i])):
            data2d[i][j]=float(data2d[i][j])
    data2d = array(data2d).T
    return data2d

def writecircle(r,z,u,nt=50,t0=0,tm=360):
    toprint= 'CIRCLE_RADIUS='+str(r)+'_TH1=' + str(t0)+'_TH2='+str(tm)+'_ZC='+str(z)+
        '_NP='+str(nt)+'_BUFFER=CIRCLE'+str(u)+'\n'
    return toprint

def changecoord(r0,theta):
    dz = -r0*cos(theta)
    dx = r0*sin(theta)
    toprint= 'SET_XLOCAL='+str(dx)+'_YLOCAL=0_ZLOCAL='+str(dz)+'_PLOCAL=0_TLOCAL='+
        str(90-theta*180./pi)+'_SLOCAL=0\n'
    return toprint

def printtable(name,u,theta):
    t1='PLOT_FILE=TEMP_COMPONENT=Bz*'+str(cos(theta))+'-Bx*'+str(sin(theta))+'\n'
    t2='TABLE_INFILE=TEMP_OUTFILE='+str(name)+str(u)+'.table_COLUMNS=1_F1=Bz*'+str(
        cos(theta))+'-Bx*'+str(sin(theta))+'\n'
    return t1+t2
## -------------------------------------------------------------------------------

## Problem execution ----------------------------------------------------------
if comi:
    f=open(base+'.comi','w')
    for u in range(0,len(t)):
        f.write(changecoord(r0,t[u]))
        f.write(writecircle(rb,z[u],u))
        f.write(printtable(base,u,t[u]))
    f.close()

if loadtbl:
    data=[]
    for u in range(0,len(t)):
        data.append(readtable(base+str(u)+'.table',3))
    if plotall:
        #figure(None,figsize=(12,9))
        x=linspace(0,2*pi,51)
        for u in range(0,len(t)):
            if u%5==0:
                figure(None,figsize=(6.5,8))
            plot(x,data[u][0]/b0,legendspec[u%5])
            if u%5==4:
                xlabel('Angle_[radians]')
                ylabel('Bx/B0')
                xlim([0,2*pi])
                ylim([-0.15,0.15])
                legend(legendtext[u-4:u+1])
```

```
if calcfft:
    F = []
    P = []
    P2=[]
    for u in range(0,len(t)):
        F.append((fft.fft(data[u][0]/b0)))
        n = len(F[u])
        P.append(abs(F[u][0:n/2])**2)
        P2.append(F[u][0:n/2])
    P=array(P)
    P2=array(real(P2))

    figure(None,figsize=(6,6))
    for u in range(0,len(t)):
        plot(range(0,len(P[u])),P[u])

    figure(None,figsize=(10,6))
    for m in range(1,4):
        plot(pathlen,P2.T[m],legendspec[m-1])
    xlabel('Distance␣from␣edge␣[cm]')
    ylabel('Magnitude␣[Bx/B0]')
    xlim([-2.0,7.5])
    legend(['Quadrupole','Hexapole','Octupole'])

    Ba = max(P2.T[1])*b0
    Seff= 9.5/(2*Ba/b0*(len(P2.T[1])-1)) * (sum(P2.T[1][:-1])+sum(P2.T[1][1:]))
    kq=sqrt(Ba/(rb*b0*r0))
    print 'a␣=␣' + str(rb)
    print 'Ba=␣' + str(Ba)
    print 'r0=␣' + str(r0)
    print 'B0=␣' + str(b0)
    print 'kq=␣' + str(kq)
    print 'S␣=␣' + str(Seff)

if plotall or calcfft:
    show()
##-----------------------------------------------------------------------------
```

B.2 Electron Range Calculations

```
PROGRAM range
! Used to calculate range of electrons in matter
! Adapted from TABATA et al 1996
 T0 = 60.66
 Z = 6.00
 AW= 12.00
 FI = 78.0
 rho = 2.2
 r=R0(T0,Z,AW,FI)
 re=REX(T0,Z,r)
 print *, r
 print *, r/rho
 print *, re
 print *, re/rho
END PROGRAM range

FUNCTION REX(T0,Z,R0)
! PURPOSE - calculate extrapolated range of electrons
! COPIED FROM TABATA et al, 1996
! T0 - incident kinetic energy of electrons in MeV
! Z - atomic number of medium
! R0 - CSDA range
! FI - mean excitation energy in eV
! REX - extrapolated range in G/cm2
!
 TAU0=T0/0.511
 ALZ=LOG(Z)
 A1=0.3879*Z**0.2178
 A2=0.4541+0.03068*Z
 A3=3.326E-16*Z**(13.24-1.316*ALZ)
 A4=14.03/Z**0.7406
 A5=4.294E-03*Z**(1.684-0.2264*ALZ)
 A6=0.6127*Z**0.1207
 REX=R0/(A1+A2/(1.+A3/TAU0**A4+A5*TAU0**A6))
 RETURN
END FUNCTION

FUNCTION R0(T0,Z,ATW,FI)
! PURPOSE - csda range of electrons
! COPIED FROM TABATA ET AL, 1996
! T0 - kinetic energy
! Z - atomic number
! ATW - atomic weight
! FI - mean excitation energy in eV
! R0 - range in g/cm2
!
 DOUBLE PRECISION C2,W
 TAU0=T0/0.511
 FI1=FI*1.E-6/0.511
 C1=LOG((TAU0/(FI1+1.1E-6*Z**0.959*TAU0))**2*(TAU0+2.)/2.)
 C1=3.6*ATW/Z**0.9882/C1
 C2=1.191E-3*Z**0.8622
 C3=1.02501-1.0803E-4*Z
 C4=0.99628-1.303E-4*Z
 C5=1.02441-1.2986E-4*Z
 C6=1.03/Z**1.11E-2
 W=LOG(1.D0+C2*DBLE(TAU0**C3))/C2
 W=W-DBLE(C4*TAU0**C5)/(1.D0+DBLE(C6*TAU0))
 R0=C1*SNGL(W)
 RETURN
END FUNCTION
```

B.3 MCNPX Input File

```
 ISIS BBGS mcnpx electron transport - REB - 110212
c MODEL B - 2.1 g/cm3
c cell specification ——————————
c
c # mt   density     geometry      args         notes
   1  1   -0.0013     -1 2          imp:e=1      $ bgd air cube
   2  2   -7.86       -2 3 4        imp:e=1      $ steel assm structure
   3  1   -0.0013     -3            imp:e=1      $ air inside assm
   4  3   -2.10       -4            imp:e=1      $ carbon target region
   5  0               1             imp:e=0      $ outside
 666   0              -666          imp:e=1      $ cookie cutter cell for beam
c
c end cell spec ——————————

c surface specification ——————————
c
c #   crd   constants                                      notes
   1  box   -20 -20 -10   40 0 0    0 60.0 0   0 0 30   $ outer bounding box (bb)
   2  box   -6    0   0   12 0 0    0 22.6 0   0 0 12   $ conv assm bb
   3  box   -2.5 11.3 0    5  0 0   0 5.0  0   0 0 12   $ air inside assm bb
   4  box   -2.5 6.3  0    5  0 0   0 5.0  0   0 0 12   $ region for carbon bb
 666  rec    0    0  -0.1  0 0 0.2 0.5 0 0   0 0.5 0   $ cookie cutter surf for beam
c
c end surface spec ——————————

c material specification ——————————
c
c MT 1 is AIR, T=20C
c mt   code          density         keywords      element
 m1    8000.01e      0.2             gas=1         $ O
       7000.01e      0.8                           $ N
c
c MT 2 is 1020 Steel, T=100C , all natural elements (AAA000)
c mt   code          wgt fraction (percent/100)   element
 m2    6000.01e          -0.0020                        $ C
       25000.01e         -0.0045                        $ Mn
       15000.01e         -0.0004                        $ P
       16000.01e         -0.0005                        $ S
       26000.01e         -0.9926                        $ Fe
c
c MT 3 is Nuclear Graphite, high density, T=300C
c mt   code          density
 m3    6000.01e          1.0
```

```
c
c end material spec ———————————
c
c tally card specification ——————
c
c specifier     notes
 f06:e 4        $f6 is for energy, 0 denotes tally 0 (name), 4 is cell to tally
 sd6   1        $sets volume of cell to value for tally
c
c mesh tally description                $ comments are not allowed in this card!
tmesh
rmesh3
cora3 -2.5 20i 2.5
corb3 6.3 20i 11.3
corc3 0 20i 12
endmd
c end tally card spec ———————————
c
c other data and solution info ————
c
 mode e          $ Mode e: electron transport only
 sdef  dir=1 vec=0 0 1 x=d1 y=d2 z=0 ccc=666  tr=1 erg=60.67 $ source def
 sp1   -41 0.200 0              $ 2.35482 * 1 (a) a=0.0851
 sp2   -41 0.200 0              $ 2.35482 * 1 (b) b=0.0213
 tr1   0 8.8 -2.1   1 0 0   0 1 0   0 0 1   1
 nps   1000000   $ Number of particles to run
c
c end other data, EOF ——————————
c
print
prdmp 0 0 1
```

B.4 Matrix Solver for BBGS Coolant Flow

```
## BBGS COOLING ----------------------------------------------------------
## Matrix solver for pressure drop
## Author: R.E. Block, reblock@alum.mit.edu
## Created: 2/1/11
## Modified: 3/3/11
## Filename: bbgscooling.py
## ----------------------------------------------------------------------

## Imports ---------------------------------------------------------------
from numpy import *
from matplotlib.pyplot import *
## ----------------------------------------------------------------------

## User Inputs -----------------------------------------------------------
NOINLET = False

# DEFINE CONSTANTS FOR EACH SEGMENT OF TUBING
d1 = 0.00315   # This is for copper tube
d2 = 0.0114    # This is for inner diameter of 1/2 flex tubing (0.45 in)

if NOINLET:
    l1=0.001
    l2=0.001
    l2=0.001
else:
    l1 = 8    #m of tubing for inlet and return
    l2 = 1.5 #m of tubing between dipole and y
    l3 = 0.5 #m of tubing between y and x

ldipole = 56.0/3 # 3 loops per coil, 56 m per coil
ly = 13.2 # thats one loop for BOTH coils
lx = 19.6 # one loop per coil, 2 loops total
lconv = 2.64  # assuming plates with one line for every cm of area on two sides

id = 187
iy = 200
ix = 200

rho = 1e3   # Density of water at 20 C
mu = 1e-3   # Viscosity of water at 20 C
cp = 4.18   # kJ/kg K

resistivity = 1.77E-6
```

```
area = 0.635**2-pi*(0.315/2)**2
rperl = resistivity/area

# BUILD REAL MODEL - USE 17 SEGMENTS
n = 17
nloops=11

# PRESSURE DROP
dP = 1.38e5 # pressure in Pa [1.38e5 is 20 psi]

L = array([l1 ,ldipole ,ldipole ,ldipole ,ldipole ,ldipole ,ldipole ,l2 ,ly ,l3 ,lx ,lx ,lconv ,l3
    ,l2 ,l1 ,lconv])
if len(L) != n:
    print "Fatal_error ,_incorrect_length_of_variable_L"
    quit()
D = array([d2,d1,d1,d1,d1,d1,d1,d2,d1,d2,d1,d1,d1,d2,d2,d2,d1])
R = L*rperl*100

current = array([0 ,id ,id ,id ,id ,id ,id ,0 ,iy ,0 ,ix ,ix ,0 ,0 ,0 ,0 ,0])

power = current**2*R/1000 # in kW
power[12] = 4.5/2 # deliver 4.5 kW to converter assembly!
power[16] = 4.5/2
# CREATE CONSTANT TERM FOR MULTIPLICATION IN LOOPS
k = 0.184*8/(4**0.2*pi**1.8)*rho**(0.8)*L*mu**0.2/(D**4.8)

# TO ACCOUNT FOR PRESSURE DROP OF TURNS IN MAGNET, ADD K2
ntd = 4*7*2
nty = 4*4*7*2
ntx = 4*4*12
ntconv = 2*22*2 # accounting for some coiled geometry on face of converter
k2factor = 0.5 #average resistance factor for 90 degree turns

k2=k2factor*rho/(2*(pi*(d1/2)**2)**2)*array([0 ,ntd ,ntd ,ntd ,ntd ,ntd ,ntd ,0 ,nty ,0 ,ntx ,
    ntx ,ntconv ,0 ,0 ,0 ,ntconv])
##------------------------------------------------------------------------

## Create matrices and solve ---------------------------------------------
b = zeros(n)
b[-nloops:] = ones(nloops)*dP

beta = 0.45

A = zeros((n,n))
posind = [[0],[7],[9],[10 ,11 ,12 ,16],[8 ,13],[1 ,2 ,3 ,4 ,5 ,6 ,14]]
```

```
negind = [[1,2,3,4,5,6,7],[8,9],[10,11,12,16],[13],[14],[15]]

for i in range(0,len(posind)):
    for j in range(0,len(posind[i])):
        A[i][posind[i][j]] = 1
    for j in range(0,len(negind[i])):
        A[i][negind[i][j]] = -1

Q1 = ones(n)

# CREATE MATRIX OF ONES DESCRIBING CLOSED LOOPS IN THE SYSTEM
I = zeros((nloops,n))
loops=[[0,1,15],[0,2,15],[0,3,15],[0,4,15],[0,5,15],[0,6,15],
    [0,7,8,14,15],[0,7,9,10,13,14,15],[0,7,9,11,13,14,15],
    [0,7,9,12,13,14,15],[0,7,9,16,13,14,15]]
for i in range(0,len(loops)):
    for j in range(0,len(loops[i])):
        I[i][loops[i][j]]=1
#f=open('output.txt','w')
#A[-nloops:]=I*9
#print >> f,A
#f.close()

#print I
error = 100
numit = 0

deltaP = arange(5,105,5)*1.38e5/20.0
# deltaP = array([20])*1.38e5/20.0
Qs = zeros(len(deltaP))
for i in range(0,len(deltaP)):
    b[-nloops:] = ones(nloops)*deltaP[i]
    while True:
        # SET NEW VALUE OF A FOR PRESSURE DROP TERMS:
        A[-nloops:]=I*k*Q1**0.8
        # INCLUDE VALUES FOR TURNS
        A[-nloops:]=A[-nloops:]+I*k2*Q1
        Q2 = linalg.solve(A,b)
        #Q2 = dot(inv(A),b)
        error = sum(abs(Q2-Q1))/sum(abs(Q2))
        if error < 1e-4:
            print 'Problem converged in '+str(numit+1)+' iterations'
            # print A
            print 'Q[m^3/s] ='
            print Q2
```

```
                break
            numit = numit + 1
            if numit >=100:
                print 'Not␣converged'
                print 'Q[m/s]␣='
                print Q2
                break
            Q1 = abs(Q2)**(beta)*abs(Q1)**(1-beta)
        Qs[i] = Q2[0]*15850.0

deltaP = deltaP*20/1.38e5

# CONVERT FINAL Q TO GPM
Qgpm = Q2*15850.0
print 'Q[gpm]␣=␣'
print Qgpm

dT = power/(rho*cp*Q2)
print 'dT␣=␣'
print dT

Re = 4*rho*Q2/(pi*mu*D)
print 'Re␣=␣'
print Re

dPf = k*Q2**1.8 + k2*Q2**2
dPf = dPf*20.0/1.38e5
print 'dP␣=␣'
print dPf
print dPf[0]+dPf[-1]+dPf[1]

print 'power␣=␣'
print power

figure()
plot(deltaP,Qs)
xlabel('System␣Pressure␣Drop␣[psi]')
ylabel('Total␣Flow␣Rate␣[gpm]')
#title('System curve, excluding converter plate cooling')
show()
## ---------------------------------------------------------------------
```

Bibliography

T.L. Albers. High-temperature properties of nuclear graphite. *Journal of Engineering for Gas Turbines and Power*, 131, November 2009.

T.A. Antaya and Z.Q. Xie. The BEAM3D Code-For the Extraction of Multiply-Charged Ions from Ion Sources. Technical report, Michigan State University, 1987. NSCL Report MSUCP-63.

Jordi Reig Armero. Design of an ECR Proton Source. Master's thesis in Industrial Engineering. School of Industrial Engineering of Barcelona, Universitat Politecnica de Catalunya, July 6, 2010.

B. Blackburn et al. Utilization of actively-induced, prompt radiation emission for nonproliferation applications. Technical report, Idaho National Laboratory, August 2006. INL/CON-06011167.

B. Blackburn et al. Detection of special nuclear material by means of promptly emitted radiation following photonuclear stimulation. Nuclear Science Symposium, 2007. N10-7.

R. Benaroya and W.J. Ramler. Deflection coil for an external accelerator beam. *Nuclear Instruments and Methods*, 10:113–120, 1961.

Matthew Bunn and Anthony Wier. Terrorist nuclear weapon construction: How difficult? *ANNALS, AAPSS*, 607, September 2006.

Opera-3d User Guide. Cobham Technical Services, 2009. VF-07-09-D2.

Don Daigler. DHS strategy for improving the national response and recovery from an IND attack. Technical report, US Department of Homeland Security, March 2010.

R. Gupta and T.D. Prasad. Extended use of Linear Graph Theory for analysis of pipe networks. *Journal of Hydraulic Engineering*, 126(1):56–62, 2000.

Siegfried S. Hecker. Toward a comprehensive safeguards system: Keeping fissile materials out of terrorists' hands. *ANNALS, AAPSS*, 607, September 2006.

Stanley Humphries. *Charged Particle Beams.* John Wiley and Sons, New York, 1990.

Stanley Humphries. *Charged Particle Acceleration.* John Wiley and Sons, New York, 1999.

James L Jones et al. Status of prototype Pulsed Photonuclear Assessment (PPA) inspection system. *Nuclear Instruments and Methods in Physics Research*, 579: 353–356, 2007.

Bonnie Jenkins. Combating nuclear terrorism: Addressing nonstate actor motivations. *ANNALS, AAPSS*, 607, September 2006.

David Kramer. Detectors could miss bomb-grade uranium at ports, group warns. *Physics Today*, May 2008.

John J Livingood. *The Optics of Dipole Magnets.* Academic Press, New York, 1969.

Poisson-Superfish Manual. Los Alamos National Laboratory, 2010. URL `FTP://SFUSER:ftpsuperfish@LAAGG1.LANL.GOV/`.

A.I. Lutcov, V.I. Volga, and B.K. Dymov. Thermal conductivity, electric resistivity and specific heat of dense graphites. *Carbon*, 8:753–760, 1970.

Fr. Thomas McGahee. American Wire Gauge Table for Bare Copper Wire, April 1998. URL `http://www.power-dc.com/losses.html`.

NIST. E-star: Stopping-power and range data for electrons, 2010. URL `http://physics.nist.gov/`.

Martin Reiser. *Theory and Design of Charged Particle Beams.* Wiley-Interscience, New York, 1994.

V.Z. Shemlet, A.P. Pomytkin, and V.S. Neshpor. High-temperature oxidation behaviour of carbon materials in air. *Carbon*, 31(1):1–6, 1994.

TRANSPORT: a computer program for designing charged particle beam transport systems. Stanford Linear Accelerator Center, 1972. Ref SLAC-91.

T.A. Antaya. Technical Topical Report for the Bremsstrahlung Beam Generation System. Technical report, Massachusetts Institute of Technology, March 2010. Data Item A003 - Rev 1.

T. Tabata, P. Andreo, K. Shinoda, and R. Ito. Energy deposition through radiative processes in absorbers irradiated by electron beams. *Nuclear Instruments and Methods in Physics Research B*, 93:447–456, 1994.

T. Tabata, P. Andreo, and K. Shinoda. An analytic formula for the extrapolated range of electrons in condensed materials. *Nuclear Instruments and Methods in Physics Research B*, 119:463–470, 1996.

Jack T. Tanabe. *Iron dominated electromagnets: design, fabrication, assembly and measurements.* World Scientific, 2005.

Neil E Todreas and Mujid S Kazimi. *Nuclear Systems I: Thermal Hydraulic Fundamentals.* Taylor and Francis, New York, 1990.

Zu Qi Xie. *The effect of space charge force on beams extracted from ECR ion sources.* PhD thesis, Michigan State University, 1989.