M9 Simulations: Summary of Work - Summer 2018

Dylan Bassi

August 22, 2018

Abstract

This is just a quick document outlining the work done over the Summer of 2018 by Dylan Bassi (University of Waterloo) on the M9 redesign. Over this time period the M9A beamline was successfully modeled in both TRANSOPTR and G4Beamline. The goal of the simulations was to acquire tunes for the beamline, while also validating various physical aspects of the beam. A tune for the beamline was successfully determined in both models. The beam envelope at the focus of the TRANSOPTR tune is: 1.96 cm in x, 1.95 cm in y. The G4BL tune results in 73% of particles surviving. 100% of these particles falls within a 2" diameter area, 80% within a 1" diameter area and 25% fall within a 1 cm diameter area. In G4BL it was determined that the Wien Filter adequately rotates the spin of muons coming down the beamline while also effectively removing 99.99% of positrons produced by muon decay for fields as low as 0.012 T. It was also determined that a minimum kicker strength of 17.9 kV is sufficient to provide muons-on-request. Furthermore, it was shown that both the M9H and M9A leg cannot be running simultaneously, as the solenoid in the M9H leg causes an extreme vertical deflection in the M9A leg. Additionally, several limitations of the models were investigated and attempts to overcome them were made. Finally the M9Q2 quadrupole was modeled in Opera to provide simulations of asymmetrically driven quadrupoles. These models will allow researchers to correct physical displacement in the M9B1 bending dipole by contributing a dipolar component in the field produced by the quadrupoles. For a directory map see: musim1:~dbassi/Summer2018/documents/Overview/directory_map.pdf

Contents

1	Introduction	5					
2	TRANSOPTR 7						
	2.1 XML2OPTR	7					
	2.2 Beam Envelope (HLA)	8					
	2.3 The Tune	8					
	2.4 Limitations	11					
3	G4Beamline	11					
	3.1 Tuning Bending Magnets	12					
	3.2 The Profile Command	14					
	3.3 GMINUIT	14					
	3.4 XML2G4BL	16					
	3.5 Wien Filter Strength	16					
	3.6 Kicker Strength	17					
	3.7 Solenoid Induced Vertical Deflection	19					
	3.8 The Tune	19					
	3.9 Limitations	23					
4	Opera	25					
	4.1 Opera2BLFieldMap	25					
	4.2 M9Q2	26					
	4.3 Limitations	28					
5	o Conclusion						
\mathbf{A}	G4BL Build Instructions for RHEL7 derived OS	31					

List of Figures

1	Beamline Layout of the M9A Beamline	5
2	Beamline Layout of the M9A Beamline	6
3	M9Q2 Opera Model	8
4	TRANSOPTR Simulation of the OPTR Calculated Tune	9
5	TRANSOPTR Simulation of the G4BL Calculated Tune	10
6	G4BL Viewer Image of M9A Simulation	13
7	Spin Polarization of Muons	13
8	G4BL Simulation of the G4BL Calculated Tune	14
9	GMINUIT Interface	15
10	Spin Polarization of Muons	17
11	Wien Filter as a Positron Separator	18
12	Kicker Ability to Provide MOR	18
13	M9A Leg Deflection from M9H Solenoid	19
14	G4BL Simulation of the OPTR Calculated Tune	20
15	G4BL Simulation of the G4BL Calculated Tune	21
16	Close Up of Lost Particles Exiting M9AB2	22
17	Final Focus of M9A in G4BL	22
18	Stopping Quadrupole Leakage with Fitting	24
19	Orientation of T2 in G4BL	25
20	M9Q2 Opera Model	26
21	Comparison of Opera3D Generated Field and G4BL Field for M9Q2	27

List of Tables

1	Comparison of Initial Tune and TRANSOPTR Calculated Tune	10
2	Comparison of the G4BL 'Profile' Tune and 'NumPy' Calculated Tunes	16

3 Comparison of the TRANSOPTR and GMINUIT Calculated Tunes 20

1 Introduction

One of the secondary beamlines in the TRIUMF Meson Hall is M9. Fig. 1 displays the overall layout of the beamlines coming off of 1AT2, hereforth referred to only as T2. The black, or lowest, path is for M20, while the red and blue paths correspond to the two legs of M9. M9A is the red, or middle, path while the blue/top path is for M9H. M9 is a beamline for muon spin resonance (μ SR) experiments that is split into two legs: M9A and M9H. The M9A leg is intended for use with surface muons, which are muons that are the decay product of stationary pions in the target T2. The dipoles and filters are set to filter exclusively for these surface muons. Surface muons are important for experiments because they have a relatively low penetrative depth, and allow for unique properties to be observed near the surface of various materials. The M9H leg works a little bit differently. M9H is not concerned with surface muons. Instead, the pions emitted from the target that have enough momentum to leave the target material, are bent through the M9B1 dipole based on their momenta. The strength of the M9B1 field will determine what momenta to accept into the M9H leg. These pions will the enter into a long solenoid where they will decay into muons which will finally be sent down the beamline.



Figure 1 – This layout shows an overview of the beamlines coming off the T2 target. The black (bottom) line is M20, the middle (red) line is M9A, and the top (blue) line if M9H.

Throughout this term, three simulation programs were used to various effect. These three programs were TRANSOPTR, G4beamline and Opera3D. TRANSOPTR is a simulation

software that uses the sigma matrix formalism to solve the equations of motion for the second moments of the beamline in its 6-dimensional phase-space. TRANSOPTR allows the user to set various fit constraints throughout the beamline. These fit constraints allow the user to set values for various elements in either the transfer or the sigma matrix. TRANSOPTR was the first simulation software used this term to model the beamline. This was followed by using G4BL to simulate the beamline. G4BL is based off of Geant4. This means that G4BL employs a Monte Carlo method to statistically simulate the beam, running many iterations of individual particles with random initial starting conditions. This contrasts to the matrix formalism used by TRANSOPTR, since TRANSOPTR only tracks the beam envelope.



Figure 2 – This layout shows an overview of the components in the M9A beamline. Although this was not used to build the simulations, it provides the clearest overview of all components necessary to simulate. Note in this blueprint that the final three quads are mislabelled and should be AQ11, AQ12 and AQ13 respectively.

As previously mentioned the M9A leg was intended for use with surface muons. The goal for the M9A leg is to provide surface muons for unique chemical physics investigations into various materials. Almost all of the work accomplished this term was on simulating the M9A leg. An overview of the M9A layout is shown in Fig. 2. It is important to notice that the final three quadrupoles, labeled AQ11, AQ10 and AQ9 respectively are mislabeled. The actual M9A beamline has AQ11, AQ12, AQ13 respectively instead. The work was accomplished with the goal of trying to achieve a realistic tune for the beamline. Considering that there were inherent limitations to each software, it was a worthy venture to obtains tunes using more than one method. This will make it so that if a tune is off, there is at least another model that could try to be implemented.

One of the issues that was required by the M9 redesign project is the realignment of the beamline that resulted from swelling and contraction of the poured-in-place concrete underneath M9B1[?]. It was hoped that it would be possible to asymmetrically supply current to the poles of M9Q2 to add deflection in the beamline. The rationale being that if

future displacements in the beamline occur, it would be ideal to correct the misalignment solely by changing the power supply. Opera3D was used to model M9Q2 and produce a magnetic field map which could be implemented in G4BL. Unfortunately due to time constraints it was not possible to determine whether a model could be produced within G4BL to allow for quickly determining a tune to realign the beamline. It is hoped that with the M9Q2 model built though, it will be possible to determine an efficient model that is practical for use with GMINUIT.

2 TRANSOPTR

The first simulation software that was used to tune M9 was TRANSOPTR. TRANSOPTR is a program written in FORTRAN from 1981, and maintained in-house at TRIUMF. TRAN-SOPTR uses the matrix formalism of beamline optics to allow for the optimization of an n-dimensional parameter space given user constraints [1]. For a description of the sigma matrix formalism, refer to Charged Particle Optics by Brown[2], the TRANSPORT manual[3] or TRANSOPTR manual[1]. This allows the beamline to be quickly simulated since the elements are represented as matrices, rather than complex geometrical objects, and thus the simulation is just a series of matrix operations, which are handled exceptionally quickly in FORTRAN, especially with modern computers. In contrast to this approach are Monte Carlo, or raytracing methods, which require vast amount of computing power to properly track individual particles of the beam over many simulations. G4BL is an example of a Monte Carlo program that can be used and will be examined in Section 3.

2.1 XML2OPTR

To help streamline the process of simulating and maintaining beamlines, TRIUMF has been developing an in-house suite of tools known as high-level applications (HLA). One of the goals of these applications is to be implemented in a way that is easy for novice programmers to understand. As such XML2OPTR was created to allow for researchers to create an XML file containing all of the necessary beamline information and quickly convert this to input to be run for TRANSOPTR. This is beneficial since the XML file format uses a simple structure to map abstract information, requiring no real coding experience. The structure of an XML file can easily be learned by someone with no programming experience in a day. All that is required is an understanding of which tags will be required for the XML2OPTR code. At present, no document type definition exists and so this can only be done by cross-referencing the already completed examples. The eventual goal of HLA is to connect the application to an established database with the beamline specifications. This would allow users to run this application through a browser merely by selecting the interested beamline, requiring no coding whatsoever. An example of this functionality is described in Section 2.2.

At present, getting access to XML2OPTR requires joining the TRIUMF gitlab at https: //gitlab.triumf.ca and requesting permission to join the HLA/ACC repository. Once the repository is cloned locally, the XML2OPTR script can be run through a Python 3 interpreter, passing the name of the beamline to generate input for in the accelerator database as an argument. It is inadvisable to change the Python installed directly to the operating system. Instead one should install Python 3 locally and ideally run it through a virtual environment, to ensure that the packages used are always the correct version.



Figure 3 – M9A TRANSOPTR envelope as seen in the Beam Envelope web application. This application allows the beamline to be tuned with a graphical interface through the TRIUMF HLA web portal, requiring no knowledge of programming.

2.2 Beam Envelope (HLA)

Another example of an HLA is the Beam Envelope application. Beam Envelope uses XML2OPTR to run TRANSOPTR interactively in a web browser based on the specifications in the accelerator XML database. This allows researchers to add fit constraints to the XML database and tune the beam, requiring next to no programming knowledge. Beam Envelope can be accessed (at present) by going to https://devel.hla.triumf.ca/beam/envelope/. Fig. 3 shows the Beam Envelope web interface for M9A, with the Tune settings window open, illustrating the ability to tune the beam online. In the future however, the application will be put onto a Kubernetes cluster and run through it's own link on the TRIUMF intranet.

2.3 The Tune

Tuning a beam in TRANSOPTR requires one to know the initial phase space shape as well as a reasonable guess as to the strength of the field applied to the quadrupoles. The user must also set their constraints to which TRANSOPTR will run a chi-squared minimization on. The user is capable of constraining the beam envelope by applying "fit commands" at various lengths along the beamline. For this simulation several fit constraints were used. The goal of these constraints was to maximize the luminosity at a focal region. This was accomplished by focussing the beam into as tight a region as possible at the target, while also trying to limit the beam envelope to being within the diameter of the beampipe and beamline elements through the entirety of the line. The aim of this was to maximize the number of particles reaching the focus, while also minimizing the final focal area. As illustrated in Fig. 4 this is not easy to achieve and some of the beam envelope gets clipped by the beamline elements regardless of the fit constraints. This illustrates a problem that will be revisited in Section 3.8, namely, that given the initial phase space chosen, it does not appear possible to get the entirety of the beam to the end. This implies that constraining the initial phase space more strictly might allow for more precise simulations of beamline.

The initial phase space parameters were determined by a couple different methods. First, the x and y widths of the beam were determined from the size of the T2-Target[4]. The initial divergence was determined empirically within the simulation software by drifing the beam to end of the second quad with no field strength applied. It was assumed that the value that results in the beam spreading to the diameter of the beam pipe is a good approx-



Figure 4 – Beam envelope produced by TRANSOPTR simulation using the TRANSOPTR calculated tune.

imation of an upper limit as to what could ever possibly reach the end. This assumption was considered valid as the addition of the quadrupole field will result in at least one direction becoming wider, which would result in the beam hitting the beampipe and decaying. Finally, the momentum spread was determined based on a measurement of surface muons produced in the M15 beamline by Donald Arseneau[5]. The initial tune values were obtained from TRI-DN-06-17[6]. Since the publication of TRI-DN-06-17[6], there have been several design changes to M9, such as the replacement of M9AQ6, M9AQ7 and M9AB2 in the M9A_request_form_585[7]. Therefore it is no surprise to observe that the optimized tune from Table 1 is greatly different from the initial tune.

Since two simulation software were used to model the beamline and obtain tunes, I felt it was appropriate to see what the result of the tunes obtained in each program would be if they were simulated in the opposing software. Fig. 5 shows the profile obtained by using the tune obtained from G4BL in Section 3.8. The disagreement between the horizontal beam width is very drastic, however the beam envelope appears to first grow outside the radius of the beampipe coming out of M9AB2 as seen in Section 3.8. Although the growth is more drastic than observed in the G4BL tune, it would be interesting if it were possible to see how much more closely the rest of the beam envelope would agree if we could account for the loss of particles here. If possible, it would be beneficial for someone to decrease the emittance and see if the profiles would look more similar. It is important to note that it is not only the horizontal beam width that is in disagreement. Although the general shape of the vertical beam envelope is similar between simulations the final focus is much larger, indicating that there is still discrepancy between the two simulations.

Element	Polarity	Initial Tune (A)	TRANSOPTR Tune (A)	% Change
M9Q1	-	136.05	114.0	17.6
M9Q2	+	58.62	57.17	2.50
M9AQ3	+	47.89	56.77	17.0
M9AQ4	-	75.00	74.84	0.21
M9AQ5	+	34.93	25.71	30.4
M9AQ6	+	25.55	30.19	16.6
M9AQ7	-	31.05	35.96	14.7
M9AQ8	-	49.38	48.39	2.03
M9AQ9	+	69.10	56.50	20.1
M9AQ10	-	49.38	15.54	104
M9AQ11	+	11.29	38.73	110
M9AQ12	-	60.98	69.29	12.8
M9AQ13	+	74.18	58.10	24.3

Table 1 – This table compares the initial tune used to start the TRANSOPTR simulation and the tune that the TRANSOPTR simulation determined.



Figure 5 – Beam envelope produced by TRANSOPTR simulation using the GMINUIT calculated tune. The tunes disagree strongly after the second group of quadrupoles. The increased envelope size by the second bending magnet is consistent with the loss of particles observed in G4BL. In particular, the regions near the separators disagree strongly.

2.4 Limitations

TRANSOPTR is very good at what it does, but it is still limited in function by its design. One such limitation is due to the fact that TRANSOPTR tracks the envelope of the beam and not individual particles. This makes any investigation into the misalignment of the beam impossible since the simulation tracks the beam width in relation to the optical axis. Another limitation of TRANSOPTR is that one can only get an idea of how optical design constraints affect the beamline. That means that if there is a physical restriction that does not influence the optics, it is impossible to investigate it. One example of this is the beam pipes. Due to the large initial acceptance for the beam phase space it proved to be very difficult to try to fit the beam through all elements within diameter of the beam pipe. In Fig. 4 several portions of the beam envelope can be seen to exceed the beam pipe radius. In reality this would affect the rest of the beam shape downstream, and thus how the profile would change. Since this is not taken into consideration, characteristics of the actual beam envelope downstream will be different than the modeled results.

One of the other issues I noticed when contrasting this optimization to the GMINUIT optimization in Section 3.3 is how difficult it was to get the fitting algorithm to squeeze through all elements. In TRANSOPTR several elements would exceed their fit constraints by a little bit, since there were so many. Ultimately TRANSOPTR seems to think that a little bit of loss everywhere provides the best tune, since it provides the lowest chi-squared value. If the density of particles is shifted to these edges this may not be ideal. GMINUIT allows the user to define the number of surviving particles as a parameter in the chi-squared minimization reducing the number of constraints from the number of elements to one. This gave GMINUIT the flexiblity to decide that losing a significant number of particles in one region is better than losing a few in many places, as observed in Fig. 16 from Section 3.8. This is preferred since it gives a more direct understanding of the effect on the luminosity.

The limitations for TRANSOPTR come not from any error in the program, but rather the inherent limitation of the model for the beamline that is used. Namely, the sigma matrix formalism only handles optical elements and the second moments. To treat these limited cases, a different approach to modeling is needed which tracks what happens to individual particles, rather than the entirety of the beam. With that being said, TRANSOPTR does have a particle tracking mode. Unfortunately, this mode does not work at present and so extra software is required at present [8].

3 G4Beamline

So far the matrix formalism of optics has been used to simulate and tune the beam envelope with user constraints[1]. Contrasting to this is using a Monte Carlo simulation, which tracks individual particles in the beam to get a statistical representation of the beam envelope. This allows understanding of how physical constraints impact the beamline rather than being constrained to optical elements only. One of the most widely used software toolkits for building particle physics Monte Carlo simulations is Geant4[9]. The biggest hurdle to running Geant4 simulations is the amount of coding that has to be done. To successfully build a simulation in Geant4 requires learning the highly convoluted language employed by Geant4 and then building a C++ program[10]. The problem with this is that C++ programs are inherently complex and requires a lot of code to build simple simulations. Simulating entire beamlines properly is generally beyond the ability of even a skilled researcher in a small period of time. A description of the physics employed in G4BL will not be included in this report, but can be referenced in [9].

Out of the need to be able to simulate large beamlines in Geant4 quickly, Muons Inc. has created the G4beamline (G4BL) program[10]. The G4BL program removes the need to generate all the C++ files for every element and instead employs a custom shell-like scripting language for creating input files. These input files finally allow researchers to build beamlines with little coding. As an example of the differences in the length of code, my longest G4BL input file was around 540 lines long many of which were redundant to allow me to quickly test various setups. In contrast, the C++ code required for genericbends, just one G4BL beamline element, requires nearly 600 lines. In addition to the length of input codes, the simple, functional scripting language is much easier to pickup for new programmers than the abstract object-oriented programming required by C++. This makes G4BL something that can reasonably be learned by a temporary, or time constrained researcher.

To obtain a copy of G4BL, you must go to the Muons Inc. website and request permission. When I tried to, there was an error in the script for signing up, so you may have to email Tom Roberts directly for access. There are pre-built installations as well as the source code available. For inexperienced linux users, it may be preferred not to build from source, as it requires linking several libraries and working with CMake. However, building the code from source yourself is the best way to install G4BL. This is for a couple of reasons. First, it ensures that the installation will be compiled specifically with your libraries, limiting the chances of various bugs. One can simply put all the installed libraries into an unused folder, as G4BL will use what is present on your system. If any library is missing from your operating system, one can simply move it back out of the unused folder.

The second, and most important benefit, is that builing the code from source means that you can link mpi with it to allow parallel computing of the simulations. Since the pre-built versions by default only use one processing thread and hyper-threaded quad core processors are being coming exceptionally common, this can provide a huge boost in computation time. On musim1, this allows eight threads to be run simultaneously, on musim0 it allows for 12 threads. In a cluster environment this could provide incredibly detailed simulation results in a relatively short period of time. Build instructions for G4BL from source on a RHEL7 computer is provided in Appendix A, courtesy of Donald Arseneau at TRIUMF.

Muons Inc. also provides a program, GMINUIT, for allowing users to generate scripts for running G4BL recursively, while trying to optimize a parameter space. This software provides the same tuning functionality to G4BL that is present in TRANSOPTR. The difference in the tuning algorithm is inherent to the model, namely the fact that now we are modeling individual particles in a Monte Carlo manner. This gives us a different of the optimization however. In TRANSOPTR we had to assume that if we could fit the beam through the beam pipe that particles wouldn't be lost. Now we are fitting a statistical representation of the beam, and can gain valuable information regarding the survivability of particles, and how a loss of particles in a region impacts the profile elsewhere.

3.1 Tuning Bending Magnets

When using centerline coordinates, the corner command needs to be used at the bending magnet positions to let G4BL know that the centerline coordinate system has rotated. This presents a bit of an issue as particles are expected to traverse around a smooth arc rather than two piece-wise line segments. This results in the magnet being out of relative position to the rest of the beamline, as the corner effectively shifts where the physical arc passes through the bending magnet and results in the simulation being misaligned.

This unfortunately results in the field strength being slightly off, as it was calculated with the assumption that the the centerline and arc are the same. G4BL has a tune command



Figure 6 – This image shows the view as seen from the G4BLGUI viewer. This image illustrates the layout of the M9A beamline used in the G4BL simulations. This beamline includes thirteen magnetic quadrupoles, two magnetic dipoles, two Wien filters, two sets of slits, a kicker and beam pipe in between practical elements.



(a) Without correcting for the corner displacement of the centerline, the beam will not properly traverse through the dipole, resulting in a misaligned simulation.

(b) If the bending magnet placement is shifted, or the downstream centerline positions are shifted, the beam should be properly realigned. This shifting required a slight change in the field strength, as it was calculated assuming the arc and the centerline are the same.

Figure 7 – The corner command must be used when placing elements on centerline coordinates. This command unfortunately does not model the bend as a continuous arc, but rather two discrete line segments.



Figure 8 – Comparison of the sigma profiles calculated by NumPy and G4BL's profile command for the GMINUIT tune. Errors in the way that G4BL calculates the profile leads to significant errors near the focus. As a result the tune from GMINUIT had to be recalculated using the NumPy profile.

which allows just this functionality. The user inputs constraints on their beamline, in this case that the beam is aligned coming out of the magnet, and an initial value. G4BL then tries to find a field strength that satisfies the user constraints. The effect of the corner misalignment and retuning is shown in Fig. 7. The initial prescribed strength was 0.1456 T. G4BL found that 0.1452 T preserves the proper beam path.

3.2 The Profile Command

The G4BL-3.04 release notes indicate that "[t]he 'profile' command has numerous problems in the way it computes values; the only workaround is to use a 'zntuple' and compute the values you want"[11]. Initially I was unaware of this as I did not have access to the release notes when I first started working with G4BL. Therefore it was necessary to compare the results of a calculated profile with that produced by G4BL's profile command. To calculate the profile from zntuples a loop was used to output a zntuple every 100 mm. Each zntuple was then put into a NumPy array and the σ_x and σ_y values were calculated, as these are what profile claims to represent. The results of this comparison can be seen in Fig. 8. From this plot it is apparent that the profile command does not accurately represent the focus region and should therefore not be used in GMINUIT.

3.3 GMINUIT

Unlike TRANSOPTR, G4BL has no native ability to tune the beam parameters. In response to this, Muons Inc. has packaged GMINUIT with G4BL. GMINUIT is a graphical interface



Figure 9 – An image of what the GMINUIT interface looks like. GMINUIT can be used to tune parameters of a G4BL input file, while showing how the changing tune affects the beam envelope.

to the MINUIT optimization engine and a user defined script. In essence, this allows a user to define a GMINUIT script in the language of their choice (shell, perl, tcl, etc.) to run G4BL while varying parameters[12]. To do this, the user creates a script that completes several tasks:

- 1. Define parameters to vary and to what extent to vary them.
- 2. (Optional) Define gnuplot file to show restrictions imposed by beam elements (such as beam pipe).
- 3. Create, or run a pre-made G4BL input file with defined parameters.
- 4. From the data output by G4BL, determine a χ^2 value based on user constraints.
- 5. Vary the parameters and repeat the process until the χ^2 does not improve within tolerance.

Initially the profile command in G4BL was used to output the data that was used to determine the χ^2 value, but as shown in Section 3.2, the profile command does not accurately represent the beam at the focus. As such, the data was written into a zntuple file every 100 mm and used to determine a σ profile for the beam. This did not drastically change the tune, but the tune did change slightly, as shown in Table 2. An example display of the GMINUIT interface is given in Fig. 9. It is worth noting that although GMINUIT was developed by Muons Inc. to be used in conjunction with G4BL, it can be used with any general parameter space so long as the user can generate a script to emulate the process described above. GMINUIT has several optimization algorithms that can be used. Based on my experience I found that using the Nelder-Mead (downhill simplex) algorithm achieved results much more quickly and with a better ability to move coarsely around the parameter space before fine-tuning its result. This may not always be true, but allowed for the best results for this simulation.

Element Polarity		'Profile' Tune (A)	'NumPy' Tune (A)	% Change
M9Q1	-	113.8	114.9	0.96
M9Q2	+	53.98	54.20	0.41
M9AQ3	+	50.57	50.52	0.10
M9AQ4	-	66.87	66.88	0.01
M9AQ5	+	25.64	25.78	0.54
M9AQ6	+	22.23	22.35	0.54
M9AQ7	-	25.61	25.74	0.51
M9AQ8	-	35.22	35.26	0.11
M9AQ9	+	51.33	51.30	0.06
M9AQ10	-	31.09	31.02	0.23
M9AQ11	+	25.02	25.59	2.25
M9AQ12	-	60.30	60.77	0.78
M9AQ13	+	64.70	65.10	0.62

Table 2 – Comparison of the tunes obtained through GMINUIT when: G4BL profile is used and when the NumPy calculated profile is used. Many elements barely changed in their tune, however M9AQ11 had a relative change of 2.25%, justifying the need to be retuned.

3.4 XML2G4BL

Although the XML2OPTR HLA described in Section 2.1 provides an easy to use python script for generating TRANSOPTR simulation files from an XML database, no such functionality exists for G4Beamline simulations. From this lack of capability, the XML2G4BL prototype script was quickly thrown together in Python. In a similar manner to XML2OPTR, XML2G4BL reads in the XML files for a beamline and generates G4BL input files. This project was undertaken in the last few weeks of my work and as such are intended to be used as a proof of concept, and a springboard for future work on this script. At present, XML2G4BL has only been tested with the M9A leg. It presently only has the elements that are present in both TRANSOPTR and G4BL scripts. This includes: genericquads, genericbends (and their associated corners) and a custom built Wien Filter element that does not take into account fringe fields.

3.5 Wien Filter Strength

One of the important aspects of the M9A beamline is to rotate the spin polarized muons so that they are transversely polarized. This is achieved by using two Wien Filters each with an applied magnetic field of 0.05 T. It is necessary to check not only that the Wien Filters effectively rotate the muons, but also that they can separate out any positrons that have been created from muon decay coming down the beamline. Fig. 10 evidences how a field of 0.05 T does in fact spin rotate the muons while Fig. 11 illustrates that down to fields as low as 0.012 T 99.97% of the positrons can be filtered from the beamline. In Fig. 11b only 3 out of every 10000 muons results in a positron reaching a focussed area in a 1" radius and resulted from decay past the final Wien filter.

This test was conducted by using the same tune but with a positron beam as opposed to a muon beam. A script was written to slowly decrease the field of the separators and checking that no more than 0.01% of the particles reach the end of the beamline. This was done to show how a positron in the beam envelope would be effectively killed by the separators. However the positrons produced from the decay of the muons do not necessarily



(a) Before entering the first Wien Filter, the muon beam has a spin polarization that is antiparallel to it's momentum, as dictated by the decay kinematics.

(b) At the focus of the beamline the spin polarization has been successfully rotated to the positive x-direction.

Figure 10 – Histograms of the spin polarization of the muon beam before and after the Wien Filters show that the beam is effectively spin rotated as it should be.

fill the full range of the beam envelope. Nor must they necessarily be constrained within the phase space. As a result of this there was a minor discrepancy between the survivability when actual muons were run, however this effect was negligible, changing the percentage of positrons killed from 99.99% to 99.97% when 10 000 events are run. From this it was concluded that the decay products of the beam can be closely modeled as a beam itself, when trying to determine if the decay products will be killed.

3.6 Kicker Strength

The purpose of the addition of the kicker is to provide the capability to create muons-onrequest at the focus. The idea here is that the application of an electrostatic field can cause a vertical deflection in the beamline that results in the beam hitting against subsequent elements. Fig. 12a illustrates how this deflection in the beam after the kicker and the subsequent collision with the Wein filter and beam pipe after it, while Fig. 12b shows how statistically, the beam is killed. This allows an experimenter to turn off the kicker momentarily to provide the focus with individual muons.

Similarly to the test in Section 3.5, a quick script was created to run G4BL iteratively. Each time the kicker strength would be decreased and run again. Unlike the separator test, this was done with the time being used a random seed each run to add an extra degree of randomness. This ensured that the beam would reliably provide the ability to selectively allow muons, as opposed to being biased by the way G4BL usually seeds, using the track number. Each iteration this was run three times. The point that one of these runs exceeded the survivability was determined to be the minimum field value applicable. It should be noted that at this field value, the particles that reach the end of the beamline are not actually focussed in a 1" region. Around this field value they began to get qualitatively close, as seen in Fig. 12b and so I chose to limit this as a precaution. It is possible that the kicker field required is actually much lower if one wants to examine strictly the 1 cm focal region, or depending on what an acceptable flux is.



(a) View of the M9A beamline after the separators. An applied voltage to the Wien Filter as low as 0.012 T provides enough of a vertical deflection to kill 99.97% of the beam. The green region indicates a radius of 10 mm while the blue region indicates a radius of 1".

(b) View of the M9A beamline at the focus. An applied voltage to the Wien Filter as low as 0.012T provides enough of a vertical deflection to kill 99.97% of the beam. The blue region indicates a radius of 1". Positrons outside of the blue region came from decay downstream of the filter.

Figure 11 – The application of a Wien Filter field as low as 0.012 T is enough to prevent a statistically significant amount of positrons from reaching the focus region. This field removes approximately 99.97% of the beam. The green tracks represent muons, the red tracks represent positrons.



An applied voltage to the kicker as low as 17.9 kV provides enough of a vertical deflection to kill 99.99% of the beam.

(a) View of the M9A beamline after the Kicker. (b) View of the M9A beamline at the focus. An applied voltage to the kicker as low as 17.9 kV provides enough of a vertical deflection to kill 99.99% of the beam. The green region indicates a diameter of 10 mm while the blue region indicates a radius of 1".

Figure 12 – The application of a voltage as low as 17.9 kV provides enough vertical deflection to kill 99.99% of the beam. This indicates the specified voltage of 25 kV will be more than sufficient to provide muons-on-request functionality.



Figure 13 – When the solenoid in the M9H leg is powered, an extreme vertical deflection is felt by particles in the M9A leg. It appears highly unlikely that both legs could be run simultaneously without a significant amount of magnetic shielding.

3.7 Solenoid Induced Vertical Deflection

It was hoped that in the future the kicker providing muons-on-request capability would allow both the M9A and the M9H leg to be run simultaneously. The limiting factor to whether this is possible or not is that the M9H leg starts with a solenoid that is relatively close to the M9A leg. It was thus necessary to determine whether the solenoid would impact the M9A leg, and if it could be corrected with an asymmetrically driven quadrupole in the M9A leg. Using the solenoid field prescribed in the M9H G4BL simulation of 4 T, it is observed in Fig. 13 that the field causes a significant amount of vertical deflection in the M9A leg. It is thus concluded that the two legs cannot be run simultaneously without a great deal of magnetic shielding to be added to the M9A leg. The amount of shielding needed, and the proximity to the primary beamline makes this a great challenge and unlikely to occur.

3.8 The Tune

GMINUIT proved capable of providing a tune for the M9A beamline. The tune is compared against the TRANSOPTR tune in Table 3. Most elements had a relatively appreciable change in their tune value, with the highest percent change of 66.5%. The G4BL tune was actually quite reasonable, with a particle survival rate around 73%. Additionally, all surviving particles fall within a 2 inch diameter focus. Approximately 90% of the surviving particles fall within a focal area that is 1 inch in diameter. Furthermore, 25% of the surviving particles are focused into a circular region that is 1 cm in diameter. This can qualitatively be observed in Fig. 17, where the blue region indicates a circular area with diameter of 1". Less than 10% of the surviving particles do not focus into this blue region. The green region indicates a focal area with diameter of 1 cm.

Table	3 – ′	This table	compares	$_{ m s} { m the tunes}$	obtained	l throu	gh a G	MINUIT	script v	when the	G4BL
profile	was	used, and	when the	e TRANSO	OPTR wa	as run.	Most	elements	have a	drastic p	oercent
differer	ice, '	with an av	verage of	20.4%.							

Element	Polarity	TRANSOPTR Tune (A)	GMINUIT Tune (A)	% Change
M9Q1	-	114.0	114.9	0.79
M9Q2	+	57.17	54.20	5.33
M9AQ3	+	56.77	50.52	11.7
M9AQ4	-	74.84	66.88	11.2
M9AQ5	+	25.71	25.78	0.27
M9AQ6	+	30.19	22.35	29.8
M9AQ7	-	35.96	25.74	33.1
M9AQ8	-	48.39	35.26	31.4
M9AQ9	+	56.50	51.30	9.65
M9AQ10	-	15.54	31.02	66.5
M9AQ11	+	38.73	25.59	40.9
M9AQ12	-	69.29	60.77	13.1
M9AQ13	+	58.10	65.10	11.4



Figure 14 – Sigma profile for the TRANSOPTR calculated tune. The simulation was run in G4BL and the profile was calculated using NumPy to analyze the statistics from zntuples. The start of the beam agrees well with the G4BL tune, however the two simulations diverge near the first separator. Approximately 32-33% of particles reach the focus. In general, the shape agrees well with the results when run in TRANSOPTR. The deviations likely come from poor handling of fringe fields and the fact that the profile is shaped by the loss of particles.



Figure 15 – Sigma profile for the GMINUIT calculated tune. The simulation was run in G4BL and the profile was calculated using NumPy to analyze the statistics from zntuples. The start of the beam agrees well with the TRANSOPTR tune, however the two simulations diverge near the first separator. Approximately 71-72% of particles reach the focus.



Figure 16 – This image shows how most of the particles lost in the G4BL simulation are lost exiting M9AB2. This matches up with the sudden drop in the profile width observed in Fig. 15 around s \sim 7000 mm.



Figure 17 – The outer blue region indicates a diameter of 1". Considering that particles outside barely don't make it in, all particles clearly reach a focus of 1" in diameter. The inner green region indicates 1 cm in diameter

Mirroring the analysis from Section 2.3, the tune calculated from TRANSOPTR was used in G4BL to to compare the profiles. When one compares Figs. 14 and 15, it is easy to see that the two profiles agree reasonably well until the second last triplet of quadrupoles. This differs from what we saw in Section 2.3, where the profiles began to diverge drastically around M9AB2. This doesn't really come as a surprise when one stops to think about what is happening here. In TRANSOPTR, the beam phase space volume is preserved by Liousville's theorem, however this is not necessarily true in real life. In reality, particles can hit something and stop, as is treated in G4BL. In TRANSOPTR the G4BL tune extends well beyond the beam pipe because TRANSOPTR is unaware that a large portion of the beam has shrunk due to the physical constraints of the pipe. This causes the shape to change dramatically and indicates why what is observed in G4BL is so different from TRANSOPTR. Looking at the results in Fig. 14, we can see that just as observed in TRANSOPTR, there are several points along the beam line where the envelope becomes too large. This resulted in only 32-33% of muons reaching the target area. This is a significant loss compared to the 71-72% loss when applying the GMINUIT calculated tune using the NumPy profile, from Section 3.3.

One of the striking features of the GMINUIT tuning was that it appeared, as indicated by TRANSOPTR, that the initial phase space acceptance was too large to be practical. Fig. 16 illustrates how the beam envelope in the horizontal direction grows to be much wider than could be accepted by the beam pipe. Still, despite the large loss here, this tune had the best results, with $\sim 73\%$ of muons reaching the focus target. Attempts at forcing the beam to fit through this region results in a greater loss of particles elsewhere. This contradicts the results given by TRANSOPTR, which implied that the best tune would hit several elements just a little bit. I personally think that the decision made by GMINUIT is likely the better choice. This is because TRANSOPTR weights each element equally, the assumption being that maintaining a tight beam width is sufficient to ensure all particles reach the end. GMINUIT, however, can read the number of particles that reach the target as an input parameter. This allows G4BL to arrive at conclusions based on the bigger picture. There were tunes that could fit through M9AB2, however these tunes often had a larger focus and almost always had a lower survivability. This is almost certainly an issue with the initial beam parameters. If the initial phase space created was smaller, an investigation could be done to try to characterize a tune that would never exceed the beam pipe through the entirety of the beamline.

3.9 Limitations

Due to the complexity of G4BL there are many things that can go wrong. Some of these things are inherent problems in Monte Carlo simulations, and some of these are issues that reside within the G4BL program itself. The most glaring of these issues is that Monte Carlo simulations are quite computationally intensive. This places a great burden on users to make trade-offs between what is technically correct and what is practically achievable. An example of this can be seen in the creation of the beam element. One could create an initially huge phase space, that creates particles isotropically. The majority of these particles would never be able to reach the beamline. For instance, it is clear just by reasoning that the particles that are ejected from the source away from the beamline will never end up in the beamline. When one starts to place elements into the bealine, such as beam pipes, that allow for the killing of particles this results in the majority of events simulated having absolutely no effect on the results of the simulation, since they will just be killed. This has a couple of effects. The first of which is that it means the number of events required to run are much higher than they would otherwise need to be, in order to get enough particles at the focal region to have good statistics. This ties into the second effect, which is that all of the extra events that are run and killed quickly still consume computational power. When one is running <100000 events this might not be very noticeable, but the effect is certainly appreciable when you consider the increased number of events required for good focusing statistics.

An issue that arises solely from limitations of the code, rather than the limitations of Monte Carlo, is the way that G4BL builds the world. In brief, all elements are placed inside of a parent element. The top-level parent is the world itself, all elements are either "daughters" of the world, or "daughters of a daughter..." of the world. This is an efficient way to build the world, as it allows the world to be built bottom-up with only one read through of the code. The problem is that not every element can be a parent. Of particular note is the genericquad element, which is the element used to model quadrupole magnets. This is a serious issue because by the nature of the quadrupoles, expansion of the beam envelope in one transverse direction is guaranteed, at least within the quadrupole field. It is known that a series of quadrupoles can have a focussing effect in both transverse directions [2]. Thus it is possible for a beam to enter a series of quadrupoles, and to actually expand beyond the diameter of the beam pipe while inside but still come out with a beam width less than it entered with. This results in some particles, which should have been killed by beam pipe within the element, actually surviving since only the poles themselves limit the particles inside. Clearly this is an erroneous result.

This is illustrated most clearly in Fig. 18. This figure illustrates this effect by taking it to the extreme. In this simulation, a source is created with an initial divergence great enough that drifting the beam from the source to the end of the quadrupole, with the quadrupole turned off, will result in expansion of the beam to be greater than the radius of the beam pipe, despite entering with a radius less than the beam pipe. In Fig. 18a it is shown that these particles are seen to exit the quadrupole outside of the beam pipe. This is not physically possible in reality, as there is beam pipe inside of the quadrupole field region, restricting expansion of the beam envelope. This is a tremendous setback, as it means that a great deal of attention must be paid to the statistics, since particles may be reported by zntuples while existing outside of the beam pipe. Essentially one would have to use a beamlossntuple and look for tracks that end outside of the beam pipe, and then remove them from the zntuple files. As mentioned before, this is a waste of computer resources. It is isn't necessary to track the particles outside the pipe as they are essentially not real, but time is still spent tracking them.

Initially I tried to alter the C++ code for the generic quad element to allow it to take





(a) The beam envelope can grow significantly larger than the beam pipe would physically allow since quadrupoles cannot take children. This requires a more careful analysis of the statistics as well as wasting computational resources.

(b) By placing kill regions just outside of the quadrupole it is possible to approximately simulate a radial constraint within the field region. This drastically increases computational efficiency while leaving the result unchanged.

Figure 18 – Fittings can be placed on the ends of quadrupole elements to make up for the inability to place beampipe segments within quadrupole elements. This allows for particles to be more accurately tracked throughout the beamline, eliminating the need to determine if tracks recorded in zntuples are outside of the beam pipe or not, as they will be killed by the fittings.

children. I was successful in allowing them to take children, however the simulation still failed to interact with the children inside of the quadrupole. The code for this attempt can be found in my directory map, within the Scripts subdirectory. Rather than try to understand specifically what was incorrect, I opted to try a quick fix, since I did not know how long rewriting the genericquad element would take me. It is my belief that the issue arises from the fact that the genericquad element is actually constructed as subtracted volumes of a larger volume, with their inner volume subtracted to be the field region. It is my hypothesis that the inner field region is already treated similarly to a daughter, and so just altering the code to accept children is not enough, as you need to ensure those children are properly being placed in the field region and not the volume union of the subtracted volumes. I believe to get around this, it could possible to actually force a pipe creation in the creation of the genericquad element, rather than trying to pass it as a daughter. To preserve time however, cylindrical kill regions were placed just on the ends of the quadrupoles. These kill regions have an inner radius equal to the radius of the beam pipe, and an outer radius equal to the outer iron radius of the quadrupole.

Another problem encountered with the G4BL program is the way that elliptical beams are generated. Elliptical beams are created as a rectangular region with x and y half-widths supplied by the sigmaX/Y parameters. The momentum of the beam is equal to the value assigned to the meanMomentum parameter. The issue comes from not being able to assign any transverse momenta to the initial beam. Assigning any value to this parameter has no effect whatsoever on the beam. This is a problem because the target at T2 is at a 45 angle with respect to M9. The beam parameter allows the user to rotate the beam on its creation, which allows for the proper target orientation. The problem is then that the whole beam comes out as if that rotated angle is the longitudinal direction, since no transverse momentum can be applied. Therefore the user must resign to not having the proper orientation of the source, considering it is vitally important that the direction of the longitudinal axis is correct. This dichotomy is best illustrated in Fig. 19.



(a) It is necessary to run the simulations with an incorrect orientation of the T2 target, as it is the only way to produce an elliptical beam heading longitudinally.

(b) Rotating the beam when creating it is not possible since no transverse momentum can be given to the particles. This results in the beam terminating.

Figure 19 – Elliptical beams in G4BL cannot have transverse momenta at their creation. This prevents users from being able to rotate their source, since the beam will not be produced longitudinally.

4 Opera

One of the issues leading up to this term with the M9 beamline is the fact that M9B1 had shifted place and caused a misalignment in the beamline. Although this will physically be corrected, it would be a useful feature of the beamline to be able to steer the beamline using the quadrupoles in case a misalignment occurs in the future. Ideally, the M9Q2 quadrupole would be able to steer the beamline in such a way as to allow it to enter M9B1 correctly should it shift again in the future. Alternatively, the M9AQ3 quadrupole, after the dipole could be used, but since there are two legs it would be preferable to control this before the split in the beamline.

Unfortunately this is impossible in TRANSOPTR since there is no notion of alignment. Neither is it possible natively in G4BL, as the user merely provides the field strength rather than the individual currents supplied to the poles. G4BL does allow users to provide electromagnetic field maps through ASCII files however. It was concluded that rather than trying to write an element that also calculated the field map for an asymmetrically driven quadrupole, that it would be less work to use a well established program for creating the field map, and just having it be read into G4BL. The software chosen to model the quadrupole was Opera3D. Opera3D is a program that allows users to build a 3D model and determine various physical characteristics of the object [13]. The functionality needed for this project was the ability to determine the electromagnetic field produced and to export it in some sort of ASCII format to be used in G4BL.

4.1 Opera2BLFieldMap

Producing a working Opera model is only part of the solution to the simulation of asymmetrically driven quadrupoles. It is also required to be able to export the Opera field map and read it into a G4BL simulation. Opera fortunately has the capability to output the field map data into an ascii file. Complementary to this, in G4BL, is the fieldmap command which allows users to supply a text BLFieldMap file to supply an electromagnetic field to the simulation. Fred Jones has written a handy FORTRAN program, Opera2BLFieldMap, for



Figure 20 – Opera generated model of the M9Q2 magnet. This model will allow for the investigation of driving the M9Q2 magnet with asymmetric current to realign the beam should a future displacement in the M9B1 magnet occur since G4BL does not have asymmetrically driven quadrupoles implemented.

converting the Opera table format into the G4BL BLFieldMap format[14]. Putting the field into G4BL using Opera2BLFieldMap provides the result in Fig. 21, indicating that the field is being imported successfully to G4BL. This enables the investigation of asymmetrically driving M9Q2 in order to offset a M9B1 misalignment.

4.2 M9Q2

Opera proved capable of successfully building a model for the magnetic field produced by M9Q2. The result of the simulation is shown in Fig. 20. For testing purposes, it was deemed necessary to compare the fields produced by Opera and G4BL to check if there is good agreement, verifying whether the Opera model is appropriate for use in G4BL. A comparison of the field's affect on the beam envelope can be seen in Fig. 21. This was accomplished by first simulating the effect of the genericquad element and observing the shape of the beam envelope. Next, the field produced by Opera was imported into G4BL using fieldmap and placed at the center at the M9Q2 element, this time supplied with no gradient. In this sense, the fieldmap acts as the field produced by the quadrupole. Qualitatively, the comparison of the effect on the horizontal beam width yields little information, as shown in Figs. 21a and 21c. When one looks at Figs. 21b and 21d it is clear that the strength of the field map computed does not agree, as much greater focusing is seen in Fig. 21d.

This is likely the result of the current density being improperly calculated for use in Opera. Due to time constraints, it was not possible to investigate this further before writing this report. Considering that the results appear generally reasonable though, correcting this should just be a matter of carefully recalculating the supplied current density. The



(a) The horizontal view of M9Q2 in G4BL using the generic quad element.



(c) The horizontal view of M9Q2 in G4BL using the field modeled from Opera3D.



(b) The vertical view of M9Q2 in G4BL using the generic quad element.



(d) The vertical view of M9Q2 in G4BL using the field modeled from Opera3D.

Figure 21 – A comparison of the transverse profiles of a simulated beamline consisting only of M9Q2 and the source. Figs. 21a and 21b Show the affect of the G4BL calculated tune on the profile, while Figs. 21c and 21d show the field calculated by Opera3D. The general affect is quite similar, but the strength of the tuning is off, indicating that an error has likely been made in calculating the current density required in Opera3D.

magnetic field map takes approximately twenty minutes to calculate in Opera3D, however once generated just takes a couple of seconds to be read into G4BL.

4.3 Limitations

The most obvious limitation to using Opera3D is that calculating the model can be quite time consuming. As an example, modeling the field in Fig. 20 took twenty minutes to complete. This means that is not possible to quickly model the asymmetric quadrupole in G4BL this way, preventing any ability to tune with a program like GMINUIT, which might require hundreds of iterations. Opera instead merely provides the ability to get a feel for what kind of driving might be required, as well as providing a reliable standard to measure future attempts of modelling against.

The Opera model can be assumed to be quite accurate, and so models built off of assumptions required to speed up the calculation can be verified with the Opera model¹. One such model that I did not have time to test is to treat each pole tip as an individual entity and calculate the magnetic field map. The hope would be that a linear combination of the fields produced by each independent pole will be a close approximation to the field produced by treating the entire system as one entity and solving it in Opera. This would be preferred as the calculation for an individual pole is relatively quick to compute and could be handled in G4BL itself, either by adding in the fields with fieldmap, or by writing a custom element to compute these fields and combine them. This would likely provide the ability to write a GMINUIT script to steer the beamline misalignment by varying the currents supplied to each individual pole.

5 Conclusion

At the completion of the work term several results had been achieved. Most notably, tunes were successfully obtained for the M9A leg using both TRANSOPTR and G4BL. There was disagreements between the two simulations due to the nature of the computations involved, but the tunes do not vary too drastically. Using TRANSOPTR, a doubly achromatic focus was obtained at. The beam width in the horizontal direction is 1.96 cm and in the vertical direction was 1.95 cm. In G4BL, the tune supplied a final focus where all particles fall within within a 2" diameter focus. Additionally, 80% of surviving particles fall within a 1 cm diameter focus. Furthermore, it was found that the survival rate of the muons was approximately 72% using this tune, with the majority of particles being stopped coming out of M9B2.

Various restrictions on specification of elements were also successfully investigated. The minimum voltage required by the kicker to provide muons-on-request was found to be 17.9 kV provided that one wishes to remove 99.99% of particles from the beamline. Similarly, it was found that a minimum magnetic field strength of 0.012 T was required for the Wien Filters to prevent positrons that occur from decay in flight to be filtered by the separators. This field strength successfully eliminates 99.97% of positrons from reaching the target focus. It was also found that the specified field strength of 0.05 T successfully spin rotates the muons as required.

The M9Q2 quadrupole was also successfully modeled in Opera3D, and it was possible to export its field as an ASCII file and to put it into G4BL as a fieldmap. The strength of the

¹I just spoke quickly with Syd about this. An asymmetric quadrupole has been solved analytically. With any luck this will provide a usable field and could implemented directly into a G4BL element.

field between the Opera model and the generic quad in G4BL were not the same however, indicating that the current densities were likely not calculated correctly in Opera. There was not time to correct this though before writing this report. In the future, it would be advisable for someone to use the Opera3D model to verify models that could be implemented directly in G4BL, so that the beamline can be realigned quickly using GMINUIT. There are a couple models that could be tested quickly. The first is a linear combination of each independent pole. The second is an analytic solution referred to me by Syd.

Work was also done in conjunction with the beamphysics group regarding the accelerator XML database as well. The M9A and M9H leg had XML files prepared for them and uploaded to the database, allowing for the HLA XML2OPTR to generate TRANSOPTR files for the beamline. Furthermore, this allows the beamline to be tuned through the HLA web portal in the Beam Envelope application, requiring no knowledge of programming. A prototype for a similar script XML2G4BL was also produced, providing similar functionality but for creating G4BL input files. The XML2G4BL script presently only generates input file for beamlines consisting of: qenericquads, genericbends and Wien Filters, such as M9A. Extra components will have to be coded in the future, but should be relatively simple provided one knows what they need. Additionally, there were several shortcuts used to outline any glaring deficiencies in the script throughout using comments so that in the future someone can quickly identify basic problems.

References

- E. Heighway, A Second Order Beam Transport Design Code with Automatic Internal Optimization and General Constraints. Chalk River Nuclear Laboratories, aecl-6975 ed., July 1980.
- [2] K. L. Brown, "First-and second-order matrix theory for the design of beam transport systems and charged particle spectrometers.," Tech. Rep. SLAC-PUB-3381, Stanford Linear Accelerator Center, Calif., July 1984.
- [3] K. L. Brown and F. Rothacker, TRANSPORT: A Computer Program for Designing Charged Particle Beam Transport Systems. Stanford Linear Accelerator Center, slac-91 rev. 3 ed., May 1983.
- [4] D. Arseneau, "1at2 dimension conversation." Private Conversation, May 2018.
- [5] D. Arseneau, "Surface muon momentum distribution measurement." Beamline Measurement Stored on Computer, May 2018.
- [6] S. Kreitzman, "The m9a surface muon channel rebuild: M9a magnet power supplies," Tech. Rep. TRI-DN-06-17, TRIUMF, Nov 2006.
- [7] G. Clarke, "M9a request form," Nov 2008. Seq. #585.
- [8] A. Pikor, "Transoptr mode 2 conversation." Private Conversation, July 2018.
- [9] S. Agostinelli, J. Allison, K. a. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, et al., "Geant4: A simulation toolkit," Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 506, no. 3, pp. 250–303, 2003.
- [10] T. Roberts, G4beamline User's Guide. Muons, Inc., 3.04 ed., Feb 2017.
- T. Roberts, "G4beamine download." http://www.muonsinternal.com/muons3/G4beamlineDownload.php? dbassi@triumf.ca, Mar 2017. Accessed: 2018-08-03.
- [12] K. Paul, "Gminuit." http://www.muonsinternal.com/muons3/gminuit, Feb 2016. Accessed: 2018-08-03.
- [13] Cobham Technical Servies, Opera-3D User Guid, 18r2 ed., May 2016.
- [14] F. Jones, "Modified 3d field map block input." Private E-mail, July 2018.

A G4BL Build Instructions for RHEL7 derived OS

Run these instructions in command line to build G4BL from scratch

```
# Install all necessary libraries
yum update
yum install gsl gsl-devel fftw-libs fftw fftw-static
yum install qt5-qtbase-devel qt5-qtbase qt5-qtbase-doc qt5-qtbase-common \
            qt5-qtbase-gui
# Optionally (recommended) install MPI to use multithreads for simulations
yum install openmpi openmpi-devel
# Build root 5.34 in msrorg -- has CINT, and is used for musrfit
# Will have to be altered off musim1, just make sure you build root 5.34 \setminus
# which is needed to use historoot.
. /home/msrorg/root/bin/thisroot.sh
# Also put preceding line in personal profile (.bash_profile) \
# Could use system root-6 instead for g4bl. Historoot needs root-5 and \setminus
# CINT though. - Donald
# I actually put it into my bashrc since I don't like using \
# too many different .bash* files - Dylan
# I also put '. /home/msrorg/root/bin/thisroot.sh' > /dev/null \
# in here too, since my OS root is ver. 6
export GEANT4_DIR=/home/asnd/build-g4bl3.04/geant4.10.03
export GSL_DIR=/usr
export FFTW_DIR=/usr
PATH=/usr/lib64/qt5/bin:$PATH
# Make build directories and untar the archives
cd
mkdir build-g4b13.04
cd build-g4b13.04
tar -xf ~/scratch/G4beamline-3.04-source.tgz
tar -xf ~/scratch/geant4.10.03-source-tjr.tgz
# Build geant4 from patched sources geant4.10.03-source-tjr
mkdir geant4.10.03
cd geant4.10.03
cmake3 -DCMAKE_INSTALL_PREFIX=$PWD -DBUILD_SHARED_LIBS=OFF -DBUILD_STATIC_LIBS=ON \
       -DGEANT4_USE_SYSTEM_EXPAT=OFF -DCMAKE_BUILD_TYPE=Release \
       -DGEANT4_USE_RAYTRACER_X11=ON -DGEANT4_USE_QT=ON ../geant4.10.03-source-tjr
make -j$(nproc)
make -j$(nproc) install
cd ..
# Build g4b1
# edit G4beamline-3.04-source/g4bl/CMakeLists.txt
# Change
> file(GLOB GEANT4_PKG ${GEANT4_DIR}/lib/Geant4-*)
```

```
# to
< file(GLOB GEANT4_PKG ${GEANT4_DIR}/lib*/Geant4-*)
If making MPI version, edit MPI.cmake to insert the block:
elseif(${SITE_NAME} MATCHES musim) # XXX = the result of 'uname -n'
set(LIBS ${LIBS} mpi_cxx mpi) # the libraries your MPI uses
include_directories(/usr/include/openmpi-x86_64) # the MPI include dir
link_directories(/usr/lib64/openmpi/lib)# # the directory for the libs
before the final else().
# Note: You will have to append 'module load mpi' somewhere into your .bashrc file \
# or call once in every new terminal before trying to run g4blmpi
mkdir G4beamline-3.04
cd G4beamline-3.04
cmake3 ../G4beamline-3.04-source
or
cmake3 -DG4BL_MPI=ON ../G4beamline-3.04-source
make install
# (I did make install and got distro files, but if you just "make" \setminus
# the executables aren't even put in the bin dir. The install puts \setminus
# all used libs in a private lib dir, which fail after system updates, \setminus
# so hide them away.)
# Cleanup
# You can add a source command for the g4bl you built into your bashrc
# Ex. source /home/dbassi/Downloads/G4beamline-3.04/bin/g4bl-setup.sh > /dev/null
cd
rm -r G4beamline-3.04
tar -xf build-g4bl3.04/G4beamline-3.04/G4beamline-3.04-Linux64.tgz
cd G4beamline-3.04
mkdir lib/unused
mv lib/*.* lib/unused
```