Designing the Matching Section using Electrostatic Quadrupoles in a Compact Linear Accelerator

Lawrence Berkeley National Laboratory
SULI Summer 2020
Brian Wynne

In our group we designed a compact linear accelerator that uses radio frequency acceleration units and electrostatic quadrupoles (ESQs) as focusing elements. A matching section is being designed to avoid the loss of beam current between the ion source and the acceleration section. Optimizing the transport from the source to the acceleration section is also known as matching the beam to the acceleration section. We simulated the optimization of this section by developing a Python program, using an existing Java program, and solving the Kapckinsky-Vladimirsky equations that describe the radial evolution of the beam profile. Following these computationally inexpensive tests, a more rigorous analysis is done using simulations in WARP, a 3D Particle-in-Cell simulation framework. Results presented here include matching sections designed with the Python and Java programs, as well as simulations of these designs for emittance or perveance dominated beams. The results are then compared to full 3D-Warp simulations of the same design. Future improvements involve implementing an interactive user interface that allows easy exploration of the parameter space. Future work will include verifying the simulation results experimentally, as well as performing tests on wafers to determine breakdown voltage to determine the maximum allowable ESQ voltage.

I. INTRODUCTION

Ion beams consist of charged particles and have applications ranging from medical devices to electronics manufacturing. One important application is with high intensity beams for heating the fuel in nuclear fusion to generate electricity. This could be beneficial in both inertial confinement and magnetic confinement fusion systems.

In order to attain high energy beams, the ions must be accelerated. One type of particle accelerators uses radiofrequency (RF) resonators. Reducing the size and cost of the accelerators is important for the development of these devices.

The objective of our group's current project is to develop a 100 keV, 1mA multi-beam linear accelerator prototype. The group had a previous project on this topic where they showed that all the components (acceleration units and focusing units) work and in the current funding cycle are building a working prototype that integrates these components. To achieve this, experiments are performed, new layouts are designed and tested together with collaborators from Cornell University. The group has been using electrostatic quadrupoles (ESQs) and radio frequency (RF) wafers, and the accelerator is made by stacking these wafers. This method, using printed circuits boards (PCBs), has the benefits of fast and cheap manufacturing. In addition to the lower cost, the current can be increased by enabling the acceleration of many parallel beams.

Previously the accelerator was built with a 3×3 array of beamlets, but it has been upgraded to use approximately 10×10 beamlets to increase the total beam current, resulting in the need for updated simulations and re-designs of other beam components.

The Multiple Electrostatic Quadrupole Array Linear Accelerator (MEQALAC) uses RF acceleration and focusing elements called electrostatic quadrupoles (ESQs). Typically, pairs of ESQs, so called doublets, are used where the second ESQ is rotated 90° in respect to the first ESQ. A single ESQ focuses in one direction while defocusing in the other direction. Through the use of a doublet, there is an overall focusing effect in both directions. In order to optimize the placement of focusing elements, one can consider the beam envelope. The beam envelope (a and b in the horizontal and vertical direction) is defined as the evolution of the radius along the Z direction. Differential equations can be used to estimate the beam behavior along the length of the accelerator.

$$a'' = \kappa a + \frac{\epsilon^2}{a^3} + \frac{2K}{a+b}$$

$$b'' = -\kappa b + \frac{\epsilon^2}{b^3} + \frac{2K}{a+b}$$

Equation 1A and 1B

The Kapckinsky-Vladimirsky (KV) differential equations shown above describe how the root-mean-square (rms) beam radius evolves in a lattice of focusing elements, and provide the basis for the Python and Java simulations. The first term of the equations relates to the focusing force, κ, and it has the opposite sign for the horizontal and vertical directions. The second term

contains ϵ, the emittance, which describes the average spread and random motion of particles. The third term is a measure of the tendency of particles to feel the effects of each other, so called space charge effects. Perveance, K, determines the level of space charge effect and causes defocusing of the beam. Assumptions of the KV equations include constant emittance and no particles lost. These KV beam envelope differential equations were used as the first step in simulating the beam in Python and later with Java. Then the Java program with the KV equations was utilized for an interactive solution output. It was investigated how many focusing elements were needed and where to put them. At least four focusing elements are required within the matching section, based on the degrees of freedom of the KV equation, but more were needed for a practical result.

The goal of the internship was to find a matched solution between the ion source and the acceleration section. A matched beam is one in which the external focusing forces of the ESQs are balanced with the defocusing effects of the beam from emittance and space charge repulsion[1]. The ion source is where the beam is produced and the RF section is where the beam is accelerated, and it is desired to match these two to lose less of the beam. The matching section goal is to decrease beam loss and match the beam conditions of the source to the oscillations of the acceleration section[2]. The source emits a symmetric cylindrical beam shown in *Figure 1A*. This beam can be transformed through an ESQ focusing lattice, in order to have the periodic oscillations of a matched solution shown in *Figure 1B*. The goal of the matching section is to transform the beam from the initial conditions to the matched beam solution with minimal losses. The steps to accomplish this goal started with obtaining and confirming the input parameters. Next, a certain number of ESQs and their locations were defined. Then the ESQ voltages were tuned and optimized using the Python and Java files. Once a beneficial and practical result was proposed, it was further analyzed in more depth using the WARP simulations.

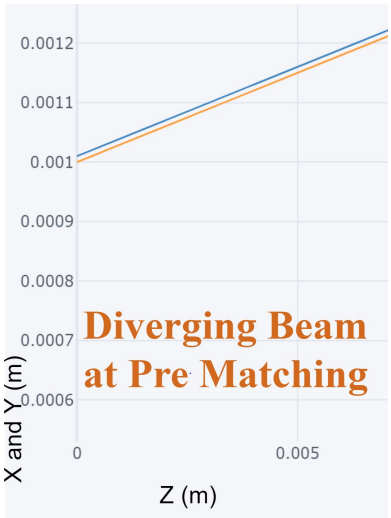

Figure 1A: *Initial beam conditions*

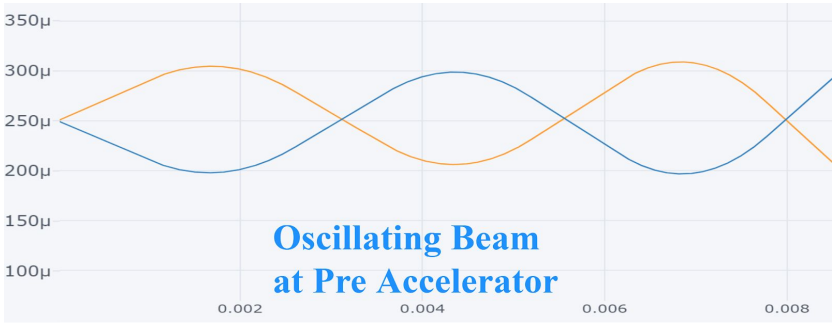

Figure 1B: *Matched beam oscillations*

A variety of techniques were used to find and optimize the matching section. Python was used to work with the differential equations, finding and modeling solutions quickly. Also, the Java program was utilized to work with focusing to find matched solution outputs. These methods treated the beams as an average of the particles, which was useful for quickly solving and identifying solutions for the matched beam envelope. Following the success of the previous 1D simulations, the resulting design was used with WARP simulations. WARP is a particle-in-cell framework for simulating beams. It calculates the electric field and then traces macro particles, that are the equivalent of thousands of ions, through the electrical fields handling space charge effects correctly..

II. MATERIALS AND METHODS

A. Methods of simulation

As a remote internship, the materials required pertained primarily to computer software and programs for simulations. The parameters needed for the simulations involved the beam envelope position and direction (the beam radius in the x and y direction and the beam angles). The beamlets from the plasma ion source are initially diverging [3] and the emitting radius and divergence angle have been measured experimentally. This was done by measuring the beam profile at two different z-positions using a scintillator. Also, in addition to the beam conditions at the beginning of the matching section, the needed beam parameters at the end of the matching section were known based on previous simulations. Additional input parameters were the beam current and emittance, which have been measured and calculated from experimental results.

Python 3 was primarily used for simple simulations. Libraries involved include numpy,[4] pandas,[5] and cufflinks[6] for general computation and graphing, and SciPy odeint[7] for solving the differential equations. Runtimes and computational expenses were low for this work, only taking fractions of a second to run. The Python simulation was instrumental in implementing quick changes, as ESQs can be easily added or removed, and their location and dimensions were simple to adjust. The Python simulation was also used for combining and plotting the results from all the simulations.

A Java code was used to optimize the beam envelope, with an interactive user interface. Input text files containing the initial conditions were imported into the Java program. The individual voltages of the ESQs could be adjusted with a slider bar to see the effect on the whole beam. Also, there was a matching feature for fine tuning the voltages to adjust to a desired exit beam radius and angle. The Java simulation was valuable for changing ESQ voltage because it had an interactive user interface. The inputs such as the number of ESQs, their positions, and their size were input using text files. Additional beam parameters were also defined as inputs using the text file, but could be adjusted easily using the GUI. There was also a matching functionality, for finding the voltages to a specified exit beam conditions. But it was limited because the solver would not converge to the result unless the input voltages were close to the solution. This feature was used for fine tuning, but was not useful for finding the possible

solutions initially. The output was the simulated beam envelope visualized in the Java program, which can be exported as a text file. Similar to the Python code, the Java program was computed almost instantly, with no significant runtimes.

A third approach was with WARP simulations, using Fortran and Python and was supported on Linux or Mac OSX, so with a Windows computer, a virtual machine was used. The VirtualBox Manager Oracle VM was used to run the Ubuntu operating system (Linux). This method provided a more accurate representation when simulating the experimental setup, since it included the real geometry of conducting elements and accounted for fringe fields. WARP was much more computationally expensive than the Python or Java simulations, and could take hours to run. Because of the long runtime, WARP was typically used only as a final check to ensure that the simulation results using the differential equations were valid. Additionally, these long runtimes drove the need for an accurate fast simulation with Python and Java, because these allow for rapid feedback and optimization that is impossible using WARP alone.

B. Beam parameters

The beam emittance is the volume in phase space occupied by a particle beam. A higher emittance corresponds to a higher divergence angle, and a smaller emittance corresponds to a smaller divergence angle. Emittance can be assumed to be conserved throughout, which is valid if there is not significant particle loss. The emittance can be measured anywhere, and should be the same, but is typically measured at the beginning.

The beam energy changes along the beam after the matching section due to the RF wafers in the acceleration section. However, the beam energy is approximately constant throughout the matching section and can be set in the simulation at the beginning by injecting a beam of constant kinetic energy particles and is normally 7-10 keV. The velocity in Z of around 2×10^5 m/s is much greater than the transverse velocity (horizontal or vertical). The linear charge density, λ_q, is almost constant in coasting (unbunched) beams such as the one in this experiment. This is because λ_q is defined as the beam current over velocity, and both are assumed to be constant.

The mass (amu) of the ions for a fixed beam energy results in different ion velocities. In the acceleration section a drift distance needs to be adjusted to the ion velocity, but in the matching section the energy is kept constant so the mass of the ions does not change the outcome of the results. Therefore, the matching results from Argon beam simulations can be applied to Hydrogen and Helium ion beams as well.

C. Matching section design

The goal of the matching section is to ensure that if the beam from the source does not have the right initial angle, then focusing elements are used to make a matched beam to minimize beam loss. Limitations while designing the matching section include the beam radius, which is limited based on how big the beam aperture can be. In the 10×10 design, the

beam-to-beam distance is fixed, therefore limiting the maximal beam aperture that can be used. Another limit is the breakdown voltage of the ESQs: the maximal ESQs focusing strength is limited by the maximum voltage that can be applied. The breakdown would likely happen in the area directly between the positive and negative electrodes. Furthermore, when focusing the beam, it is desirable to avoid a very small beam-envelope radius, as this can lead to undesirable and unpredictable beam effects.

Limitations to consider include the voltage on the ESQs and the beam radius. The maximum voltage is limited by electrical breakdown, across the wafer surface or vacuum gap. This breakdown is associated with large current spikes and discharge arcs. This leads to damaging hot spots on the surface, causing it to be unable to hold its original voltage and rendering it unusable.

It is preferred to have small oscillations of the beam envelope, where the ratio of horizontal and vertical beam radius never gets too high. The matching section can be as long in Z direction as it needs to be, but a shorter matching section is preferred because the vacuum chamber is limited in length, and the aim is to design a compact accelerator.

D. ESQ wafer parameters

As the PCBs were upgraded from 3×3 arrays to 10×10, they were densely packed. The ESQ wafers were PCBs with drilled out and metal coated holes in yellow for the electrodes, as seen in *Figure 2*.

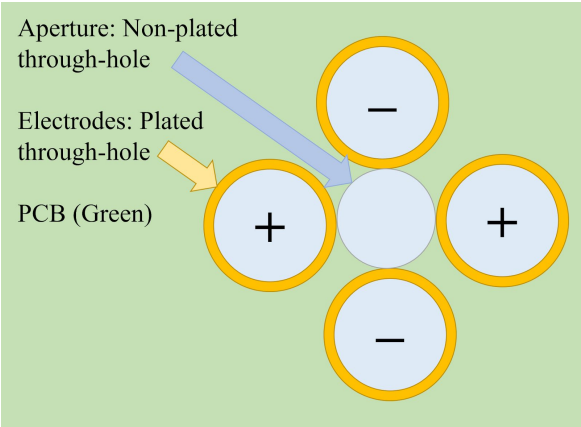

Figure 2: *Diagram of One Unit Cell of ESQ Wafer*

This geometry required less electrodes than previous models because they can be reused by the adjacent beam apertures. More wafers may be needed than in previous matching section designs because the wafers are shorter in thickness in the Z direction. These wafers are desirable to use because of their low production costs, only a few dollars each. The parameters for the beam and ESQ wafers are shown in Table 1.

Table 1: *Beam and ESQ Parameters*

Aperture diameter	1.1 mm
Electrode diameter	1.3 mm
Total thickness of PCB	1.59 mm
Beam current	7-10 keV
Minimum distance between ESQs	0.3-1 mm

III. RESULTS

A. Primary Results

The overall desired result for the matching section was to determine how many focusing elements to use and where to put them: a solution to the matching section. This had been found initially using Python and Java simulations, and was confirmed with the WARP simulations. The WARP result in *Figure 3* shows the beam envelope, with the horizontal and vertical 2•rms, as well as the voltages applied using ESQs. Table 1 displays the 19 ESQ voltages and center positions used in WARP simulations for this matched solution.

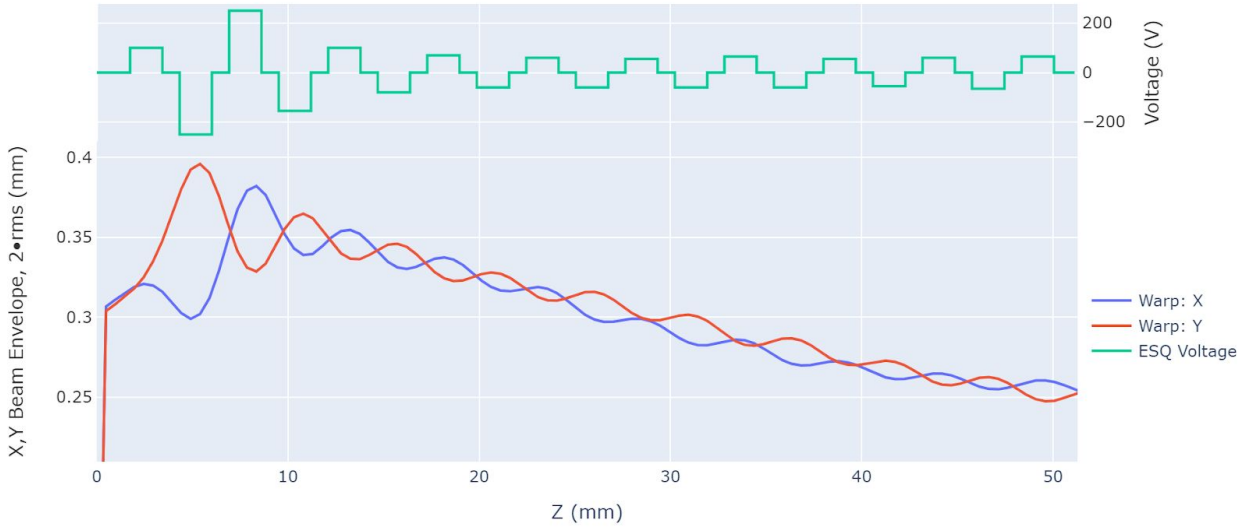

Figure 3: *Beam Envelope of a Matched Solution with WARP*

Table 2: *Voltages and Center Positions of ESQs for Matched Solution from WARP*

ESQ Wafer #	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Voltage (V)	90.9	-227.3	227.3	-14.9	90.9	-72.7	63.6	-54.5	54.5	54.5
Position (mm)	2.59	5.18	7.77	10.36	12.95	15.54	18.13	20.72	23.31	25.9
ESQ Wafer #	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	
Voltage (V)	50	-54.5	59.1	-54.5	50	-50	54.5	-59.1	59.1	
Position (mm)	28.49	31.08	33.67	36.26	38.85	41.44	44.03	46.62	49.21	

Figure 4 shows the ESQs with their electrostatic potential in the Z-X Plane. *Figure 5A* and *5B* show the x component of the electric field in the X-Y Plane at ESQ1 and ESQ2 respectively. This indicates how the polarity of the ESQs are inverted, focusing the beam in one direction and defocusing in the other.

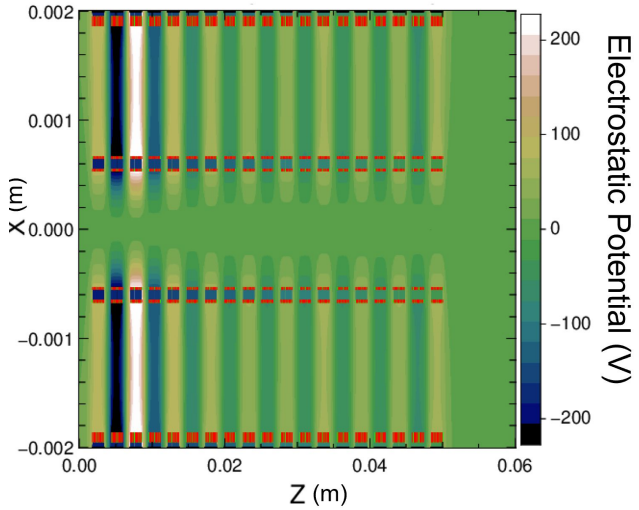

Figure 4: *Electrostatic Potential (V) in Z-X Plane in WARP on ESQs*

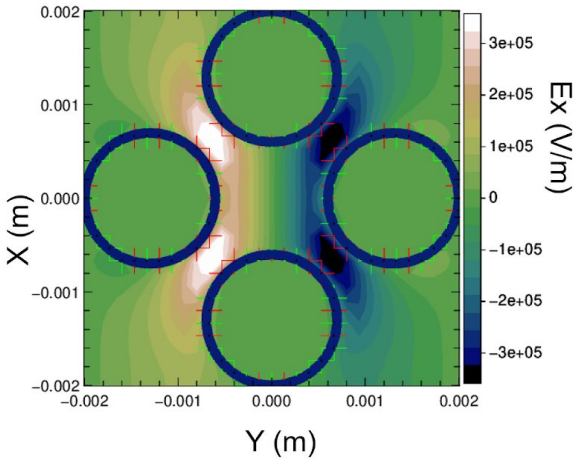

Figure 5A: *Electric Field (x component) at ESQ1*

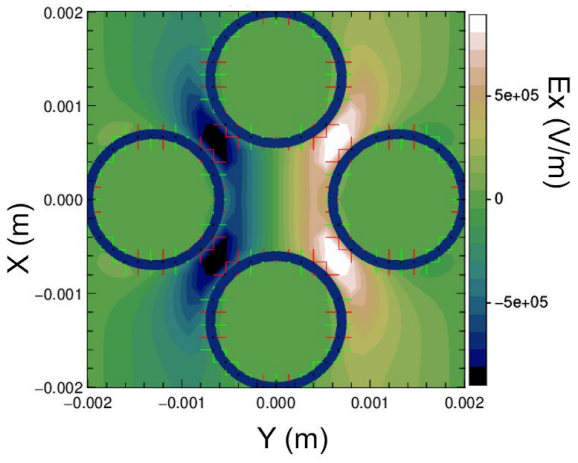

Figure 5B: *Electric Field (x component) at ESQ2*

B. Stability Analysis

In order to test the stability of the matched solution in *Table 2*, a stability analysis was performed on key parameters using WARP simulations. The simulations were run using voltages from *Table 2*, an initial beam radius of 0.3mm, diverging angle of 9mrad, beam current of 0.09E-6 A, and emittance of 1.42E-6 m•rad. First, the divergence angle was adjusted with a constant radius of 0.3mm in *Figure 6*. This showed the solution is stable for a divergence angle of approximately 3 to 12mrad. Then the beam radius was adjusted with a constant divergence angle of 9mrad in *Figure 7*. The matched solution is most ideal with beam radii of 0.25 to 0.35mm. All ESQ voltages were adjusted as a whole in *Figure 8*, with significant variances past -10% or +20%. The voltages were randomly varied individually by +/– 10% in *Figure 9* and *Figure 10*, and these voltages are shown in *Table 3*. The beam current was adjusted while keeping emittance constant at 1.42E-6 m•rad in *Figure 11*. The matched solution was stable for emittances of 0.09E-4 A or less. The emittance was adjusted while keeping the beam current constant at 0.09E-6 A. The solution was unstable at emittance of 1.42E-5 m•rad, and was similar but not exact at emittances lower than 1.42E-6 m•rad.

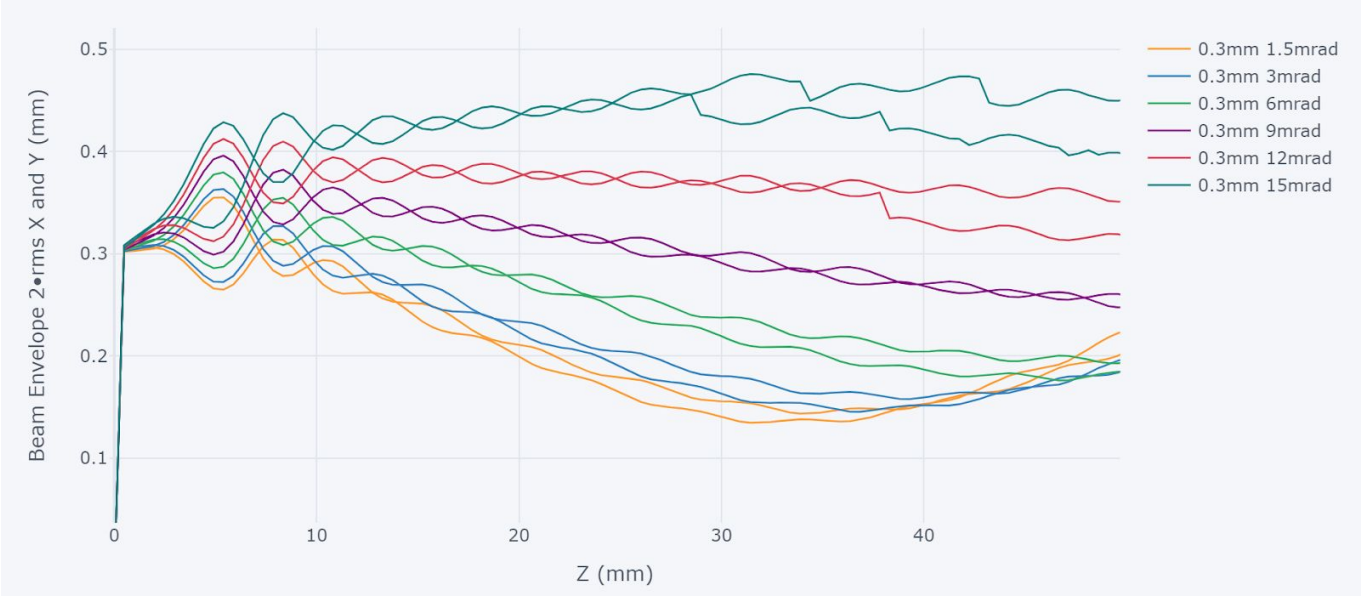

Figure 6: *Adjusting Divergence Angle with Constant Radius*

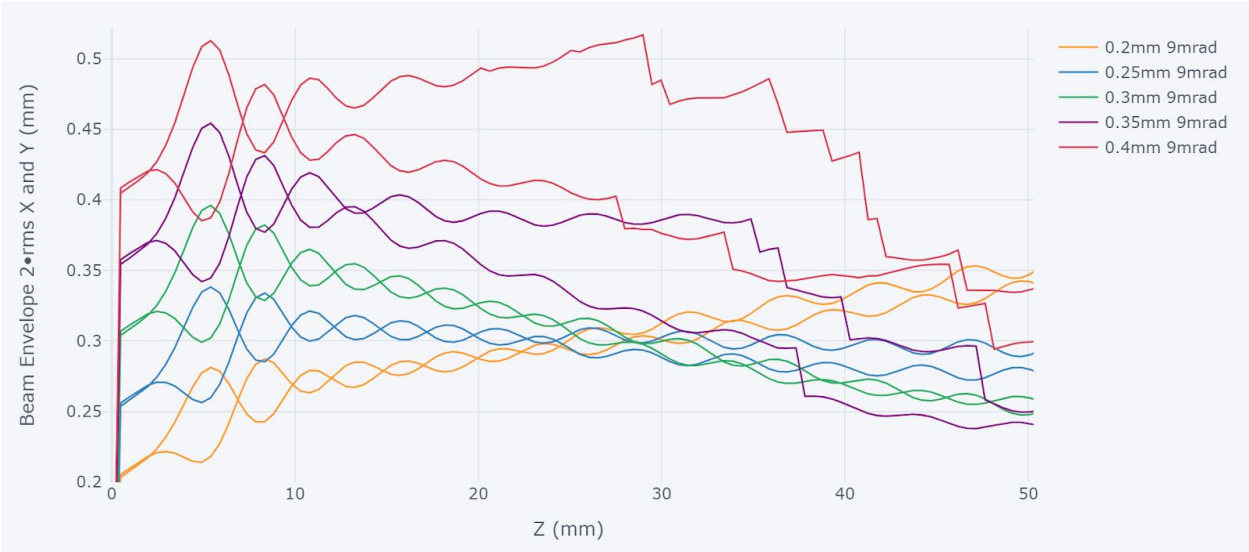

Figure 7: *Adjusting Radius with Constant Divergence Angle*

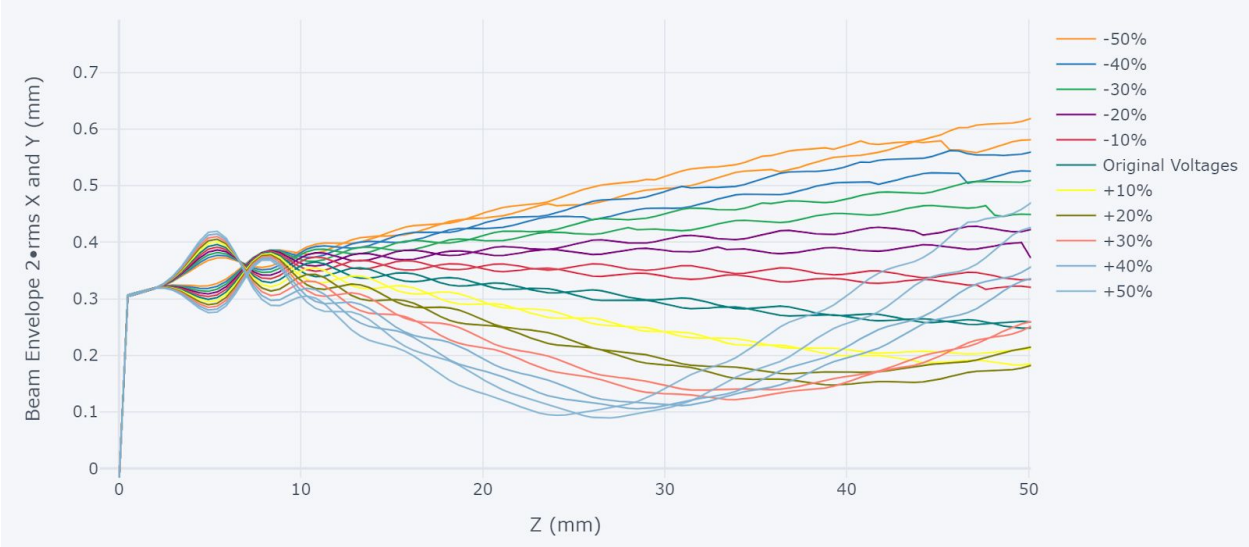

Figure 8: *Adjusting All Voltages Together*

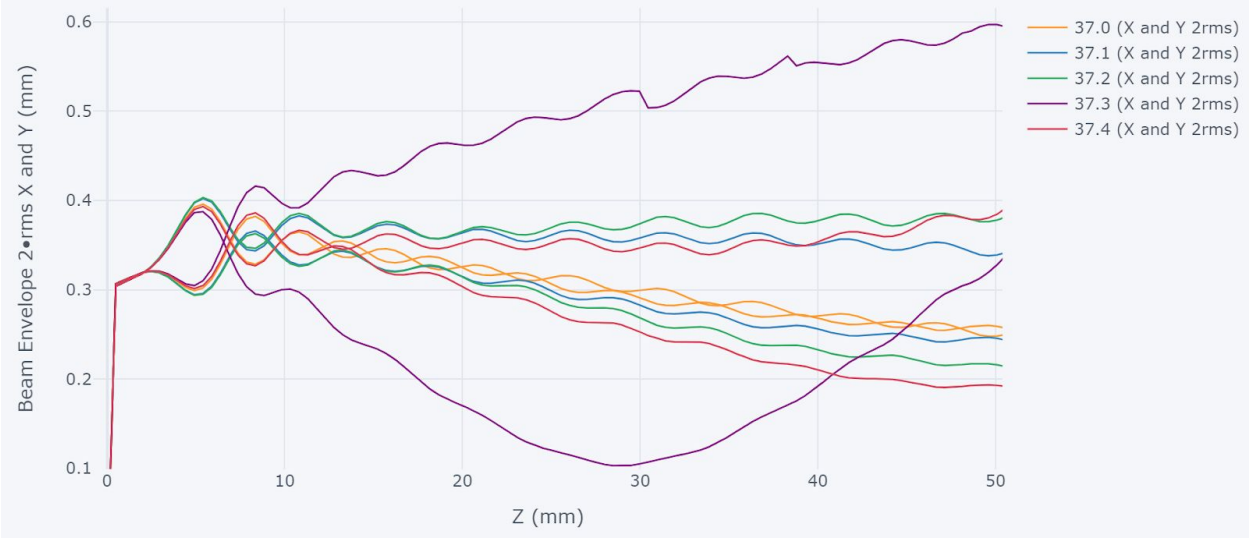

Figure 9: *Randomly Varying ESQ Voltages within (+/– 10%) 37.0-37.4*

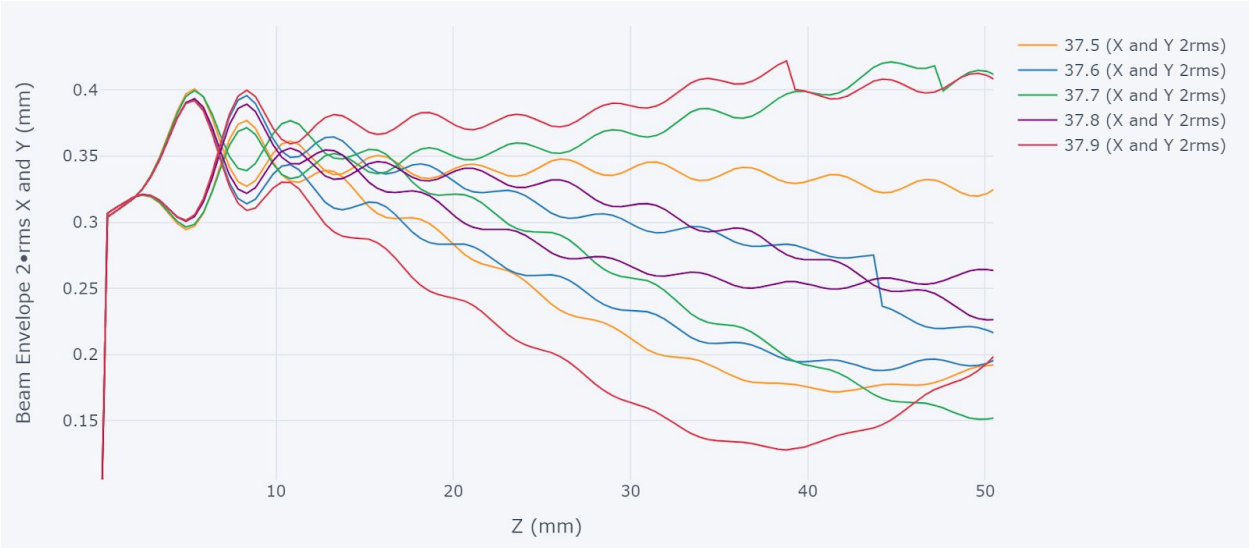

Figure 10: *Randomly Varying ESQ Voltages within (+/– 10%) 37.5-37.9*

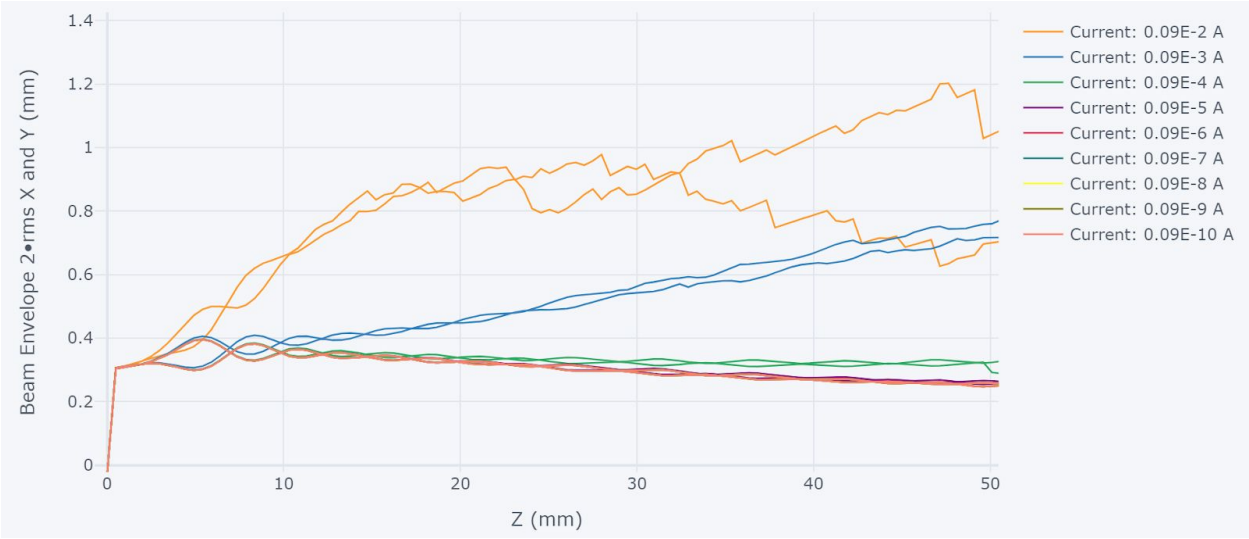

Figure 11: *Adjusting Current with Constant Emittance (1.42E-6 m•rad)*

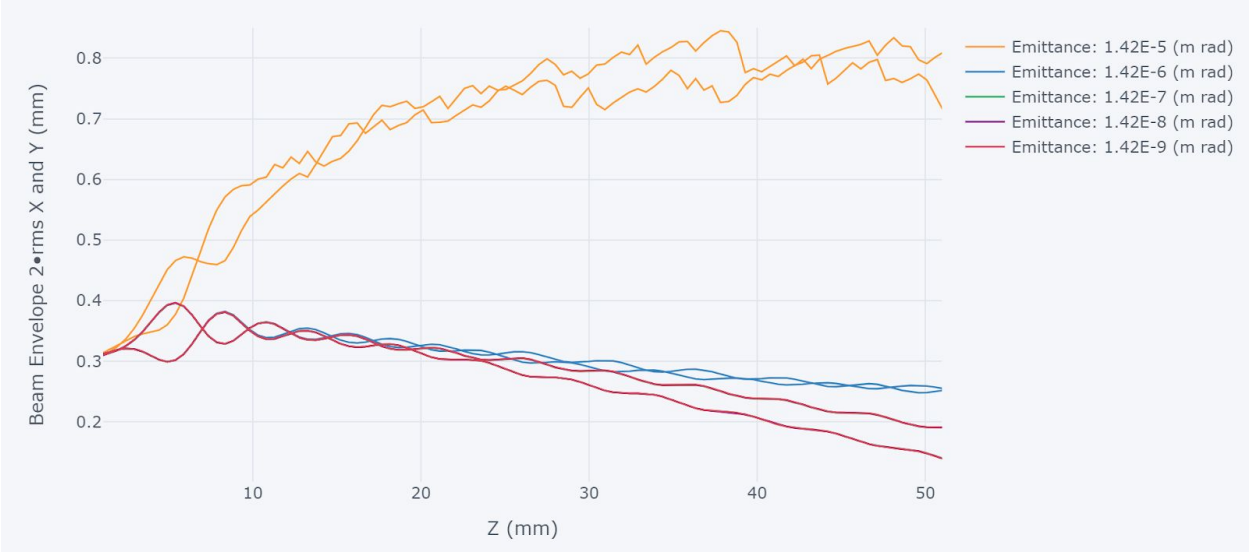

Figure 12: *Adjusting Entrance with Constant Current (0.09E-6 A)*

Table 3: *ESQ Voltages (V) with Random +/– 10% Adjustment*

ESQ #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
37.0	90.9	-227.3	227.3	-140.9	90.9	-72.7	63.6	-54.5	54.5	-54.5	50.0	-54.5	59.1	-54.5	50.0	-50.0	54.5	-59.1	59.1
37.1	98.4	-220.0	209.9	-132.5	88.0	-67.4	59.4	-55.3	57.6	-55.6	52.0	-51.3	54.9	-55.7	49.8	-47.6	49.9	-55.4	57.1
37.2	99.6	-218.3	206.6	-136.7	97.6	-77.2	68.8	-52.1	50.9	-49.4	50.6	-57.2	60.7	-55.5	48.7	-53.2	59.0	-53.4	53.4
37.3	81.9	-248.5	219.5	-135.1	85.6	-70.9	62.0	-54.7	52.3	-52.2	47.4	-53.0	54.0	-52.9	51.6	-47.2	53.7	-57.5	58.8
37.4	86.8	-223.8	231.1	-134.9	90.5	-73.4	60.7	-51.3	53.2	-57.2	53.9	-53.5	64.5	-51.5	49.8	-48.2	57.2	-55.3	63.7
37.5	99.1	-240.8	234.3	-134.6	94.1	-72.5	63.6	-49.4	51.4	-56.2	52.6	-56.0	56.6	-50.7	45.4	-49.7	55.8	-56.4	63.6
37.6	87.3	-239.5	242.5	-146.3	96.2	-79.6	65.8	-53.0	52.9	-59.0	53.0	-57.1	63.6	-51.1	47.3	-46.3	58.4	-62.8	58.8
37.7	95.1	-220.6	213.8	-138.0	83.4	-78.8	65.0	-53.0	49.3	-56.9	52.8	-58.6	62.9	-59.7	49.9	-54.3	57.7	-57.7	58.5
37.8	88.0	-230.9	232.2	-132.1	88.6	-65.6	57.5	-56.6	49.8	-56.1	49.2	-51.8	54.8	-56.9	49.9	-53.7	57.9	-60.8	58.1
37.9	87.6	-244.2	236.1	-141.4	83.0	-70.3	65.3	-57.7	50.9	-59.5	48.9	-53.5	55.4	-50.6	52.0	-49.7	55.3	-58.5	59.1

C. Overview of Additional Results

The first major result was the creation of a Python ordinary differential equation (ODE) solver for KV equations. Next, initial trials were completed in Java and Python for comparison. Another result was the addition of plotting the ESQ voltages on top of Python output, to be able to easily see how the placement and voltage of the ESQs impact the beam envelope.

Other simulations were run to test the code and the parameters. Simulations were run to see if changing the error settings in the Python differential equation solver affected the result. It did not change the beam envelope solution unless the error was increased to be very large. Other simulations were also done in Java to see if the beam was emittance or perveance dominated. This was completed by varying current and emittance and examining beam response, to see which is dominating. This can be important for experimental implications because it indicates what researchers must watch out for. It was found that the beam was in-between, not dominated by one or other, with both having some effect.

Another important result was getting the Python and Java results to align with the WARP results. This was critical because testing the effect of and adjusting ESQ voltages would take hours to complete in WARP. The instantaneous nature of the KV equation solvers was much more practical for coarse adjustments and seeing the effect of changes in real time. But if the fast simulations did not accurately predict the 3D simulation results, then they would be of little use. At first, after ensuring that all simulations had the same inputs for initial beam radius, divergence angle, current, and emittance, the Java and Python results matched up well but the WARP was not as similar. This was expected as both the Python and Java 1D simulations were based on the same equations, while WARP provided a more realistic representation. Two corrections were implemented to help the 1D simulations match up better. The first was by lengthening the thickness of the ESQs in the Z direction with the Python and Java inputs. This helped account for the fringe fields by extending the effects of the ESQs beyond the reach of the hard edge fields. The second correction was a correction factor of 1.1 for the ESQ voltages used by Python and Java compared to that of WARP. For a given voltage being modeled in Python or Java, WARP gave a similar result if this voltage was divided by 1.1. For example, if the ESQ voltages found in Python or Java were [100,-250,250,-155], these would correspond to results in WARP with voltages [90.9, -227.3, 227.3, -140.9]. With these adjustments, the three simulations were able to match up much better, and therefore adjustments made with the fast 1D simulations could be made, with WARP only needed as a final check. *Figure 13* compares the results from the three types of simulations, which shows considerable agreement.

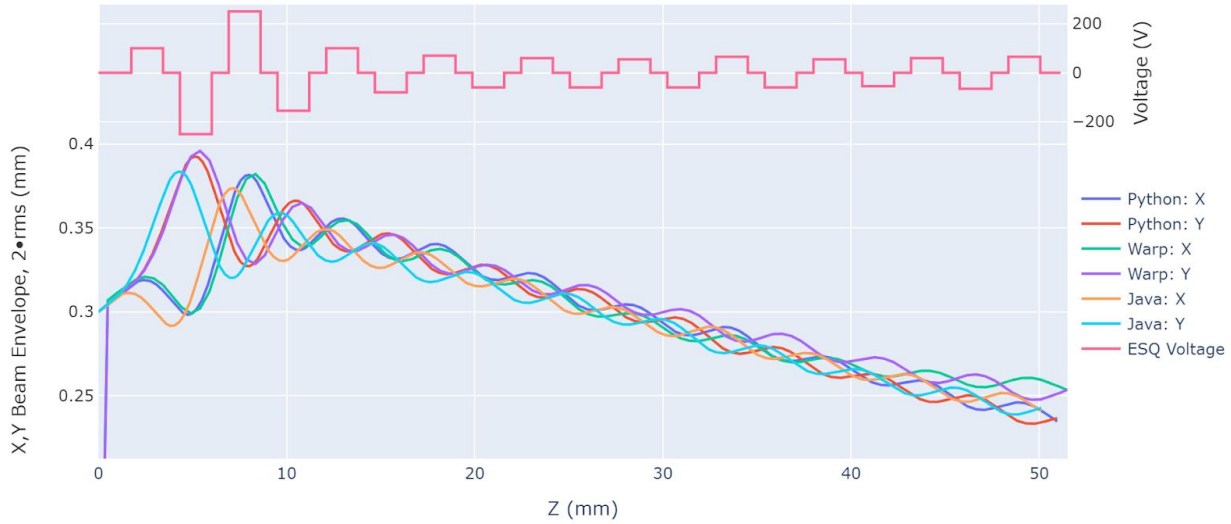

Figure 13: *Comparison of Beam Envelope of a Matched Solution with Python, Java, and WARP*

Another result had been investigating the geometry file for the WARP Single Species Simulation. One aspect that needed to be changed was that WARP with the original geometry file was using an ESQ doublet, but for the matching section the use of an ESQ singlet would be better. In the beamline, it is desired to have doublets to focus in both X and Y. For the matching section, more flexibility is preferred, as adjacent ESQs can have the same sign for voltage. This can be useful, as the ESQs are relatively short with low voltages, so their focusing effects can be combined. After changing the ESQs to be singlets, the geometry still needed to be updated to the new design. The initial geometry, shown in *Figure 14A* is being modeled for the ESQs in the acceleration lattice. However, this is not the geometry for the PCB boards that are going to be used in the matching section, which uses the geometry in *Figure 14B*. This was implemented by adding the ESQ_singlet() class into the geometry file, using hollow cylinders by defining each annulus as conductors. It was not required to divide up into quarters anymore, just specifying the center position of each electrode annulus. The green PCB is ignored as only conducting metals are defined.

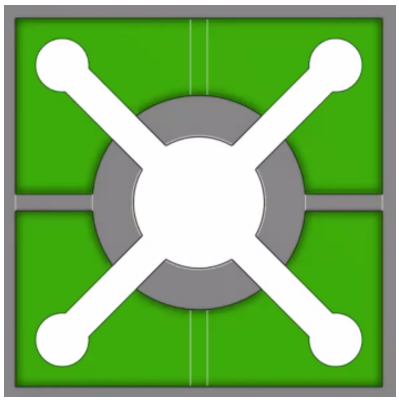

Figure 14A: *Current geometry of ESQ in warp*

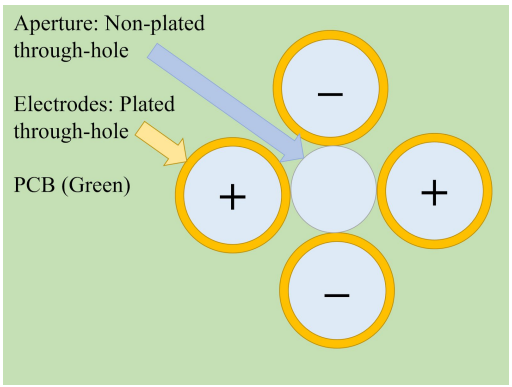

Figure 14B: *New geometry in warp*

IV. DISCUSSION

In terms of differences between WARP compared to Python and Java simulations, WARP can be a more realistic representation because it explores space charge much better and models non-uniform fields. Also WARP computes edge effects, while the KV equations assume hard edge fields.

Although the simulations may match with each other, there will likely be further deviation when compared with the experimental result. This can be because ESQs normally have a thickness that is long compared to their radius. Those used in this experiment however are unusual, shorter and approximately equal to radius, which can cause fringe fields.

Further differences between simulations and experiential results will be due to the initial conditions not being the same for multiple beamlets in the 10×10 array. There will not be uniformity in the beamlets, the current will be lower in the beamlets on the outside, the variation is because the plasma is not uniform across such a large surface. Therefore it is desired to find a solution that can work for a range of currents to take care of the variation in matching.

Another difference could arise because in the simulations, most particles are contained within the beam envelope, but in experiments there can be particles outside this. The beam can lose this halo of particles that are beyond the beam envelope. If the particles hit metal, depending on angle and energy, they can be deflected or go into material. This can cause displacement of atoms, damage the material or cause it to become brittle, and can kick out electrons which will bounce around and heat up material. Since the wafers in this experiment are plastic, the PCB can heat up and melt if a lot of the beam hits it.

V. FUTURE WORK

A useful addition to the Python simulations would be to add an interface with interactive inputs similar to that of the Java simulations. This could be done using matplotlib, plotly, or streamlit. It could involve options to easily adjust ESQ voltages or initial beam conditions, and quickly seeing the results. These inputs can currently be done using Python and manually changing the parameters and resolving the KV equations, but it is not as smooth an experience as it could be. The user interface was not added due to time constraints, but would be beneficial for future use.

Another improvement would be work with the Kapchinsky ENVelope (KENV) project [8]. This would be useful as it provides straightforward visualizations, an interactive user interface, and a genetic algorithm for adjusting the beam envelopes. The challenge to this is that it was designed for electron beams, and would need to be adapted to work with positive ions. Upon inquiry, the creators of this said it would be possible to use the charge to mass ratio parameter for implementing ions. One would need to rescale the electric and magnetic fields

according to the q/m ratio, and decrease current by a factor of (e/q)(m/m_e), where q is the charge of the ion, e is the charge of an electron, m is the mass of the ion, and m_e is the mass of an electron. This could be worthwhile, as beam parameters could change in the future, which would require manually finding another matched solution each time. This program would allow for adaptability, selection, and mutation, having a framework for finding future matching section solutions.

Additional work can be done experimentally to confirm the results of these simulations. One option is to add a couple ESQs to the accelerator before the RF section. Then measurements could be taken by moving a Faraday cup to obtain the beam profile, and see if the beam voltages match what was expected based on the simulated results. Variables to consider would be how many ESQs to add and how far away to put the Faraday cup. Another option is to simply measure the beam current using a Faraday cup before and after the addition of the ESQ wafers in the matching section. If the current increases then it can be assumed that the simulations were valid and the solution is beneficial. Additionally, to experimentally confirm results from simulations, the beam profile can be examined by doing a knife edge scan. A final test should be done experimentally on the ESQ boards for breakdown. This can be completed by placing a wafer in a vacuum chamber, and seeing how high voltage can go to determine the breakdown voltage.

VI. CONCLUSION

A compact linear accelerator was designed that uses radio frequency acceleration units and electrostatic quadrupoles as focusing elements. The matching section for this accelerator transforms the beam from a symmetric cylindrical diverging beam to that of a matched beam of an alternating gradient periodic lattice. The main goal and result of this project was a design of the matching section, including the positions and voltages of the ESQs. This was completed through the use of 1D Python and Java simulations, as well as with 3D simulations using WARP. Future experiments can be done with the accelerator to verify the results of these simulations.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internship (SULI) program.

REFERENCES

[1] Vinayakumar, K. B., Ardanuc, S., Ji, Q., Persaud, A., Seidl, P., Schenkel, T., & Lal, A. (2019). Demonstration of waferscale voltage amplifier and electrostatic quadrupole focusing array for compact linear accelerators. Journal of Applied Physics, 125(19), 194901.

[2] Seidl, P. A., Persaud, A., Ghiorso, W., Ji, Q., Waldron, W. L., Lal, A., ... & Schenkel, T. (2018). Source-to-accelerator quadrupole matching section for a compact linear accelerator. *Review of Scientific Instruments*, *89*(5), 053302.

[3] Siebenlist, F., Thomae, R. W., Van Amersfoort, P. W., Schonewille, F. G., Granneman, E. H. A., Klein, H., ... & Weis, T. (1987). Matching of a cylindrical ion beam to a periodic quadrupole channel. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *256*(2), 207-218.

[4] NumPy: The fundamental package for scientific computing with Python. https://numpy.org/

[5] Pandas: a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. https://pandas.pydata.org/

[6] Cufflinks: This library binds the power of plotly with the flexibility of pandas for easy plotting. https://pypi.org/project/cufflinks/

[7] Scipy.integrate.odeint. https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html

[8] Fuodorov, V., Petrenko, A., & Nikiforov, D. (2019). Fuodorov/kenv. Retrieved August 10, 2020, from https://github.com/fuodorov/kenv