A Novel Optimization Platform and Its Applications to the TRIUMF Energy Recovery Linac

by

Chris Gong

B.Sc., The University of British Columbia, 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate and Postdoctoral Studies

(Physics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

October 2015

Abstract

A novel software platform for global optimization was developed to create a baseline design for the TRIUMF Energy Recovery Linac (ERL). The platform is parallel capable, scalable, and allows flexible combinations of various accelerator tracking tools such as Madx and Free Electron Laser (FEL) tools such as Genesis. The TRIUMF machine includes simultaneously a two-pass ERL and a rare isotope line. Many parameters are coupled, including RF and the separator system which are shared for all three linac passes. The global optimization platform can study dynamic relationships between different processes, a practice not easily performed with piecewise optimization. The FEL induced energy spread, which grows by an order of magnitude after deceleration and increases the difficulty of beam disposal, creates a tradeoff, or Pareto front, between the gain and the dump energy spread. Another front forms between energy recovery and final energy spread due to RF settings. The Pareto fronts give insights on how objectives are related and the repercussions of design decisions. Pareto relationships are presented, along with potential lattice solutions found by the optimization platform.

Chris Gong

Preface

This dissertation is based on the creation of a computational apparatus and resulting data of the TRIUMF Energy Recovery Linac optimization. The work is the effort of myself and Y.C. Chao.

The computational apparatus is a multiobjective optimizer that works in a massively parallel environment (WestGrid), and can integrate and link multiple modeling tools and codes, i.e. engines, for a flexible method of creating global models of particle accelerators. All code for linking different engines together are performed by the optimizer without need for user intervention.

The majority of the platform codebase deals with handling multiple engines, transition between engines, and distribution of work to massively parallel computing clusters. These portions of the code are wholly designed and implemented by myself. The genetic optimization algorithm used is an implementation of SPEA2 by I. Bazarov. The code is refactored to suit the needs of the platform. The fundamental algorithm is unchanged.

The modeling engine used to simulate the accelerating cavities is the Empirical Model (EM). The idea for EM was proposed by Y.C. Chao. The EM code was wholly written by myself, originally in Java for TRIUMF high level applications development. It was ported to C++, also by myself, for optimization purposes. Interpolation tables used by EM were extracted by Y.C. Chao.

The setup of the optimization problem involving 62 free parameters, 13 objectives, and 3 constraints, presented in chapters 3 and 4 are my own work, with guidance and discussions with Y.C. Chao.

The optimization results and analysis presented in chapter 5, as well as the analysis of the accelerator baseline solution presented in chapter 6, are my original work. This includes studying the tradeoffs between different machine parameters and presenting an accelerator baseline solution complete with layout coordinates of all optical components in TRIUMF standard format.

All images and tables used in this dissertation were produced by myself.

Publications arising from the work presented in this dissertation:

C. Gong and Y.C. Chao, "A New Platform for Global Optimization," IPAC12, New Orleans, 2012.

C. Gong and Y.C. Chao, "The TRIUMF Optimization Platform and Application to the E-Linac Injector," ICAP12, Rostock, 2012.

Table of Contents

Appendices

List of Tables

List of Figures

List of Programs

Acknowledgements

I would like to thank the members of my doctorate committee, Yu-Chiu Chao, Tom Mattison, Rick Baartman, Reiner Kruecken, and Mike Craddock, for providing valuable guidance during the course of my studies.

This work benefited from insightful correspondences with Sven Reiche, Gabe Marcus, Michelle Shinn, and members of the TRIUMF Beam Dynamics Group and Accelerator Division.

Thanks to the members of the UBC Physics Department and of TRIUMF for providing a supportive environment for this work to take place.

Lastly, a tremendous thank you to my supervisor Yu-Chiu Chao, whose guidance was instrumental.

Dedication

This work is dedicated to my parents for their continued support, and to Catherine Kwan for her seemingly endless patience and empathy.

Chapter 1

Introduction

The goals of this research are:

1. Design and implement a novel software optimization platform, capable of multiobjective optimization in massively parallel computing systems.

2. Use the platform to study the underlying physics of the TRIUMF Energy Recovery Linac (ERL).

3. Use the platform to find a baseline solution to the ERL, including RF settings and a complete coordinates layout of all optical components.

The ERL is a future upgrade to the Electron linear accelerator (E-linac), a machine currently under development at TRIUMF. The questions of concern are, for a given ERL machine layout where attributes such as drift lengths and quad gradients are variables, does a transport solution exist which has good lasing, energy recovery, and beam disposal? How does such a machine work? What are the relationships between RF, gain, and beam transport? This matter is complicated by the fact that currently, no simulation software has the capacity to model all the relevant physics of an ERL from start to end, therefore performing global optimization on such a machine is very difficult.

The E-linac is the accelerator at the center of the Advanced Rare IsotopE Laboratory (ARIEL), a TRIUMF facility designed to advance Canada's capabilities in science and technology. The E-linac is a 0.5 MW, 10 mA, continuous wave (CW) rare isotope beam (RIB) driver for the photofission of actinide targets, with emphasis on neutron-rich isotopes. The linac shares commonalities with Energy Recovery Linac (ERL) designs and can be upgraded to fulfill such a purpose. Further descriptions of ARIEL and ERLs can be found in chapter 2.

The optimization problem can be defined as follows: what is the underlying physics of the TRIUMF ERL? Given a minimalist layout of accelerator optical elements, can a good transport solution be found (or exist), and

can this solution satisfy other functions of the ERL, namely, maximize lasing and energy recovery, and coexist with existing RIB operations? What quadrupole strengths and positions, and RF parameters are required for the aforementioned solution?

The study and design of accelerators is typically done with piecewise optimization. For example, find the optimal beam conditions to maximize lasing in the Free Electron Laser (FEL), then design the arcs to match those conditions, and then produce RF conditions necessary for the arc designs. This type of piecewise, or local optimization, does not explore the global solution space or show tradeoffs between important parameters. For example, is it acceptable if lasing is reduced slightly but has a large benefit for energy recovery and beam disposal? This piecewise scheme arose from necessity and practicality. No accelerator modeling tool currently has the capabilities to model every physical process contained in an ERL. The software for modeling the FEL cannot model the transport arcs or accelerating cavities, and vice versa. At the same time, attempting to perform global optimization by exploring a "grid" of different local optimizations quickly fails for large problems due to the large number of permutations needed. If only two free parameters exist, we can scan them, but if a large number of parameters exists, scanning is impractical. The ERL optimization contains over 60 free parameters. An exhaustive scan is unfeasible.

We solve this problem by creating a software platform for optimization which can model the global ERL, and can make use of the many modeling tools available to model any physics necessary. The software architectural design of the optimization platform is provided in Appendix B. A summary of the platform's major functionalities:

1. Able to perform global ERL optimization. We are primarily interested in modeling the ERL from acceleration, to lasing, to deceleration, to beam disposal. Although designed with the ERL in mind, the platform is generic and supports many optimization problems from outside accelerator physics. It has been used in several other optimization problems, including the E-linac injector [41], CSR [39], and the VECC injector linac [33]. Global optimization also allows us to see tradeoffs and dynamic relationships of the machine.

2. Interface with different modeling tools, or *engines*. The platform can call MADX [20] for arc design, GENESIS [65] for lasing, and automatically transfer values from one to the other for continuous modeling, e.g. use the MADX output beam parameters as the input beam parameters for GENESIS. This mitigates the need for local optimizations

or permutations. The engines can be combined in serial, parallel, or any combination thereof. This defines an optimization problem *topology*, with each modeled section being a *vertex*. The ERL topology can be seen in 4.2.

3. New modeling engines are easily added to the framework if necessary. This increases the flexibility of the platform and its adaptability to new problems.

4. Parallel capable - reduces running time by taking advantage of computational clusters. The platform is designed to take advantage of Canada's WestGrid [8] and similar batch systems.

5. Exception handling - by allowing different engines, a multitude of errors can occur, including engine hang-ups that can destroy a time-consuming run. Dealing with large parallel jobs can also produce networking and file system issues. The optimization platform is smart enough to handle the multitude of exceptions that can occur. This required significant investments in good software design.

6. The platform uses a flexible XML input scheme to define the list of objectives, constraints, and parameters, as well as the optimization topology. The scheme is generic and not tailored toward any one modeling engine.

7. Allows the user to inject code for data manipulation and post processing.

Designing the platform to operate with arbitrary engines with multithreading capabilities was a significant software challenge. Multithreading and parallel processing especially increased the complexity of the software design due to concurrency issues and threading exceptions.

The optimization platform uses the genetic algorithm, a class of multiobjective optimization algorithms. The advantages of genetic algorithms are:

- Multiobjective - the ERL problem involves several very different objectives: maximize lasing, optimal energy recovery, beam disposal, and beam transport. A multiobjective algorithm allows us to see the tradeoffs, or *Pareto fronts*, between competing objectives. Optimizing using a single objective algorithm requires the different objectives to be combined into a single objective, in an arbitrary way.

- Can find multiple solutions - this is critical in a multiobjective problem, as one solution may not be optimal in all dimensions. An ERL that provides the best gain may not have the best energy recovery. Multiple solutions can form Pareto fronts, which allow us to compare different options and see the tradeoffs.
- Global - genetic algorithms do not require specific search space geometries or convexity. Note: the platform is "global" in two senses. It allows the global modeling of the ERL, and the optimization algorithm can search the global search space.
- No gradients required.
- No initial search point required.

Mathematical definitions of optimization and Pareto fronts can be found in section 1. An example population with the Pareto front defined is shown in fig. 1.1.

The particular variant of the genetic algorithm used is the Strength Pareto Evolutionary Algorithm (SPEA2) [74]. The algorithm works as follows:

1. Create a set of ERL designs. Each ERL design has a random combination of parameters, such as quad gradients and RF phases. This is the initial *population*.
2. Each design in the population, or *individual*, is evaluated on how well they satisfy the design objectives and constraints (*fitness*). The worst designs are thrown out.
3. New designs are created by tweaking the best of the old designs (*evolution*), and added to the population to replace the ones thrown away. This is the new *generation*.
4. Go to step 2, until a predetermined number of generations have passed.

SPEA2 performed well in a comparison of genetic algorithms and justifies this choice [75]. Genetic algorithms are also comparable in efficacy to Particle Swarm Optimization, another popular class of population based search algorithms [42, 46, 71]. More information on SPEA2 and genetic algorithms can be found in section 1.

The implementation of the TRIUMF optimization platform follows the Platform and Programming Language Independent Interface for Search Algorithms (PISA [36]) concept which, for flexibility purposes, divides the

platform into two portions, Variator, responsible for evolution, and Evaluator, responsible for fitness testing, i.e. how well does a solution satisfy the objectives. Other precursors to the TRIUMF optimization platform are Alternate PISA (APISA [13, 14]) written by I. Bazarov and Yet Another PISA (YAPISA [56]) by G. Goh. The TRIUMF platform is the first to allow for multiple simulation engines in the same optimization setup, and extending the software to allow for parallel computing was a challenge and step forward.

More details of the software aspect of the optimization platform are in Appendix B. For the application of the platform to the TRIUMF ERL see chapters 3 and 4. ERL optimization results can be found in chapters 5 and 6.

A list of notations can be found in Appendix A.

Multiobjective Optimization

We now provide the mathematical formulation of a multiobjective optimization problem. A generic optimization problem can be stated as, given some optimization parameters (or free parameters)

$$\begin{aligned} & x = (x_i) \\ & \text{s.t. } L_i \leq x_i \leq U_i \end{aligned} \tag{1.1}$$

where constants L_i and U_i are the lower and upper bounds on x_i and $i = 1, ..., N$, the goal is to optimize one or more objective functions

$$\min\ g_j(x) \tag{1.2}$$

where $j = 1, ..., P$. At least one objective is required. The system is subject to multiple constraints

$$F_k(x) \leq C_k \tag{1.3}$$

where $k = 1, ..., Q$ and C_k are constants. Optimizing with zero constraints is allowed.

The space of all x which satisfies the constraints is the search space. We do not assume the search space is convex (for definition of convexity, see [16]), hence our choice of the genetic algorithm, a pseudo-random search method. In theory, if given enough time, this algorithm will always find the optimum solutions. Note that the algorithm SPEA2 can generate individuals that do not satisfy all constraints, but these points are heavily biased against in fitness selection and should disappear in later iterations.

A point S_1 in search space is *dominant* over point S_2 if for all objective functions g_j

$$\forall g_j(S_1) \leq g_j(S_2) \wedge \exists g_j(S_1) < g_j(S_2) \tag{1.4}$$

and S_1 is *nondominant* over S_2 if

$$\exists g_j(S_1) > g_j(S_2). \tag{1.5}$$

Solutions to the problem 1.2 is defined as the *Pareto front*. A point is in the Pareto front if it is not dominated by any other point. The goal of optimization is to find the Pareto front.

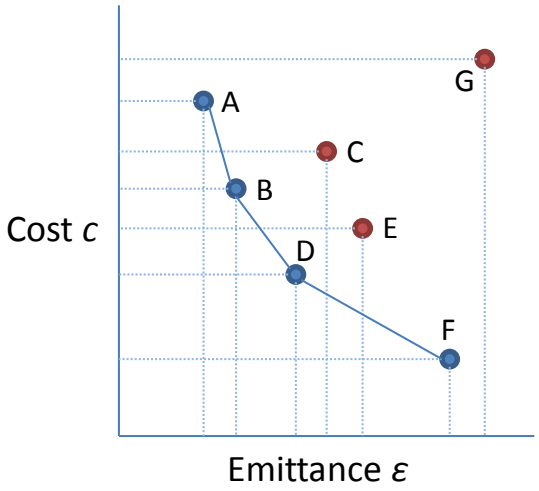

Figure 1.1: Example population in objective space. Each blue point is part of the Pareto front because they are not dominated by any other points.

An example of a Pareto front is shown in fig. 1.1. Given a problem with two objectives: 1) minimize the machine cost c and 2) minimize the beam emittance ε, the optimization platform generates a population of different machines. The genetic algorithm ranks the fitness of each machine based on Pareto dominance, shown in table 1.1. We define the fitness of the machine by how many other machines dominate it (the lower the fitness score, the more fit it is).

Table 1.1: Pareto dominance of points shown in fig. 1.1. The definitions of dominance and nondominance are given by eqns. 1.4 and 1.5. The fitness of the machine is equal to how many machines dominate it. The lower the fitness score, the more fit that machine is.

Machine	**Dominated by**	**Fitness score**
A	-	0
B	-	0
C	B,D	2
D	-	0
E	D	1
F	-	0
G	A,B,C,D,E,F	6

Points not dominated by any other points form the Pareto front, e.g. A-B-D-F. Each iteration, the algorithm generates new machines which can achieve a better front. An example is shown in fig. 1.2. A new machine, H, is created by the optimization platform, forming a new front A-H-F.

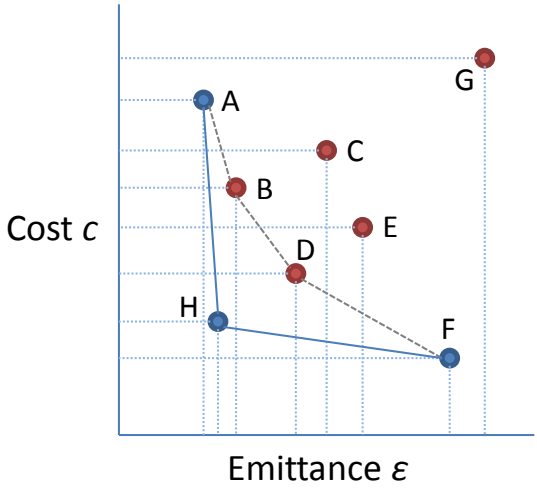

Figure 1.2: Example population in objective space. New machines are generated which can change the Pareto front.

The dominance shown in table 1.1 is also used by the genetic algorithm as weights. The most dominated machine, G, is the most likely to be tossed

away, whereas machines on the Pareto front are the most likely to be kept and used as parents for new machines.

Chapter 2

Description of the TRIUMF E-Linac and ERL

The TRIUMF E-linac, currently under construction, is a driver for producing rare isotope beams (RIB) via photofission of actinide targets [51]. The emphasis is on producing neutron-rich isotopes for studies on nuclear structure physics and material properties.

The intention is to upgrade the E-linac to an energy recovery linac (ERL) and use the E-linac as the driver of an infrared light source. This requires the addition of a recirculation lattice, a free electron laser (FEL), and a new ERL gun to the main linac. The operation is as follows (refer to fig. 2.1):

1. Beam from the ERL injector is sent into the main linac and accelerated from 7.5 MeV to 45 MeV.

2. ERL beam transported to the recirculating loop via RF separator.

3. The loop is divided into two arcs, each introducing a 180° bend to the beam. The first arc turns the beam antiparallel to the main linac.

4. The beam travels through the Free Electron Laser (FEL), where electron energy is converted to coherent radiation via electron-light interaction (fig. 2.2).

5. After lasing, the beam goes through the second arc where it is brought back to the main linac out-of-phase for deceleration back to 7.5 MeV. Beam energy is returned to the RF system.

6. The decelerated beam finally is dumped after the second linac pass.

The TRIUMF ERL is designed to be a dual RIB and ERL machine. The two beams are produced in two separate guns but share the injector and linac transport. RIB operates at 650 MHz and fills every second bucket of the 1300 MHz RF system. The ERL beam will occupy unused RF buckets

(RF periods unoccupied by RIB), therefore does not lower RIB performance and allows for simultaneous operation.

A complete overview of ARIEL and the E-linac can be found at [51].

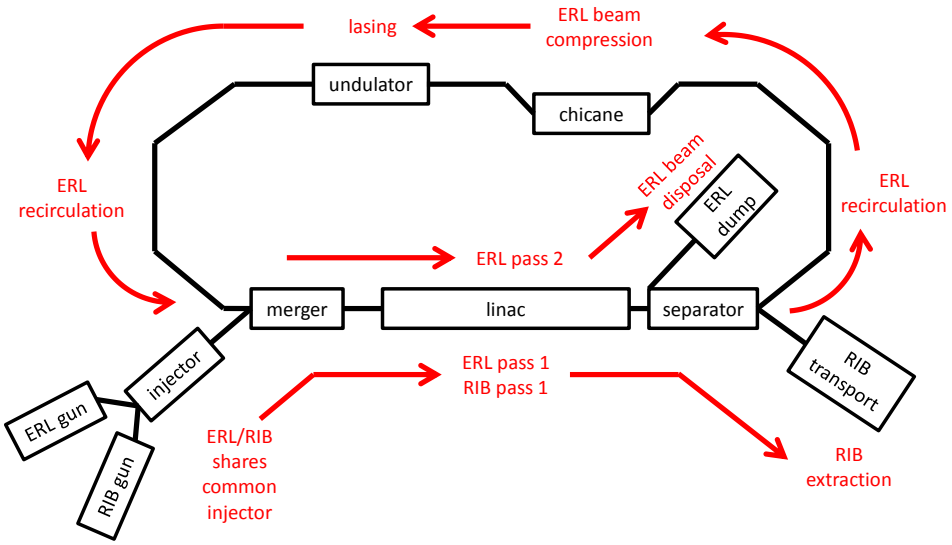

Figure 2.1: Overview of the TRIUMF dual ERL/RIB.

Principle of Free Electron Laser

A Free Electron Laser (FEL) is a high powered light source, opening the door to many diffraction and microscopy experiments [15, 43, 51].

The FEL converts beam energy into coherent radiation. The device consists of an optical cavity formed from two mirrors surrounding an undulator, a series of alternating dipole magnets. Electrons travel in a sinusoidal trajectory, producing initial light via synchrotron radiation. The light is trapped in the optical cavity. Past this initial seeding, further radiation is produced via electron-light interactions [47].

Due to the close proximity of the undulator dipoles, they are constructed from permanent magnets [32]. The E-linac undulator is designed as a planar undulator.

The fundamental laser wavelength is given by the undulator equation [17, 47]

$$\lambda = \frac{\lambda_u}{2\gamma_r^2}\left(1 + K^2\right) \tag{2.1}$$

Where λ_u is the spacing or period of the undulator magnets, γ is the Lorentz factor of the bunch centroid, and K is the dimensionless undulator parameter, dependent on the dipole magnetic field and spacing. K have typical values on the order of unity [54]. Taking $\lambda_u = 4$ cm and $K = 0.7$, estimated from machines of similar energy [30], we find λ to be several microns, in the short- to mid-wavelength infrared region. It is theoretically easy to operate the FEL up to several hundred microns in the far-infrared region, by tuning the electron beam energy or K, which can be tuned by changing the magnetic gap size [61].

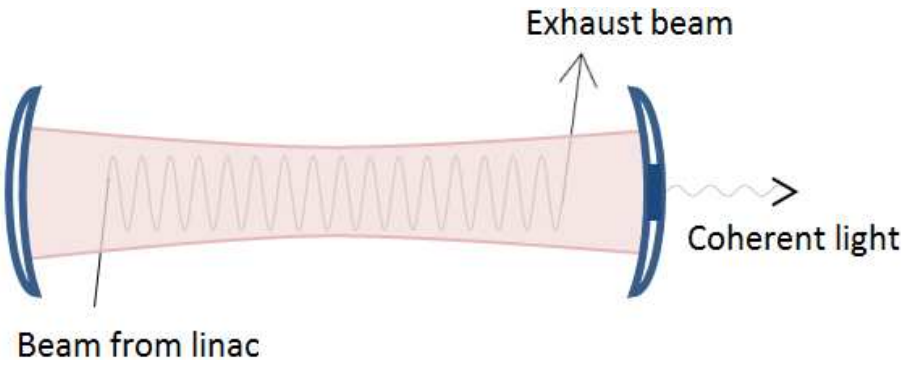

Figure 2.2: An FEL-oscillator converts beam energy into laser energy. The beam travels in a sinusoidal path in the undulator, a series of alternating dipole magnets. The energy is captured in a cavity by two high-reflectivity mirrors and increases until saturation. A small amount of power is outcoupled for useful purposes.

These parameters are suitable for an FEL of the low gain oscillator type, i.e. using synchrotron radiation to induce lasing and having a mirror system to capture the light while allowing a small amount through. Mirrors of high-reflectivity are relatively easy to design for infrared light. Using an optical cavity to store energy negates the need for a very long undulator, which is required for SASE [17], an alternative FEL with a one-pass optical system. A short undulator also causes less beam distortions, making the beam suitable for energy recovery.

Physical Requirements

At the writing of this document, the scientific requirements are not well-defined; therefore parameters of both the FEL and ERL can only be ap-

proximated. The undulator period and number of periods were chosen to be 4 cm and 25, respectively, for a total undulator length of 1 m. These values were based off the Peking University FEL design [30], a machine with similar beam energy and parameters to the E-linac.

Principle of Energy Recovery

The recirculation lattice brings the ERL beam back into the linac for a second pass. The path length of the lattice is adjusted such that the beam arrives for the second pass at a decelerating RF phase, i.e. 180° offset from the accelerating pass, where the beam energy is reduced to the initial injection energy. The advantages of energy recovery are 1) the decelerated beam is easy to dispose of and 2) the beam energy is transferred back to the linac, which intensifies the RF field for acceleration of further bunches. The net beam loading in an ERL is close to zero, thus consuming only a modest amount of RF power [59].

Physical Requirements

The ERL adds a recirculation lattice and a second gun to the E-linac. This high brightness gun shares the injection cryomodule and main linac with the photofission beam. The gun is projected to operate at 130 MHz, as opposed to the 650 MHz RIB, for a RIB to ERL beam ratio of 5:1. The ERL beam occupies empty RF buckets, so does not interfere with RIB operation or require RIB to operate sub-optimally. The bunch charge is 100 pC, resulting in 10 mA beam current and 0.5 MW power.

The injector transport, merger, main linac, and separator systems are shared by RIB and ERL operations. The separation system consists of an RF separator, followed by a drift and ending with a septum. The recirculation lattice begins at the septum exit, turns the beam 180°, goes through the FEL, followed by another 180° arc, for a full 360° turn back into the merger and the main linac for deceleration. The merger system and the main linac accommodate three beams: ERL injected beam, ERL recirculated beam, and RIB injected beam. Refer to fig. 4.1 for the machine layout.

Beam loading vector prefers a deceleration phase exactly 180° offset from the acceleration phase [38, 51]. With no beam loading, the optimal Q-value $Q_{L,opt}$ for an ERL is $Q_{L,opt} = f/(2\Delta f)$ [53], where f is the cavity fundamental mode and Δf is the cavity detuning driven primarily by microphonics. When the ERL is operated in 180° mode, the machine operates as a high Q machine with $Q_L = Q_{L,opt}$ [63]. The two beam current vectors cancel

and the RF system resembles one with no beam loading. The JLab IRFEL energy recovery demonstration shows that RF power was reduced from 36 kW with no energy recovery, to 16 kW with energy recovery, at 1.1 mA beam current [58].

Design Tradeoffs

The TRIUMF ERL contains many design challenges. Some items of interest are, but not limited to,

1. How to design the bunch compression system? How do the RF and arc transport parameters play off against each and the undulator gain?
2. How does the arc time-of-flight (TOF) impact the RF deceleration phase and energy recovery?
3. How does the RF deceleration phase impact energy spread?
4. Can lasing cause issues leading to beam loss?

Results from optimization detailing tradeoff issues can be found in chapter 5.

Chapter 3

Description of the Global Optimization Platform and Engines Used for ERL Modeling

The software architecture of the platform follows the PISA interface [36]. The platform is divided into two parts: Variator and Selector.

Variator handles the creation and evaluation of new ERL designs in the population. The evaluation of each ERL design is broken into sections, with each section being handled by a modeling engine. Each section is assigned to a worker node in a parallel environment for execution. Variator handles all parallel computing work assignments and ensures work is performed smoothly through a series of process control and exception handling mechanisms.

Selector handles the optimization using the genetic algorithm SPEA2 [74]. Given an optimization population, Selector tests for Pareto dominance and stochastically chooses ERL designs to use as parents for the next iteration. Steps taken by the algorithm was outlined previously in chapter 1.

A complete description of the design and implementation of the optimization platform is provided in Appendix B.

Connecting Different Engines for Global Modeling

Each engine used in the ERL optimization (detailed below) is treated by the optimization platform as independent. They are modular in that no engine, or machine section, need to explicitly know about any other. The C++ code of the platform automatically creates the necessary input files for these engines to run and read the outputs from them.

Transitions between engines are handled by the optimization platform.

For instance, MADX and the Empirical Model (EM) are often used in serial. EM is used to model a cavity, followed by a transport section modeled by MADX. The transitioning is quite involved. For example, units need to be converted (EM uses MeV, MADX uses GeV). EM uses the beam size, which needs to be converted to Courant-Snyder parameters for MADX. This complicated process is hidden from the user. The platform automatically performs unit conversions, parameter translations, and other niceties.

For a detailed explanation of how the engines are incorporated into the platform and how to define the engine landscape for an optimization problem, see Appendix B.

Engines Used for Modeling

An *engine* in the context of optimization refers to a modeling software. Different engines are used to model different sections of the machine, allowing us to take advantage of the features of each engine. The different engines used for optimization are listed in this section.

MADX

MADX [20] is an accelerator design code from CERN. The MADX Twiss module is a convenient method to retrieve beam parameters and transport map elements of the lattice. However, MADX does not have good capabilities in tracking longitudinal parameters and Twiss calculation is 4D only. Sections that involve non-RF elements are modeled by DIMAD and MADX in parallel, which mitigates each engine's disadvantages. The input files of both engines are similar, and can be used to cross-check the validity of the output.

DIMAD

DIMAD v2.9 [9] is a tracking code with a similar input format to MADX. DIMAD has many modules, providing easy and flexible access to beam information. (for example, beam size is readily available in the BEAM module, including contributions from betatron oscillations, dispersion, and higher order terms). Both envelope and particle tracking are used. DIMAD's MATRIX module is useful for displaying second order transfer map elements for an entire section, whereas MADX requires manual concatenation of map elements for individual optical devices.

A drawback of DIMAD is that the floating point output behaviour is often unpredictable. Output of the MACHINE command may replace floating point values with the string ********** if the floating point requires more than ten significant digits to display. The output can also run-on floating point values, i.e. no delimiters, making automatic text parsing difficult or impossible.

Genesis

Genesis v1.3 [65] is an industry standard for simulating undulator radiation. In the ERL optimization Genesis is used to produce the gain of the FEL at the resonance wavelength. Genesis requires an input wavelength λ. Beam parameters can shift the resonance, therefore the gain is not optimal if λ is off-resonance. We use Genesis' scan feature to look for λ which maximizes gain. The scan feature is compatible only with running Genesis in steady-state mode. The bunch length is $\sim$ 100 times longer than the radiation wavelength (σ_z =100 µm vs λ =1 µm), therefore the steady-state mode is justified.

The beam envelope is calculated from the distribution from Genesis. The beam coordinates x, x', y, y', δ are translated directly from the Genesis distribution. The coordinate z is not easily obtained due the difficulty in translating from the ponderomotive phase θ (see Appendix A for definition). Genesis models the bunch as a series of slices, each with a radiation wavelength thickness. Transition between slices is not modeled [57] and this complicates the translation from θ to z. We choose to keep z constant throughout the undulator and the justification is as follows.

The bunch length after the undulator is assumed to be identical to the initial bunch length. This can be justified by looking at the longitudinal phase space evolution. For a single particle in the distribution, expand to first order the phase θ and $\xi \equiv \gamma/\gamma_r - 1$, the energy deviation from the bunch mean energy γ_r [47]:

$$\begin{aligned} \theta(r) &= \theta_0 + \epsilon\theta_1 \\ &= \theta_0 + \frac{\epsilon}{\xi_0}\left(\frac{\sin\theta_0 - \sin\phi_0}{2k_u\xi_0} - r\cos\phi_0\right) \\ \theta_0(r) &= 2k_u\xi_0 r + \phi_0 \end{aligned} \tag{3.1}$$

where r is the position of the bunch center within the undulator and ϕ_0 is

the initial phase. The expansion parameter is given by

$$\begin{aligned} \epsilon &= \frac{eE_0K[JJ]}{2\gamma_r^2 mc^2} \\ [JJ] &= J_0\left(\frac{K^2}{4+2K^2}\right) - J_1\left(\frac{K^2}{4+2K^2}\right) \end{aligned} \tag{3.2}$$

where the undulator parameter $K = 0.7$. The radiation field E_0 can be estimated from the energy density S and beam cross sectional area A

$$E_0 = \sqrt{Sc\mu_0} = \sqrt{\frac{Pc\mu_0}{A}} = \sqrt{\frac{Pc\mu_0}{\pi\sigma_x\sigma_y}} = 60 \text{ MV/m} \tag{3.3}$$

With $P = 8$ MW, $\sigma_x = \sigma_y = 0.5$ mm, this leads to $\epsilon = 5\times10^{-3}$ m^{-1}. For a large energy spread of 0.01, the change in phase $\Delta\theta = \theta(r = 1 \text{ m}) - \theta(r = 0 \text{ m})$ is 2.6 at $\phi_0 = 0$, 2.8 at $\phi_0 = \pi/2$, 3.6 at $\phi_0 = \pi$, and 2.6 at $\phi_0 = 2\pi$. Converting θ to z using the conversion factor of $\lambda/2\pi$, the change in bunch length is several radiation wavelengths, even less at the beam interior. Given that the bunch occupies several 100 radiation wavelengths, particles within the distribution experience very little longitudinal movement, therefore changes in z are considered negligible. Under this assumption, the envelope parameters evolve in the undulator as

$$\begin{aligned} \sigma_{z,+} &= \sigma_z \\ \sigma_{\delta,+} &= B\sigma_\delta \\ V_{z\delta,+} &= BV_{z\delta} \\ \varepsilon_{z,+} &= B\varepsilon_z \end{aligned} \tag{3.4}$$

where the (+) denotes the end of the undulator, B is a scaling constant determined from the Genesis output distribution, σ_z is the bunch length, σ_δ is the bunch energy spread, $V_{z\delta}$ is the covariance between the coordinates z and δ, and ε is the emittance.

A distribution is created from the genesis output and tracked in arc 2. Momentum tail is a concern and can result in particle loss in the arc dipoles. We create a distribution using the momentum profile of one Genesis slice. Fig. 3.1 shows that the momentum profile varies little across simulation slices, justifying this approach.

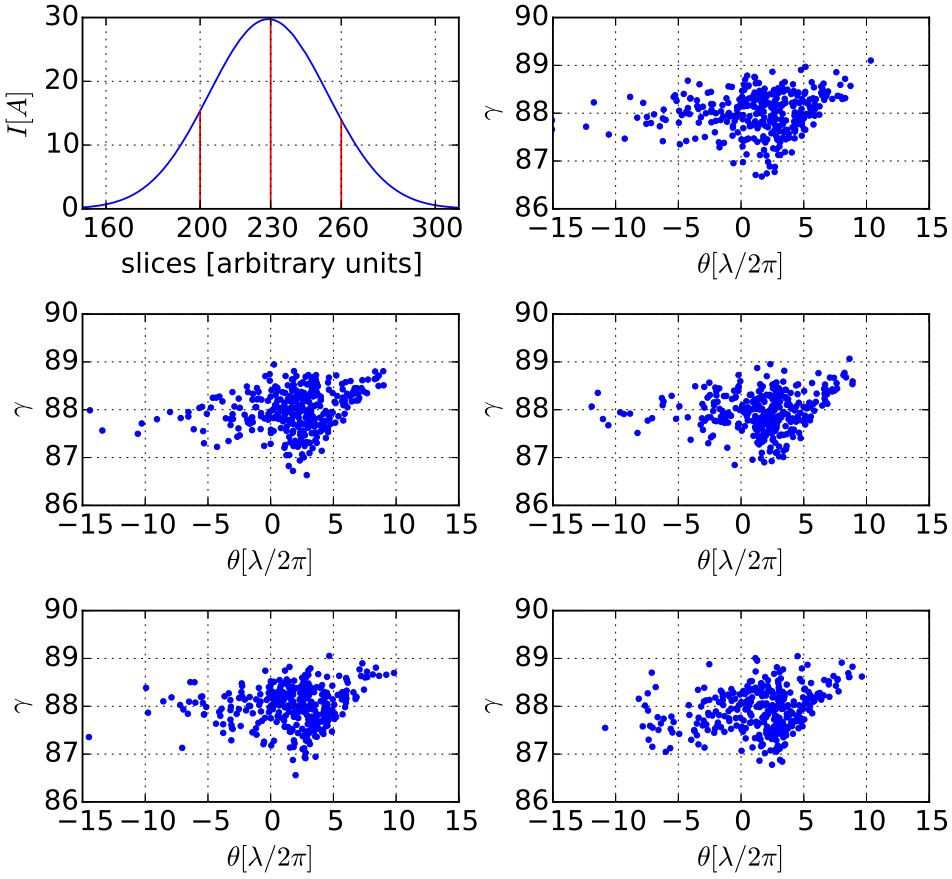

Figure 3.1: Top left: Genesis simulation slices vs the current in each slice, in time-dependent mode, at the undulator exit. The five following plots (left to right, top to bottom) are the longitudinal profile at the five vertical red lines. The phase spaces undergo near identical distortions in the undulator, thus justifying using the steady-state mode in Genesis.

Genesis outputs a particle distribution in binary format. In order to track the distribution through the return arc, the binary was converted into ASCII via the Python script 3.1.

Empirical Model

At TRIUMF, tracking the electron beam through RF devices is typically done using ASTRA [37]. The Empirical Model (EM) was written (by the present author) as a replacement to ASTRA, and works by using transfer

Program 3.1 Python program that converts the Genesis output distribution from binary to ASCII.

```
    data = [] # each row of 'data' is one slice
    buf = NCOLS*NPARTS # number of values for each slice

    # Read all data for a slice at a time,
    # append each to the array.
    # Each value is 8 bytes.
    # The final array has dimensions [NSLICES,buf]
    with open(filename,mode='r') as fp:
for d in iter(lambda: fp.read(8*buf),''):
            c = struct.unpack('d'*buf,d)
            data.append(c)

    # transform to numpy matrix
    data = numpy.array(data)
    NSLICES = len(data)

    slicedata = data[slice_to_read-1,:]

    # NOTE: np.reshape differs from Matlab reshape.
    # Suppose x=1...10:
    # np.reshape(x,[2,5]) returns
    #  [[1 2 3 4 5]
    #   [6 7 8 9 10]]
    # Matlab reshape(x,2,5) returns
    #  1 3 5 7 9
    #  2 4 6 8 10
    slicedata2= numpy.reshape(slicedata,[NCOLS,NPARTS])

    gamma = slicedata2[0,:]
    phase = slicedata2[1,:] # ponderomotive phase
    x     = slicedata2[2,:]
    y     = slicedata2[3,:]
    px    = slicedata2[4,:] # gamma*beta
    py    = slicedata2[5,:] # gamma*beta
```

maps pre-generated from ASTRA data [27, 28, 31, 40]. EM uses particle tracking; each particle is propagated using map elements interpolated from particle momentum, RF phase, and RF amplitude.

The recirculation time-of-flight (TOF) is important for ERL phase matching between the two linac passes. EM tracks using time instead of position, thus the TOF through the cavity is easily obtainable. For devices with energy changes, such as cavities, calculating the TOF is tricky because it cannot be easily inferred from the path length and velocity, thus making EM an important component in the model.

EM was created with the intention of having ASTRA's accuracy, but with running time suitable for online tracking in TRIUMF high level applications using Java [10]. The low running time also makes EM suitable for global optimization and a C++ port was made with such a purpose. Fig. 3.2 shows the validation of EM tracking results compared against ASTRA.

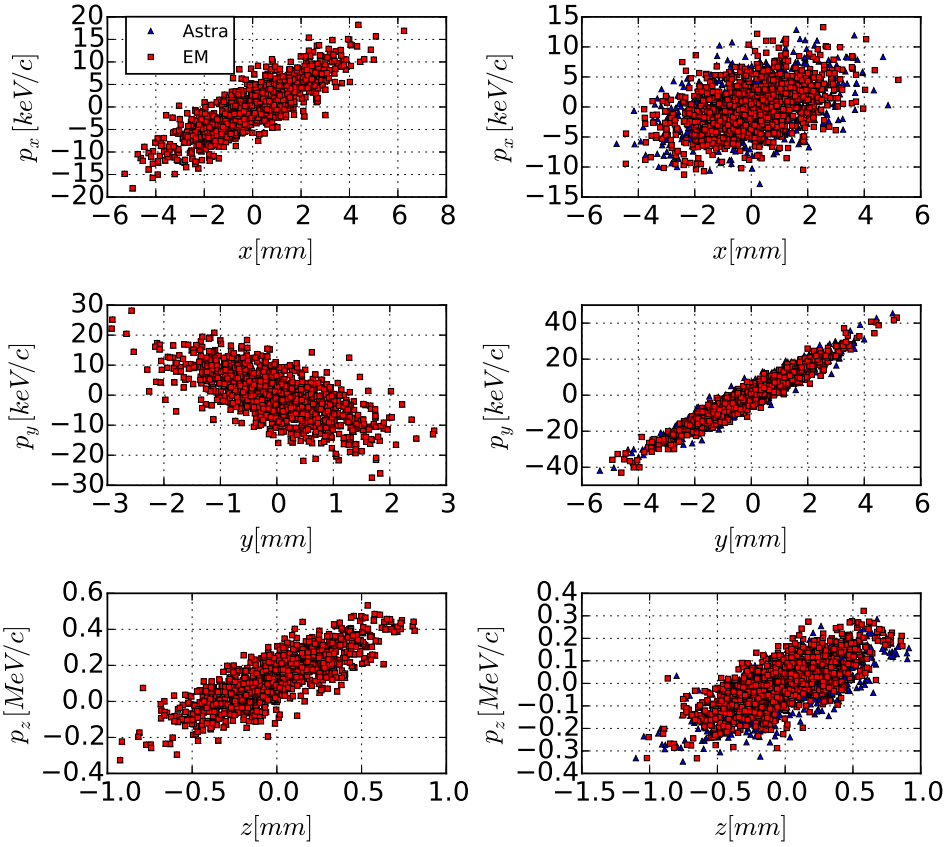

Figure 3.2: Tracking through a 1.3 m 9-cell cavity, followed by a 0.65 m drift, and another 1.3 m cavity (blue is Astra, red is the Empirical Model). The left column represents the initial beam phase spaces. The right are the phase spaces after tracking.

A sample of an interpolation table is shown in program 3.2. The table entries are coefficients of a Taylor expansion. For each column heading, the six digits after the 'x' are the exponents of the dependencies on the six beam coordinates. The column `1x102000` for instance, is the map element that represents the dependence of x on x_0 and y_0^2. It is a third order transport element, with one derivative against x_0 and two against y_0. Each line in the table represents one grid point.

The interpolation algorithm assumes the data grid is non-uniform. This is useful for elements that require different fineness for different parameter regions. Interpolation is performed using a combination of inverse-distance

Program 3.2 Sample interpolation table for the Empirical Model.

```
# Empirical Element Data
MaxB(G)     P(MeV)      3x000000    6x000000   1x102000
1.6000E-02  5.0380E-01  7.9411E-01  5.1924E-01 -2.6807E-01
1.6000E-02  5.3529E-01  8.1728E-01  4.9543E-01 -2.5850E-01
1.6000E-02  5.6678E-01  8.3556E-01  4.7528E-01 -2.4887E-01
1.6000E-02  5.9827E-01  8.5181E-01  4.5536E-01 -2.3963E-01
1.6000E-02  6.2976E-01  8.6582E-01  4.3676E-01 -2.3072E-01
1.6000E-02  6.6124E-01  8.7789E-01  4.1937E-01 -2.2245E-01
```

weighting and nearest neighbor, and in the case of a regular grid, should simplify to linear interpolation. Program 3.2 shows interpolation against two coordinates (momentum and B-field amplitude). However, the implemented algorithm works for arbitrary interpolation coordinate dimensions. Linear interpolation is performed. See the EM design document [40] for algorithmic details.

Chapter 4

Modeling the ERL

This chapter details the modeling of the ERL and decisions made in the optimization setup.

The machine is modeled starting from the main linac entrance. The injector is not modeled in the optimization due to 1) RIB injection transport is well-defined at the writing of this document, with the gun and injection cryomodule built, thus unlikely to be affected by the results of optimization, 2) simulating the already-built RIB injector only increases simulation running time, and 3) a preliminary ERL transport tune exists for the linac, therefore we can take the beam conditions at the linac entrance as the ERL input. Modeling software was chosen and/or written to provide full 6D transport description. Another important factor is for the software to be suitably accurate in the TRIUMF energy regime (10 - 50 MeV). The modeled machine is shown in fig. 4.1. The beam, for most of the modeling, is represented by first order envelope parameters, which proved to be sufficient.

The ERL beam is modeled with the deceleration pass and transport to the exhaust beam dump. RIB modeling tracks the high energy beam past the separator septum and two more quads. The rest of the RIB delivery line is not modeled.

A 3 mm maximum transverse RMS beam size is strictly enforced at all points of transport. These are the acceptable conditions to minimize beam loss in the E-linac and EHAT beam delivery design and we choose to impose them globally in the recirculation loop as well.

The main linac layout design is concrete and not debatable. However, the cavity gradients, cavity phases, and strengths of the quadrupole triplet in the center of the linac are free parameters for optimization. Likewise, positions of the merger and separator elements adhere to existing design notes, but strengths of quadrupole magnets in these areas can be optimized. The rest of the recirculation lattice, from exit of the separator septum to the merger entrance, is not tied to existing designs, thus all elements in this section are free parameters.

Objectives and Constraints

Here we define the objectives and constraints used in the global optimization problem. Top level criteria are used when possible.

The optimization objectives are:

1. Find viable electron transport for the ERL. Solution must also support RIB transport.

2. Maximize gain - maximum FEL lasing, therefore radiation power.

3. $E_{dmp} = E_{in}$ =7.5 MeV - energy recovery condition. If dump energy equals injection energy, the beam loading vectors cancel for maximum RF efficiency [52].

4. $\sigma_{x,dmp} = 7$ mm, $\sigma_{y,dmp} = 7$ mm - at the ERL dump, we relax the beam size condition. Instead, we want to blow up the beam to reduce radiation heating.

5. $\eta_x = \eta_{x'} = 0$ at end of both arcs - dispersion suppression for both arcs. This separates the lattice into dispersion independent sections for better modularity and tuning. Note that here arc 1 includes the separator and mirror separator.

6. $\alpha_x = \alpha_y = \eta_{x'} = 0$ at both arc centers - we try to look for a design with arc symmetries in β_x, β_y, and η_x. Although not critical to ERL operations, symmetries make tuning easier. The layout of the arc optics is symmetric as well to accommodate these conditions. η_y is zero everywhere and does not need to be considered. There is overlap with item 5.

7. Maximize E_{RIB} - ERL operations cannot interfere with RIB operations. High RIB energy is desired for photofission.

The design constraints are:

1. $\sigma_x \leq 3$ mm, $\sigma_y \leq 3$ mm - minimize beam loss by restricting transverse beam size everywhere. The bunch is modeled as Gaussian thus restricting the sigmas is sufficient. In the case when the bunch deviates from Gaussian see item 3.

2. $\sigma_{x,EDBT} \leq 3$ mm, $\sigma_{y,EDBT} \leq 3$ mm - for beam disposal, the beam size in the dump section must be constrained. This is complicated by the large energy spread obtained from lasing and amplified by deceleration,

which is converted to beam size by the EDBT dipole (fig. 4.7). This constraint overlaps with item 1, but we list it again to emphasize its importance.

3. beam loss $\leq 10^{-5}$ - additional beam loss condition imposed in arc 2. Lasing creates a momentum tail which can cause beam loss in the arc dipoles. Particle tracking is used in arc 2 to enforce this condition, in addition to item 1.

Free parameters in the optimization run are the RF settings and lattice optics. The list of parameters and their search ranges can be found in Appendix C. List of initial beam parameters and constants are also listed in Appendix C.

Design Considerations

The major design consideration is the E-linac itself. While most of the recirculating lattice are free variables, the E-linac design is concrete, and all upgrades must be compatible with it. The following were considered when choosing the initial layout.

- Space for the machine is limited. The ERL is housed in the hall where the E-linac is assembled. The size of the hall cannot be changed.
- The size of the machine must match the linac length of 8.31 m, extending from the beginning of cavity 1 to the exit of cavity 4.
- A four-dipole design is chosen for both arcs 1 and 2. The higher number of dipoles provides greater tunability. Arcs with greater than four dipoles were not considered as they require larger space and cost.
- Totally symmetric arcs and chicane dispersion suppression. Arc 1 is inherently not symmetric because the RF separator and septum cause the beam to enter at an off-angle and non-zero dispersion. We choose to restore symmetry by adding a mirror separator (fig. 4.1), which is a dipole-quad-dipole system which mirrors the real separator. The inclusion of the separator and mirror separator means the four-dipole arc 1 has a combined bending angle less than 180°. While symmetry is not compulsory, it is a desired transport characteristic and lends intuitiveness in tuning.

- All eight arc dipoles have identical geometry for simplicity and practicality. The arc 1 dipoles have smaller bending angles (see above point), achieved by operating at a lower gradient than the arc 2 dipoles.
- Initial beam parameters are taken from existing E-linac transport designs.
- While in theory all physical parameters can be included as free parameters for the optimization software to sort out, our baseline choices must be grounded in reality. For example, the upper limit of the quadrupole gradient is limited by magnet designs already used for the E-linac. This removes immediate spurious solutions.
- A minimum spacing of 25 cm between optical elements. This provides space for support and diagnostic equipment.
- A minimal set of constraints and objectives are chosen to demonstrate that local micromanagement is not necessary. Top level requirements force local parameters into place. For example, we do not specify matching conditions at the undulator, only requiring that FEL gain is maximized.

Machine Layout

The ERL component layout is shown in fig. 4.1. The translation of machine to simulation topology is shown in fig. 4.2. Note that the linac sections are shared between three beams: ERL pass 1, ERL pass 2, and RIB. Parameters in these sections are coupled and difficult to optimize individually without the global optimization platform.

ERL section information are listed in table 4.1. Several sub-systems in the ERL modeling make use of multiple engines. This allows the drawbacks of one engine to be offset by a different engine, and is made possible by the variable engine combination feature of the optimization platform.

A combination of MADX and DIMAD is used to model sections of the machine without energy changes. The MADX output files contain numerical values with greater number of significant digits and behave better than DIMAD for text parsing. For example, an ill-behaved design can produce dispersion up to 10^{15}. MADX displays this large value, whereas DIMAD displays the string **********. Although dispersions of such large magnitude are not physically realistic, it is useful for optimization as it provides information on search direction.

The section from separator to FEL matching is commonly referred to as 'a1'. Likewise, the section from arc 2 matching to linac pass 2 matching is referred to as 'a2'. The modeling information for the RIB beam is listed in table 4.2.

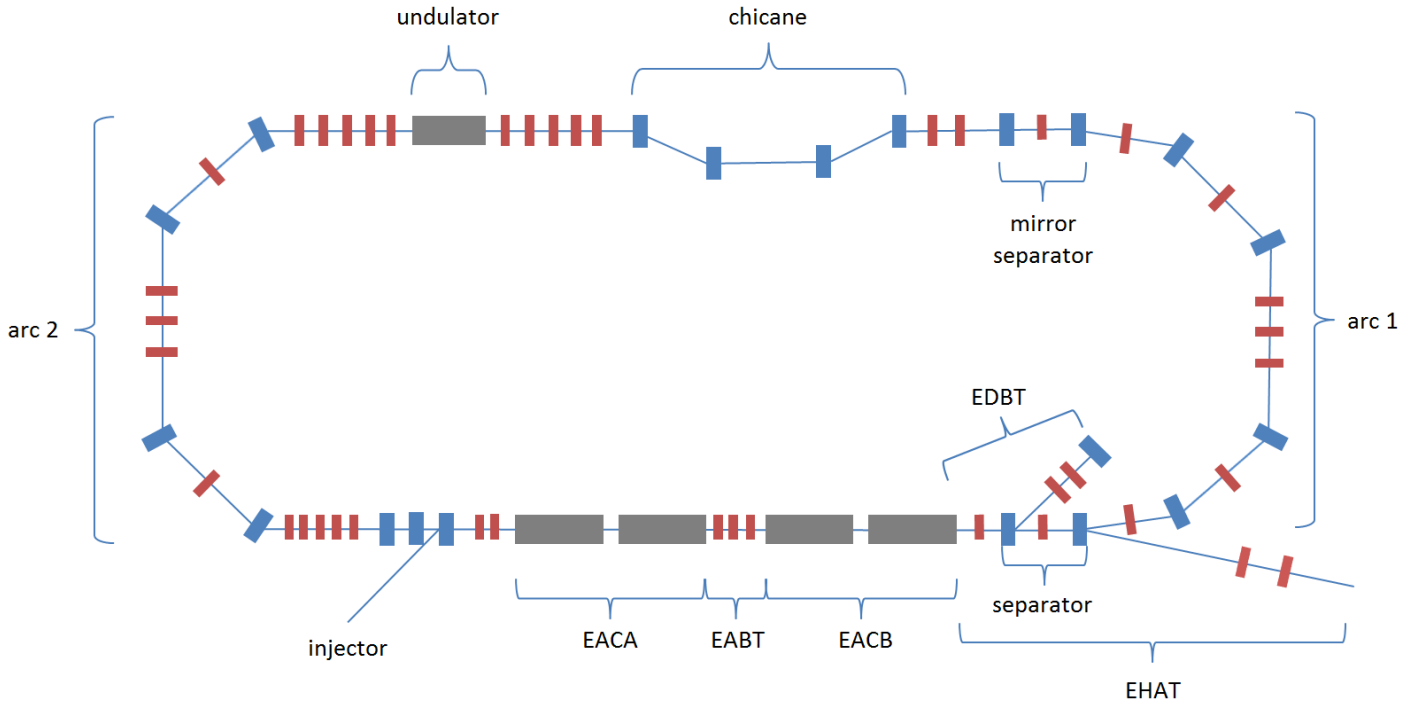

Figure 4.1: ERL baseline layout. The modeling of the machine starts at the first pass of the linac. Also included is the RIB beam which operates simultaneously with the ERL beam.

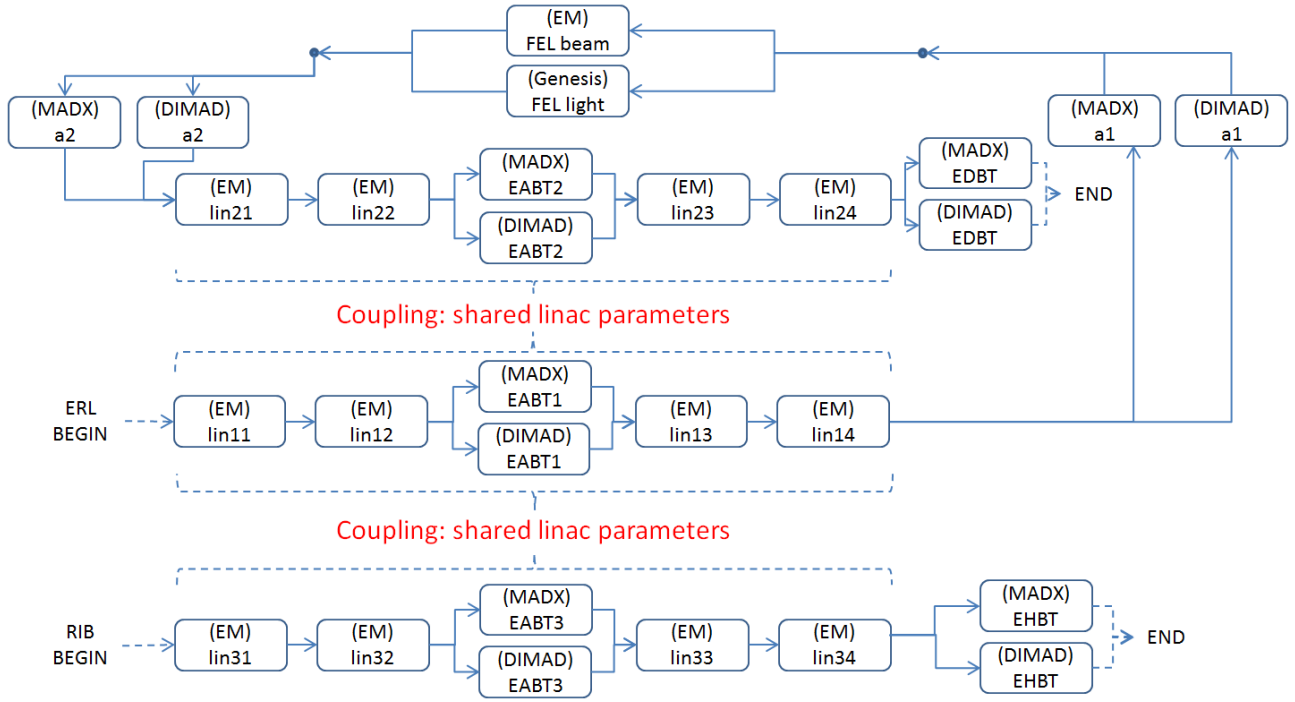

Figure 4.2: The ERL represented by the simulation tools used to model each section. The subsystems are combined together to form the optimization problem topology. The optimization platform allows engines to be combined in arbitrary order, parallel or serial. The accelerating elements are modeled by the Empirical Model, for which MADX and DIMAD analytical models are less than ideal for E-linac energies. Allowing for an arbitrary topology of different modeling tools is a key and novel feature of the optimization platform.

Table 4.1: List of ERL sections and their modeling information. The sections are listed from upstream to downstream.

Section	**Modeling Tool(s)**	**Description**
LIN11	EM	Linac cavity 1, ERL pass 1, including inter-cavity drift
LIN12	EM	Linac cavity 2, ERL pass 1
EABT1	MADX,DIMAD	ERL pass 1
LIN13	EM	Linac cavity 3, ERL pass 1, including inter-cavity drift
LIN14	EM	Linac cavity 4, ERL pass 1
SEP	MADX,DIMAD	RF separator
ARC1	MADX,DIMAD	First arc
SEP2	MADX,DIMAD	Mirror separator to cancel the effects of the separator
CHI	MADX,DIMAD	Compression chicane
FELM	MADX,DIMAD	FEL matching section
UND	Genesis	1 m undulator
A2M	MADX,DIMAD	Arc 2 matching section
ARC2	MADX,DIMAD	Return arc
MERG	MADX,DIMAD	3-dipole merger
LIN21	EM	Linac cavity 1, ERL pass 2, including inter-cavity drift
LIN22	EM	Linac cavity 2, ERL pass 2
EABT2	MADX,DIMAD	ERL pass 2
LIN23	EM	Linac cavity 3, ERL pass 2, including inter-cavity drift
LIN24	EM	Linac cavity 4, ERL pass 2
EDBT	MADX,DIMAD	ERL dump line

Table 4.2: List of RIB sections and their modeling information. The sections are listed from upstream to downstream. The linac is shared with the ERL beam.

Section	**Modeling Tool(s)**	**Description**
LIN31	EM	Linac cavity 1, RIB pass, including inter-cavity drift
LIN32	EM	Linac cavity 2, RIB pass
Continued on next page		

Table 4.2 – continued from previous page

Section	Modeling Tool(s)	Description
EABT3	MADX,DIMAD	RIB pass
LIN33	EM	Linac cavity 3, RIB pass, including inter-cavity drift
LIN34	EM	Linac cavity 4, RIB pass
EHAT	MADX,DIMAD	Transport to RIB target

Linac Pass 1

The 1.3 GHz linac [51] is composed of four superconducting niobium cavities accelerating the 100 MHz ERL beam to 45 MeV and the 650 MHz rare isotope beam to 50 MeV going into the separator. A quad triplet (EABT) is inserted between cavities 2 and 3. The two beams operate concurrently, occupying different RF buckets (RF periods with no beam). The linac layout is shown in fig. 4.3.

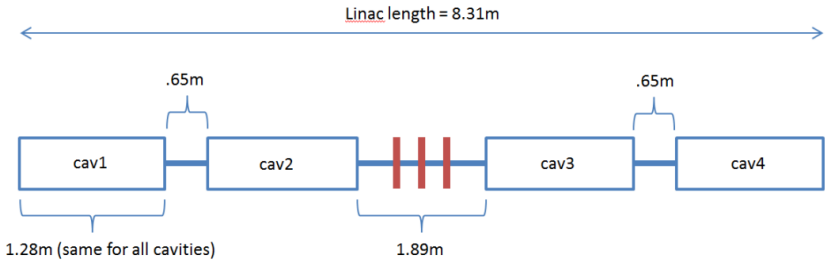

Figure 4.3: The main linac consists of four superconducting niobium cavities with a quad triplet. The linac is shared between ERL and RIB operations.

Tracking starts at the entrance of the linac with Gaussian beams. RIB parameters are taken from current design notes, with 10 MeV injection energy. ERL parameters are taken from initial design estimates, with 7.5 MeV injection energy. Certain parameters, in particular the longitudinal parameters, are estimated from the Peking University FEL [30, 34, 55, 72], which at 35 MeV and 120 pC bunch charge are comparable to the envisioned TRIUMF ERL. The injector introduces zero or negligible dispersion into the linac. Initial beam parameters are not variables in optimization so they do not affect existing injection tunes.

A phase difference is allowed between ERL and RIB operations. This represents different time-of-arrival of the two different bunches at the linac. The relative phase differences between individual cavities are equivalent for the two modes of operation.

Transport to Undulator

The linac exit beam is delivered to the FEL via the first arc transport. The initial recirculation beam is separated from the photofission beam via the separator, an RF kicker and septum combination, both modeled as dipoles, followed by the four-dipole arc 1, a mirror separator system, compression chicane, and a five quad matching section into the undulator. The complete section is shown in fig. 4.4. Magnets in the recirculation loop conform to existing TRIUMF designs [11, 68, 69].

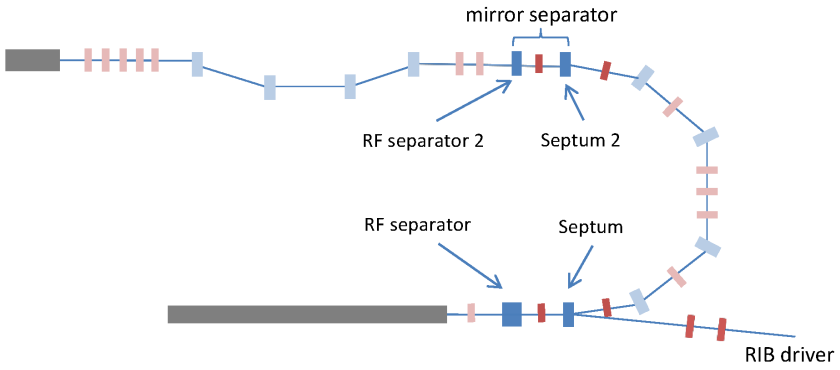

Figure 4.4: The first arc transport delivers the beam from the linac to the undulator. It includes the separator, arc1, mirror separator, chicane, and FEL matching section. The mirror separator corrects for distortions from the separator and restores symmetry to the system. Note that the RF Separator 2 and Septum 2 are plain dipoles.

The separator introduces a bend angle of 9.62° and a non-zero horizontal dispersion η_x leading into arc 1. Dispersion symmetry is restored by the addition of a mirror separator system after the fourth arc dipole. The mirror separator contains two dipoles designed to reflect the effects and bend angles of the septum and RF separator. The mirror septum is a simple dipole containing one chamber as opposed to two of the actual septum. The mirror RF separator is a basic non-RF dipole, but shortened to 30 cm as opposed to 80 cm for the actual RF separator, with the shortening designed to save

space and cost. The RF separator is a weak dipole and the shortening has negligible impact on its behaviour, with MADX simulations showing changes of 10^{-3} m and 10^{-6} in η_x and η'_x, respectively.

The arc dipoles and quadrupoles are designed to be totally symmetric. The bending angle is evenly divided amongst the four arc dipoles, which, together with the bending angles from the separator and mirror separator, create a 180° turn. Solutions with symmetries β_x, β_y, and η_x are desired (η_y is 0 everywhere due to the lack of vertical dipoles).

A totally symmetric four-dipole chicane is used to compress the beam leading into the undulator [66]. The chicane dipoles are identical with rectangular faces. Both the length of the chicane drifts and dipole bending angle are variables in the optimization, allowing for a wide range of chicane M_{56}. No objectives or constraints are placed on chicane M_{56} or compression factor. The top level requirements of beam transport and undulator gain are used as selection criteria. A variety of M_{56} can be achieved by manipulating the arc optics. In general, arc 1 rotates the beam from expanding to contracting and the chicane compresses the beam (fig. 4.5). The combined M_{56} of the two systems determines bunch length and peak current at the undulator (fig. 4.6).

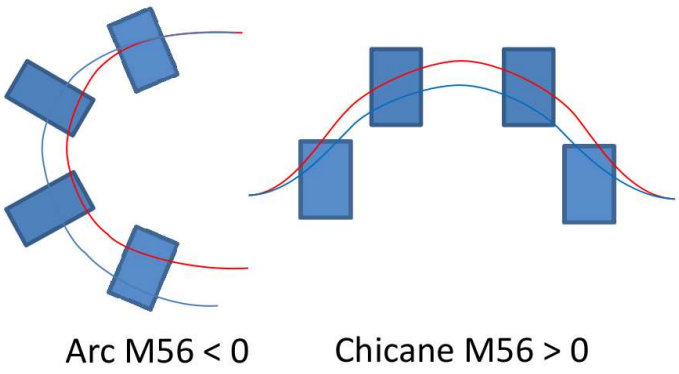

Figure 4.5: M_{56} of the linac-to-undulator transport is determined by arc 1 and the chicane. Due to the layout of the dipoles, arc 1 has a natural $M_{56} < 0$. A particle with less energy (red) takes a shorter path in arc 1 than a particle with more energy (blue). The chicane has $M_{56} > 0$. A particle with less energy (red) takes a longer path in the chicane than a particle with more energy (blue). The total M_{56} determines bunch compression at the undulator.

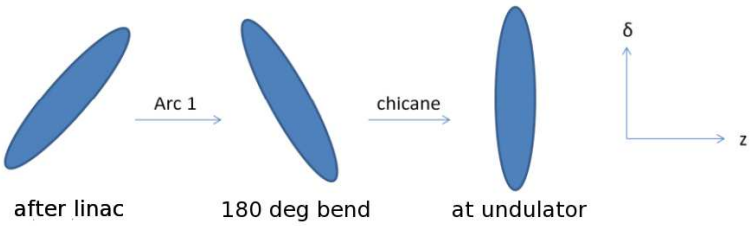

Figure 4.6: Longitudinal phase space manipulation in the first transport arc.

Both the linac-to-undulator and undulator-to-linac transports are designed for dispersion suppression. While efficient lasing operation does not require dispersion suppression, it leads to modularity in the system. Non-zero dispersion couples the two arcs, adding potential complications to beam tuning. The compression chicane is a completely symmetric system with no quads, therefore is inherently dispersion-free. This reduces the dispersion condition to purely the two transport arcs.

Undulator

The focus of the modeling is a one-pass tracking of the electron beam in the undulator at saturation using Genesis. Our goal is to find how the undulator, at steady state, affects beam dynamics. The FEL startup regime is not modeled, since the light-beam interaction is weakest at this stage. The FEL at saturation imparts the greatest disturbance on the beam, and it is our interest whether this exhaust beam can be adequately transported to the dump. A complete study of the optical system requires engineering decisions, e.g. choice of reflectivity for the mirrors and output coupling, which are too early to adopt at this stage of the design without scientific requirements set. Rough values estimated from machines with similar parameters are used.

Based on the PKU-FEL ([30]), a machine of similar scope, the undulator parameter K is chosen to be 0.7, and the undulator period λ_u is chosen to be 4 cm. The optical wavelength is estimated from the undulator equation ([32, 47]):

$$\lambda = \frac{\lambda_u}{2\gamma_r^2}(1 + K^2) \approx 4 \text{ µm} \tag{4.1}$$

where the Lorentz factor γ_r is based off the elinac energy of 45 MeV. The intra-cavity saturation power P_{sat} used for optimization is 8 MW. The value is taken from the Daresbury ERL Prototype ([45, 67, 70]), which has similar beam parameters and also 4 micron optical wavelength. A theoretical esti-

mate ([47] eqn. 4.61) using a combined mirror reflectivity R =0.99, beam power P_{beam} =0.5 MW, and number of undulator periods N_u =25,

$$P_{sat} \approx \frac{1}{2N_u(1-R)} P_{beam} \sim 1 \text{ MW} \tag{4.2}$$

supports the assertion of megawatt saturation power. Energy loss from undulator radiation is also incorporated into the simulation and carried into the return arc. The list of undulator parameters is shown in table C.5.

Transport to Dump

The return loop transport consists of a five quad matching section, arc 2, another matching section, and a three dipole merger. After the linac pass 2, the exhaust beam is disposed of in EDBT (fig. 4.7). The four arc dipoles are geometrically identical to those of arc 1 for ease of construction and design, each bending the beam by 45°. The totally symmetric design is also dispersion free if the incoming dispersion from arc 1 is zero. The merger is taken from existing design [24]. During deceleration, energy spread can grow by a factor of 10 due to anti-damping, potentially complicating beam disposal. This necessitates energy spread compression in the linac. The longitudinal phase space manipulation is shown in 4.8.

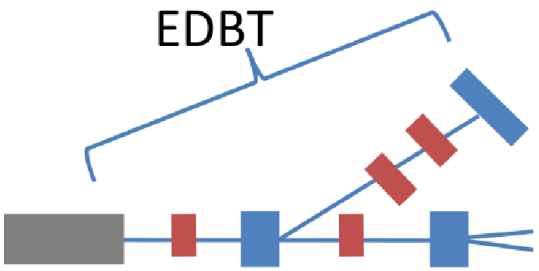

Figure 4.7: The dump transport consists of a quad shared with the accelerated beams, a dipole, followed by a quad doublet. The large energy spread out of the linac is converted to beam size by the dipole, potentially complicating disposal.

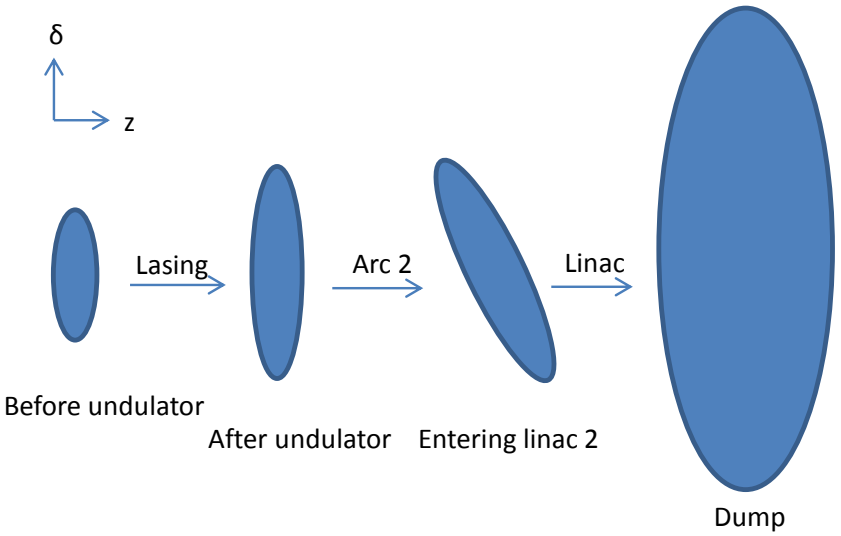

Figure 4.8: Longitudinal phase space manipulation in the second transport arc. The undulator induces energy spread, which increases due to linac anti-damping, and can complicate beam transport. Energy spread is shaped in the linac second pass for easier beam disposal.

Special care is paid to tracking beam loss in the return arc. The undulator introduces a momentum tail in the bunch which can be converted to beam loss by the arc dipoles. The tail can be seen in fig. 5.12.

No free parameters exist in the second pass of the linac. All RF parameters and EABT quad parameters are coupled to the first pass parameters.

Lattice Time-of-Flight

The lattice time-of-flight is an important variable for phase matching between linac passes. The TOF depends on

- TOF in linac pass 1
- TOF in recirculating lattice
- Variable-length chicane
- Beam energy
- Variable length drifts in lattice

The last parameter is a length parameter inserted into the arcs which allows the recirculation TOF to vary with a value between 0 and 1 RF wavelength, allowing for all possible pass 2 phases. No constraints are placed on the pass 2 phase. The phase is automatically determined by the top-level energy recovery objective $E_{dmp} = E_{in}$, where E_{in} is injector energy and E_{dmp} is the energy at the dump after recovery.

Each transport section creates a *time slip* variable designating the TOF through that section. The method of calculating the slip depends on the tool used to model the section. The Empirical Model inherently tracks using time, whereas the slip in Madx can be inferred by the path length s. A global variable keeps track of the cumulative time slip of the beam and converts it to phase when needed.

The four linac pass 1 cavity phases, ϕ_{11}, ϕ_{12}, ϕ_{13}, and ϕ_{14}, are allowed to vary independently, where ϕ is the phase when the beam is at the cavity entrance. For convenience, we define ϕ_{110}, ϕ_{120}, ϕ_{130}, and ϕ_{140}, which represent the cavity phases back-propagated to the simulation beginning, i.e. the cavity phases when beam is at linac entrance (by this definition, $\phi_{11} = \phi_{110}$). The second pass phases, ϕ_{21}, ϕ_{22}, ϕ_{23}, and ϕ_{24} are calculated by

$$\begin{aligned}
\phi_{21} &= \phi_{110} + \Delta t(\text{to pass 2, cav 1 entrance}) \\
\phi_{22} &= \phi_{120} + \Delta t(\text{to pass 2, cav 2 entrance}) \\
\phi_{23} &= \phi_{130} + \Delta t(\text{to pass 2, cav 3 entrance}) \\
\phi_{24} &= \phi_{140} + \Delta t(\text{to pass 2, cav 4 entrance})
\end{aligned} \tag{4.3}$$

where Δt is the globally cumulative time slip. A similar set of phases exist for the RIB beam, labeled ϕ_{31}, ϕ_{32}, ϕ_{33}, and ϕ_{34}. An optimization variable, $d\phi_{\text{ERL-RIB}} \equiv \phi_{31} - \phi_{11}$, allows the ERL and RIB beams to be phased independently, but the relative phases between cavities are identical since both beams share the same linac, e.g. $\phi_{330} - \phi_{320} = \phi_{130} - \phi_{120}$.

Rare Isotope Beam Transport

A small section of RIB transport section EHAT is included in the modeling to show consistency between ERL and RIB operations. The first section of EHAT is shown in fig. 4.9. The layout of elements in EHAT conform to the existing baseline design [26] and are not variables for optimization.

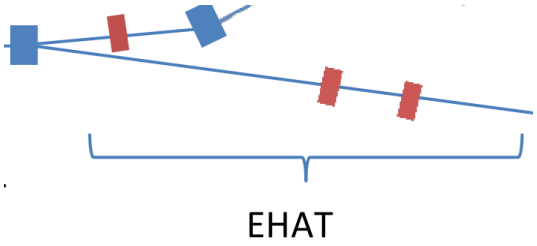

Figure 4.9: The rare isotope beam is delivered to the RIB targets through the EHAT line. RIB is separated from the ERL beam using the RF kicker and septum.

Other Considerations

Synchrotron Radiation

The largest sources of coherent synchrotron radiation (CSR) in the ERL come from the chicane and arc dipoles. CSR simulations were performed using the tracking program CSRtrack [2, 35], with 45 MeV, 100 pC Gaussian bunches. The chicane was modeled with four dipoles using identical geometric attributes as the chicane from the optimization setup. Tracking showed a relative energy loss of $4{\times}10^{-4}$, a negligible value and thus CSR calculations were excluded from the optimization setup.

The arc dipoles were similarly modeled using 45° dipoles from the optimization setup. Four dipoles produced a relative energy loss of 10^{-4}, and again small enough to be neglected from optimization. Together this shows that CSR is not an issue in the recirculation lattice and therefore not modeled.

Beam Breakup

Beam breakup (BBU) causes deflections in the beam when traveling through a structure where electrons give up energy to the deflecting mode. If Q_L, the loaded Q-value of the mode, is sufficiently high, the electromagnetic fields of the mode will increase until the beam scrapes an aperture, resulting in beam loss. The threshold current I_{th} is given by [44, 64]:

$$I_{th} = \frac{2c^2}{e(R/Q_L)Q_L\omega}\frac{1}{M_{12}\sin(\omega\Delta t)} \tag{4.4}$$

where e is the electric charge, $(R/Q_L)Q_L$ is the impedance, ω is the angular frequency of the mode, and Δt is the recirculation time. Extensive simula-

tion based on E-Linac recirculation geometry and optics has led to an upper limit for the characteristic impedance of any higher order modes (HOM) in the E Linac cavity of $R/Q * Q_L \leq 10^7$ Ohm [50]. This was shown to allow operation of the E-Linac ERL safely below the beam breakup (BBU) threshold for all conceivable geometry and optics.

2-pass ERL BBU instability was modeled with BI [12]. E-linac HOM data up to 4 GHz was used in the simulation. Table 4.3 shows the most damaging dipole modes. Simulations showed I_{th} to be ≈30 mA, which is greater than the maximum E-linac current of 10 mA. The four cavities are identical to first order, thus for modeling purposes the HOM data was treated to be identical for all cavities. In real life the frequencies could spread by .1% to 1%, R/Q and Q by up to 10% [48]. In addition, the quads of the recirculation loop contain enough freedom to shape M_{12} (or M_{34}) and mitigate unwanted excitations, with optimization results showing more than 2π range in phase advance, both horizontal and vertical.

Table 4.3: List of the most damaging dipole modes in the E-linac 9-cell cavity. Data provided by P. Kolb [49].

f (GHz)	**QL**	**R/Q (Ohm)**
3.837065314	2.89e6	0.00016778
3.840685874	1.34e6	0.000537193
3.844854002	0.884e6	0.001068607
1.890036165	0.482e6	0.035737708

Space Charge

A question is whether space charge (SC) is a concern for tracking, which can have an effect for energy regimes less than 10 MeV (ERL beam starts at 7.5 MeV) [60]. Tracking shows SC is not an issue. Fig. 3.2 compares tracking from the Empirical Model (without SC) with ASTRA (with SC), with no significant differences.

Chapter 5

Optimization Results and Tradeoff Studies

This chapter details results from the TRIUMF ERL optimization. Our chief goal is to understand the underlying physics of the ERL transport and tradeoffs between parameters. We detail the following:

- How does the RF and arc transport compress the bunch to maximize lasing at the undulator?
- How does lasing affect beam parameters? Does this cause problems for energy recovery or the control of energy spread? Can lasing lead to beam loss in the return transport? How does energy spread evolve due to lasing and energy recovery, as this is an important parameter which can lead to beam loss?
- Competition between objectives, such as energy recovery, energy spread, and how they affect the RF.
- Effect of the reciculation loop time-of-flight on energy recovery.
- Understand the physics of optimal input beam parameters into the undulator as obtained by global optimization.
- Transverse phase space control and the impact of demanding arc symmetry.
- How do higher order transport terms impact lasing?

We also show the evolution of important ERL parameters as the optimization proceeds.

Most plots shown in this chapter represent either the optimization population, or a subset of the population. Each point in the plots should be interpreted as an instance of the ERL, i.e. a particular machine design. These population plots are useful in illustrating the physics of the ERL, and

how different design parameters play against each. Often the plots show Pareto fronts between certain parameters, illustrating design tradeoffs.

Some plots show a percentage of the population in order to reduce the clutter of showing the full population.

Bunch Compression

Proper phase space manipulation is important to rectifying many beam dynamics challenges [62]. Proper bunch compression is important for producing a large peak current to maximize gain. We start by developing a relationship between RF phase and gain. We link the gain to the beam parameters at the end of the linac, then those beam parameters to the initial RF phase, and finally show that the RF phase is related to the gain. Beams at the start and end of arc 1 are related by

$$V_{zz,a1+} = V_{zz,a1} + 2M_{56}V_{z\delta} + M_{56}^2 V_{\delta\delta,a1} \tag{5.1}$$

Where (+) indicates the arc end and V_{ij} is the covariance between i and j. This arises out of $V_{a1+} = MV_{a1}M^T$ [19]. Fig 5.1 shows that the three terms of eqn. (5.1), related to $\langle z^2 \rangle$, $\langle z\delta \rangle$, and $\langle \delta^2 \rangle$, all contribute to the bunch length at the arc end. Since the bunch length impacts peak current, all three terms are important to gain.

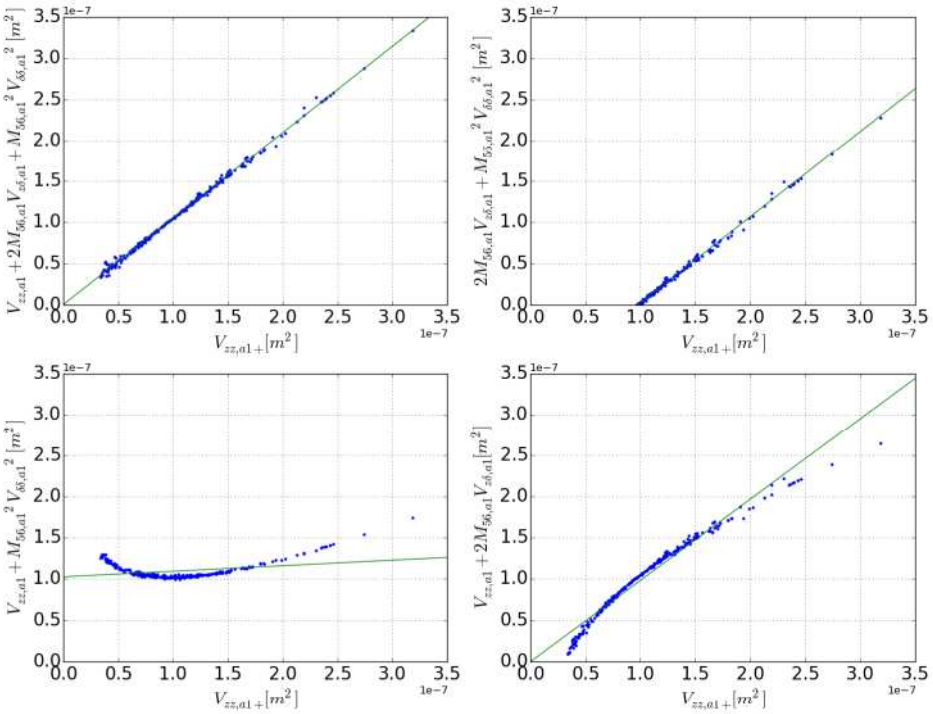

Figure 5.1: Top left: eqn. (5.1) relates z, $z\delta$, and δ at the arc entrance to z_+, where $(+)$ refers to the arc end. The line of best fit is $f(\langle zz\rangle+) = 1.047\langle zz\rangle_+ + 4.956 \times 10^{-10}$. The errors on the slope and y-intercept are 6.096×10^{-2} and 7.086×10^{-9}, thus showing the data is consistent with a completely linear relationship. Top right, bottom left, bottom right: neglecting the first, second, and third term from $f(\langle zz\rangle_+)$, respectively. None of the three can produce the desired relationship with z_+, thus showing all three terms are important contributors to bunch length, and therefore gain. The top right plot, although linear, results in the impossible situation of a point-beam creating a finite-sized beam.

The dependence of beam parameters on the linac acceleration phase ϕ_1 is shown in fig. 5.2. As shown in fig. 5.1 the beam parameters after the linac impacts peak current at the undulator. We therefore expect ϕ_1 to impact lasing. Fig. 5.3 and 5.4 illustrates the effect of ϕ_1 on peak current and gain.

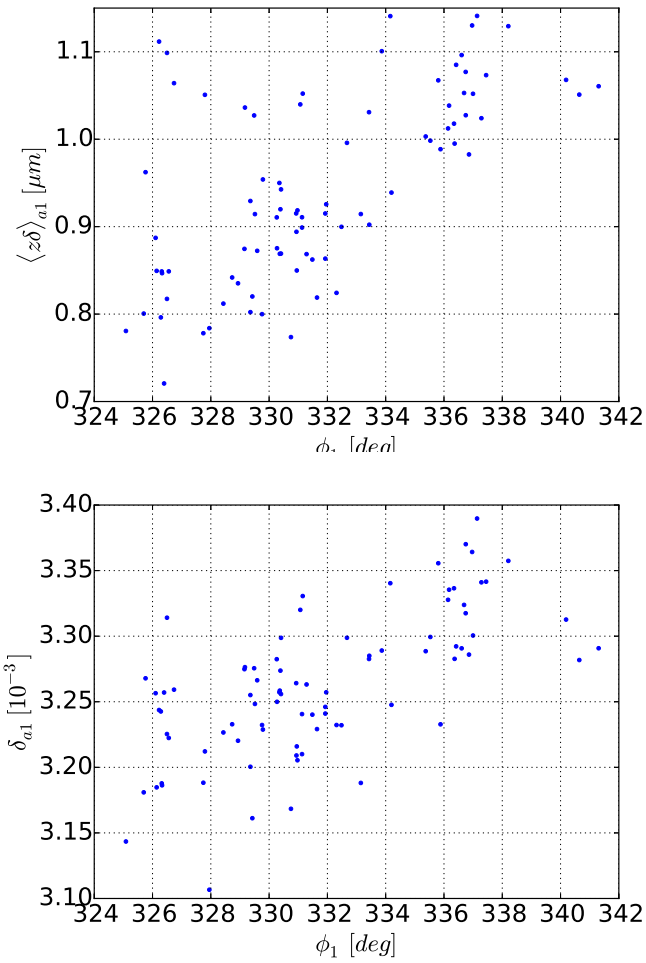

Figure 5.2: Beam parameters at the arc entrance is affected by the linac acceleration phase ϕ_1. Top: ϕ_1 vs $z\delta$ correlation. Bottom: ϕ_1 vs energy spread δ. Both have roughly linear relationships with ϕ_1 and shows the impact ϕ_1 can have on longitudinal beam parameters. Points shown have cavity gradients > 18 MV/m.

Interestingly, there is no optimal phase corresponding to optimal gain. The two plots of fig. 5.3 shows overlapping phase regimes produce different peak current I_p, and therefore gain, at different M_{56} (M_{56} is for combined arc 1 and chicane). The top plot is a slice in negative M_{56} and shows a positive slope in I_p as ϕ_1 increases. The bottom plot is a slice in positive M_{56} and shows a negative slope in I_p as ϕ_1 increases. Note the preference

for negative M_{56} slice which produces higher I_p. Since large positive M_{56}'s are not useful for compression, they are biased against in the optimization (large positive values do not even exist in later iterations, and the bottom plot uses data from an earlier iteration).

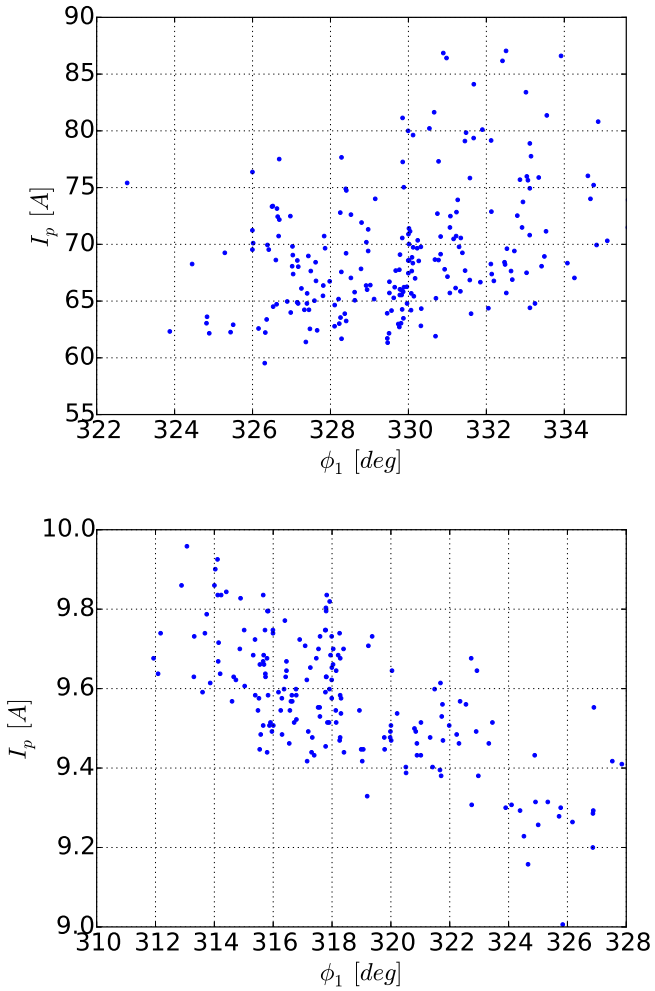

Figure 5.3: The acceleration phase ϕ_1 affects beam parameters and therefore FEL gain. Top: peak current I_p vs ϕ_1 with M_{56} in [-.15, -.05] m. Bottom: same parameters with M_{56} in [.3, .305] m. I_p depends on both the linac and the arc transport, thus slices in M_{56} are required to isolate the effects of ϕ_1. The difference in slope between the two plots corresponds to how M_{56} is matched to ϕ_1.

This illustrates that the combination of ϕ_1 and M_{56} are needed to properly compress the bunch, and that the ERL compression scheme is a combination of massaging the bunch in the linac, and compression in the arc and chicane transport. The highest peak current occurs near the region $M_{56} \approx -0.1$ m. The direct effects of RF on the gain are described next.

Fig. 5.4 directly shows the effects of ϕ_1 on the gain. The data forms the typical shape of the RF curve, demonstrating the important role of RF in shaping the bunch for compression and lasing.

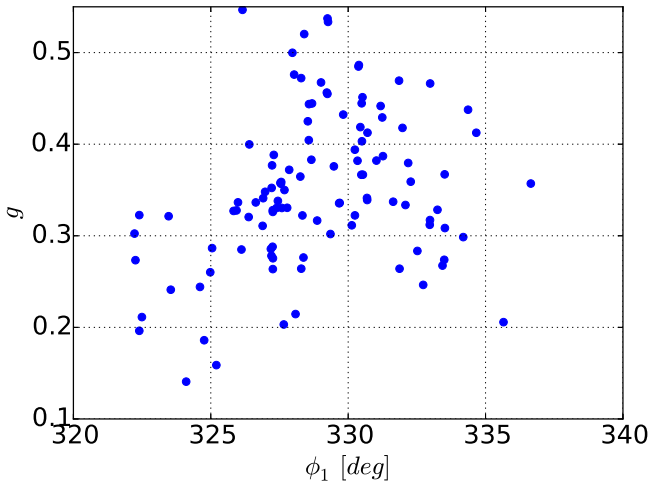

Figure 5.4: FEL gain vs acceleration phase ϕ_1. Slices made on CS parameters at the undulator entrance: $\beta_x \leq 2$ m, $\beta_y \leq 2$ m, $-0.5 \leq \alpha_x \leq 0.5$, and $0.5 \leq \alpha_x \leq 1.5$. These slices are centered around the optimal transverse matching conditions for the undulator (eqn. 5.10), in order to isolate longitudinal effects. The solution set encompasses physics of the linac, arc transport, and lasing. No single simulation tool can provide all the physics modeling necessary.

The highest gain occurs when ϕ_1 is several degrees before crest (335°). The baseline ERL layout presented in the next chapter has a ϕ_1 in this region, roughly 10° before crest. Since the gain is affected by both longitudinal and transverse parameters, cuts were made in the transverse parameter space to isolate the effects of ϕ_1. The cuts made are described in the fig. 5.4 caption.

The data of fig. 5.4 includes multiple engine tracking through the linac, arc 1, chicane, and undulator, and is difficult to achieve without the combined modeling capabilities of the global optimization platform.

Interestingly, at the beginning of optimization, M_{56} can reach large positive and large negative values (fig. 5.5), courtesy of the sizable degrees of freedom given to the arc and the chicane. Also note that the linac is given the full search range for phase, on either side of the RF crest. Why then, is only a negative $M_{56} \approx -0.1$ m capable of compressing the beam? We can imagine a bunch accelerating on one side of the RF crest resulting in a longitudinal profile that requires a negative M_{56} for compression. Then the bunch accelerating on the opposing side of the RF should require a positive M_{56}. Fig. 5.5 should contain two peaks to either side of $M_{56} = 0$.

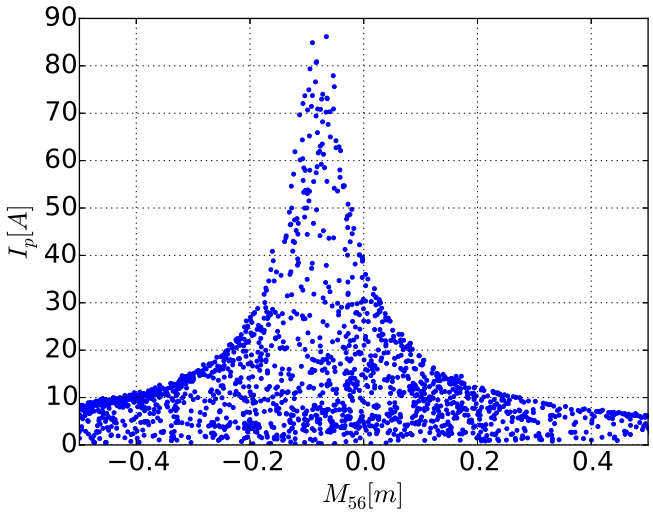

Figure 5.5: M_{56} of arc 1 and chicane vs peak current in an earlier generation. A negative M_{56} is required for compression.

The reason that only one sign of M_{56} is selected for compression is because a longitudinally expanding bunch is used as the starting bunch ($\alpha_z \approx -1.5$) before acceleration. A final contracting bunch is required for positive M_{56} compression. The RF has a large impact on the resulting longitudinal space, but the initial bunch is expanding too quickly and the RF cannot drive the expanding bunch to a contracting bunch (fig. 5.6). Therefore, no positive M_{56} is required for compression.

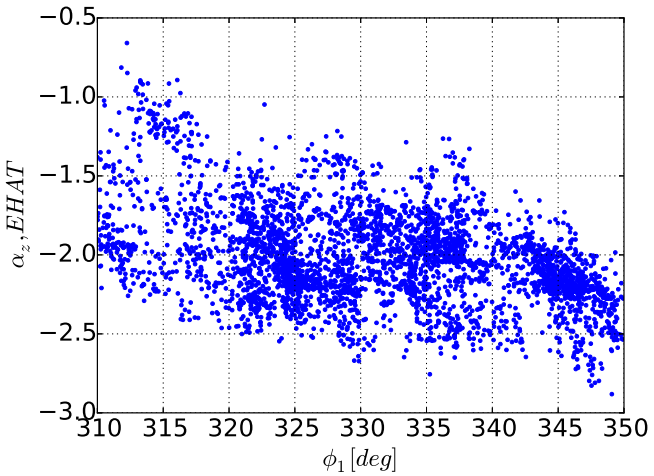

Figure 5.6: Effects of phase on longitudinal parameter α_z. The RF has a large impact in shaping the bunch, shown by the large spread in α_z. However, α_z started with a large negative value and the RF cannot drive it to positive.

To confirm that the RF cannot drive the bunch α_z from negative to positive, we estimate the RF effects on energy compression. The initial ERL beam was given a bunch length $\sigma_z = 0.0003$ m and energy spread of 0.019, which translates to an absolute energy spread $\sigma_E = 0.14$ MeV at an initial beam energy of 7.5 MeV. The RF phase range σ_ϕ occupied by this bunch is

$$\sigma_\phi = \frac{\sigma_z f}{c} 2\pi = 0.008 \tag{5.2}$$

where $f = 1.3$ GHz is the RF frequency and c is the speed of light. Suppose the bunch is accelerated at ϕ degrees before crest. The energy spread after acceleration is estimated from the change in particle energy E:

$$\begin{aligned} E &= E_0 \cos\phi \\ \mathrm{d}E &= -E_0 \sin\phi \ \mathrm{d}\phi \\ &\approx -E_0 \phi \ \mathrm{d}\phi \\ &= -(40 \text{ MeV})(\pm 20^\circ \pi/180^\circ)(0.008) \\ &= \mp 0.11 \text{ MeV} \end{aligned} \tag{5.3}$$

where $E_0 \approx 40$ MeV is the energy gain in the linac if the particle is accelerated on-crest (7.5 MeV to 45 MeV). ϕ was allowed to vary in the optimization

20° to each side of the crest. We therefore use $\phi = \pm 20^\circ$ because these are where the RF slope is greatest. The phase spread was estimated $\mathrm{d}\phi \approx \sigma_\phi$.

$\mathrm{d}E = \mp 0.11$ MeV is therefore the change in σ_E. The difference in sign represents which side of the RF crest the bunch is accelerated at, where the RF slope can increase or decrease σ_E. Adding this value to the starting $\sigma_E = 0.14$ MeV, the final σ_E can be 0.25 MeV or 0.03 MeV. Using the initial $\alpha_z = -1.5$, α_z after acceleration can be

$$\begin{aligned} \text{new } \alpha_z &= \text{old } \alpha_z \frac{\text{new } \sigma_E}{\text{old } \sigma_E} \\ &= -1.5 \begin{cases} \frac{0.25 \text{ MeV}}{0.14 \text{ MeV}} & \text{accelerating on right of RF crest} \\ \\ \frac{0.03 \text{ MeV}}{0.14 \text{ MeV}} & \text{accelerating on left of RF crest} \end{cases} \\ &= \begin{cases} -2.7 & \text{accelerating on right of RF crest} \\ \\ -0.2 & \text{accelerating on left of RF crest} \end{cases} \end{aligned} \tag{5.4}$$

The new α_z range of [-2.7, -0.2] agrees with fig. 5.6, and shows that the linac did not drive the bunch from expanding to contracting.

We test this hypothesis by reoptimizing with an initial longitudinally upright bunch ($\alpha_z = 0$). α_z after the linac therefore depends only on the RF phase. The results are shown in fig. 5.7. Depending on the phase with which the bunch is accelerated, the output bunch can be longitudinally contracting or expanding. Two peaks in M_{56} are seen, one near -0.1 m and one near +0.1 m, corresponding to the compression of an expanding and contracting bunch, respectively. This is important, because the ERL gun is not yet designed and the exact bunch parameters at the linac are not known. This reassures that the arc design has enough freedom to produce the necessary M_{56} for the compression in either direction of α_z and RF phase.

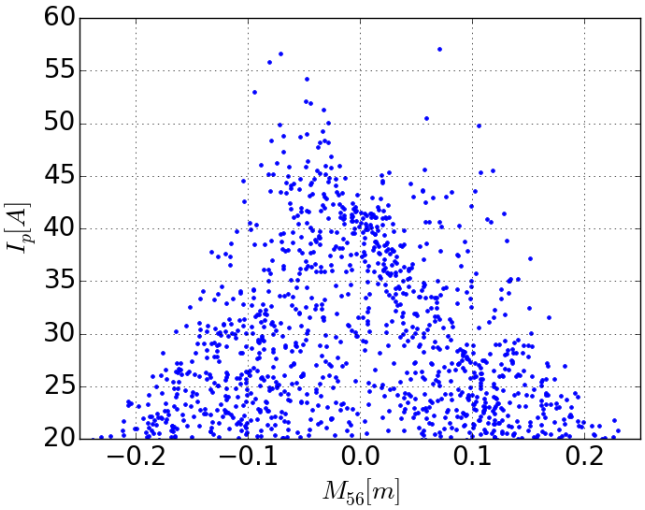

Figure 5.7: M_{56} vs peak current, starting with no longitudinal correlation in the initial bunch ($\alpha_z = 0$).

Effects of Lasing on Beam Transport

At the undulator, the electron-light interaction changes the electron beam properties, including energy loss and increase in energy spread, which can have consequences for the return transport.

Lasing reduces the beam energy (fig. 5.8). However, even with maximum beam-laser interaction, the energy lost represents 0.001 of the beam (45 MeV beam losing 40 keV). This is insignificant and not a factor for energy recovery.

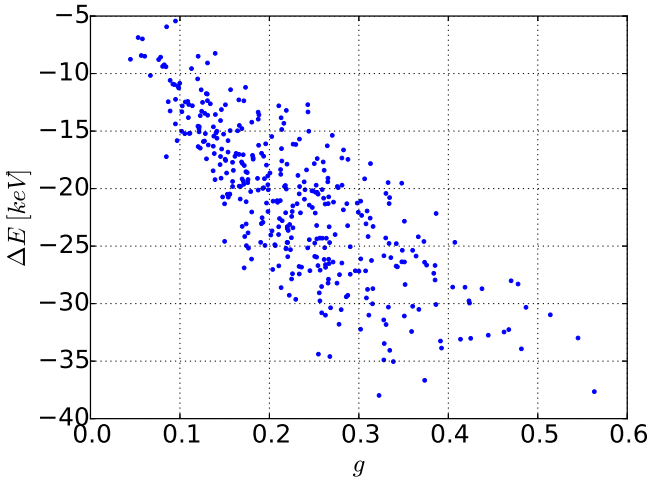

Figure 5.8: Energy loss ΔE through lasing. At maximum lasing, the 45 MeV beam loses 0.1% energy. 10% of the population is shown.

The lasing process also induces a beam energy spread (fig. 5.9). We are concerned with whether this distorted beam can be adequately energy recovered and transported. Fig. 5.10 shows how the energy spread δ at the beam dump and its dependence on lasing (notice the magnitude of δ before and after deceleration, compared with fig. 5.9). Both figures show points in the same slice of the deceleration phase ϕ_2. The increase in δ due to deceleration is evident.

The lasing process induces an energy spread directly correlated with the gain. Thus, high lasing complicates beam disposal. This is unavoidable since maximizing the gain is a top level design requirement.

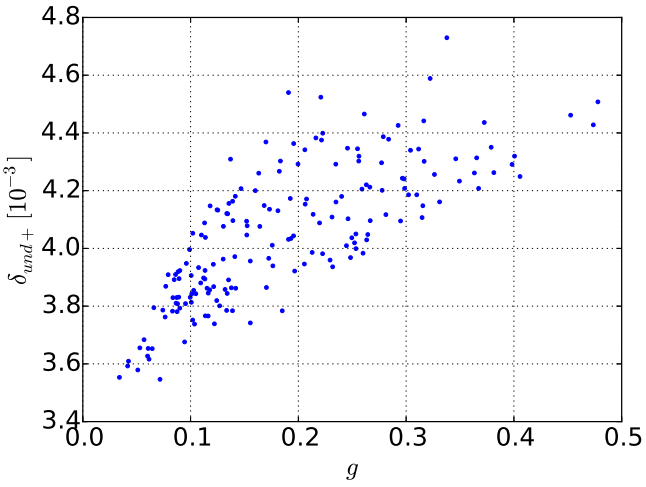

Figure 5.9: Energy spread after the undulator δ_{und+} increases with an increase in gain, potentially complicating beam disposal in the return arc. Points are taken from a slice in ϕ_2 between 150° and 160°.

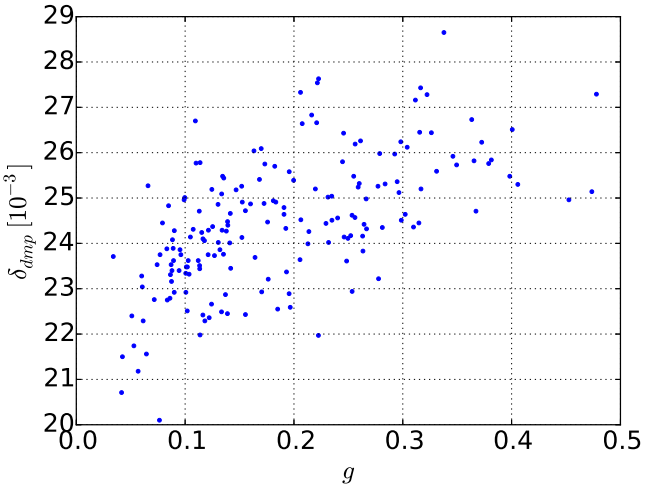

Figure 5.10: δ at the dump transport EDBT, as a function of gain. Lasing induced energy spread increases the final energy spread at the dump. This figure is a representation of all the physics in an ERL, ranging from lasing to energy recovery, and is made possible by the optimization platform. Points are taken from a slice in ϕ_2 between 150° and 160°, the same points from fig. 5.9 after deceleration.

We now explore the relationship between gain and energy recovery.

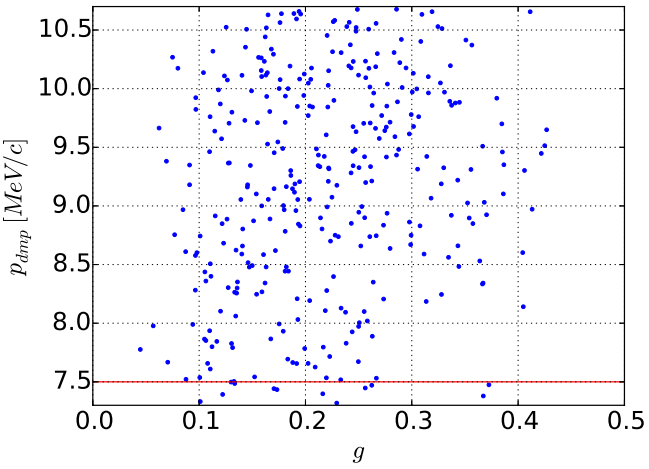

Figure 5.11: Top: gain vs energy recovered momentum. Gain and energy recovery are two high level objectives, and it is possible to satisfy both concurrently. The red line indicates the optimal energy recovered momentum for RF efficiency, and it is possible to maximally lase while maintaining optimal RF efficiency. 10% of the population is shown.

Fig. 5.11 shows the two top level ERL objectives of high gain and optimal energy do not compete with each other. Optimal energy recovery depends strongly on the phase change between acceleration and deceleration, which in turn depends on the recirculation loop time-of-flight.

Energy recovery depends on the 0th order transport term (TOF), which lasing does not change, explaining the lack of correlation between the two objectives.

Lasing distorts the longitudinal phase space, creating a momentum tail. Fig. 5.12 shows the momentum tail in the skewed normal distribution from Genesis.

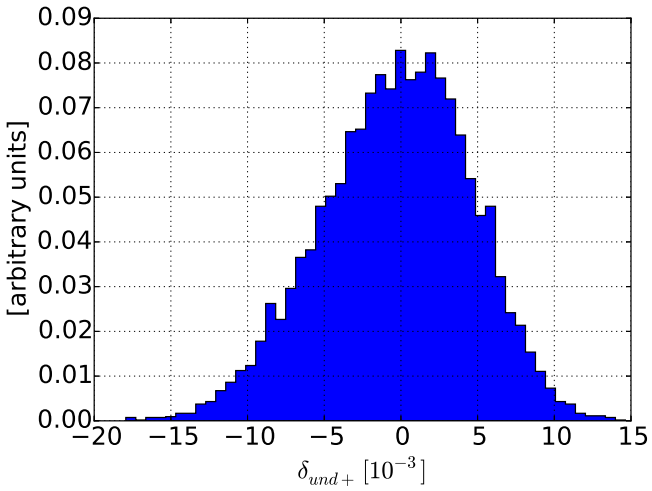

Figure 5.12: Energy deviation of particles after lasing. A slight tail can be observed. Its effects need to be tracked in the second transport arc.

If not controlled, the tail can become an issue in the return arc transport due to the arc dipoles, where dispersive effects can cause beam loss. In arc 2 we demand three constraints:

$$\begin{aligned} \max \sigma_x &< 3 \text{ mm} \\ \max \sigma_y &< 3 \text{ mm} \\ \text{beam loss } &< 10^{-5} \end{aligned} \tag{5.5}$$

The third condition is not required for arc 1 because the beam in arc 1 is Gaussian and has no tail.

The beam loss condition is for particles whose radius exceeds 10 mm. Fig. 5.13 shows the result of imposing all three constraints in optimization. The addition of constraining beam loss proves to be a tighter condition than constraining beam size only, particularly for individuals with large maximum σ_x or σ_y. Although the beam envelope proves small enough, beam loss results from the momentum tail and requires tightened global beam size for lossless transport. Fig. 5.13 shows that beam loss tightens the beam size constraints to $\sigma_x < 2.9$ mm and $\sigma_y < 2.8$ mm. Fortunately, the figure also shows that approaching the maximum allowed beam size is not necessary to provide a valid transport solution, with the optimization platform able to find solutions with $\sigma_x < 2$ mm and $\sigma_y < 1.5$ mm, although such stringency is not required.

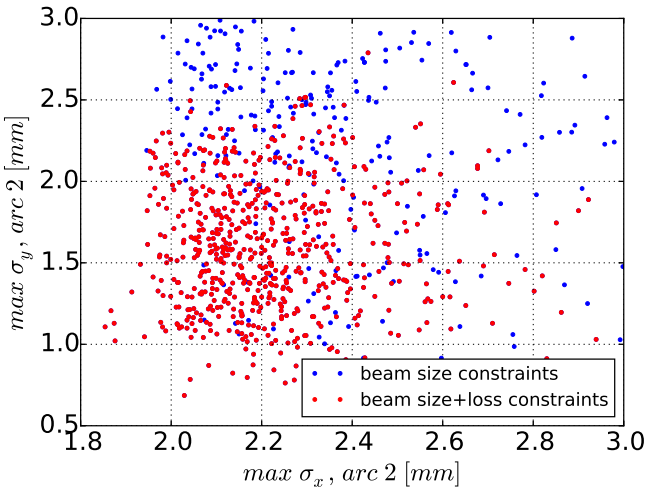

Figure 5.13: Maximum horizontal and vertical beam size in the return arc. Blue are solutions which satisfy the first two constraints of eqn. (5.5) regarding beam size. Red are solutions which satisfy all three constraints, including beam loss $< 10^{-5}$. The red solutions are a subset of the blue solutions. Included in the plot are only solutions with high gain ($g > 0.3$). High gain is an optimization objective and also introduces stronger beam distortions, thus making the beam more susceptible to loss through the momentum tail. Imposing beam loss does constrain the arc transport further, as can be observed at the high beam size regions where only blue exist. However, plenty of individuals exist which satisfy all constraints; therefore beam loss should not be an issue for transport.

Effects of Deceleration on Energy Recovery and Energy Spread

The deceleration pass in the linac is important for both energy recovery and energy compression. The deceleration phase ϕ_2 is determined by the acceleration phase ϕ_1 and the loop recirculation time Δt. Δt is allowed to vary in the optimization over more than one RF period. We now illustrate the consequences of choosing a ϕ_2.

Fig. 5.14 shows the important effects of RF phase. Both momentum and energy spread are greatly affected by the deceleration phase. The closer the bunch enters the linac on-crest, the greater deceleration it experiences, but also leads to a decrease in RF slope and therefore less control of energy

spread.

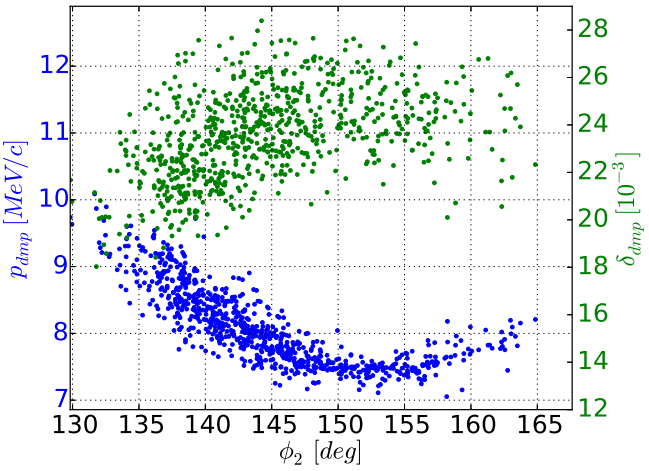

Figure 5.14: RF deceleration phase ϕ_2 vs energy recovered momentum p_{dmp} and energy spread at dump δ_{dmp}. This figure shows the importance of RF phase matching. The greatest decrease in p_{dmp} occurs at the RF crest, which also corresponds to a lack of control in δ_{dmp} due to less RF slope. The two plots suggest a negatively correlated relationship between δ_{dmp} and the ability to energy recover. Showing points with a $180° \pm 10°$ phase change between acceleration and deceleration phases.

Since the time of arrival at the deceleration pass depends on the phase change $\Delta\phi \equiv \phi_2 - \phi_1$, both p_{dmp} and δ_{dmp} show similar dependences on $\Delta\phi$ (fig. 5.15). The p_{dmp} data agrees with the theoretical prediction that 180° is optimal for energy recovery to 7.5 MeV/c.

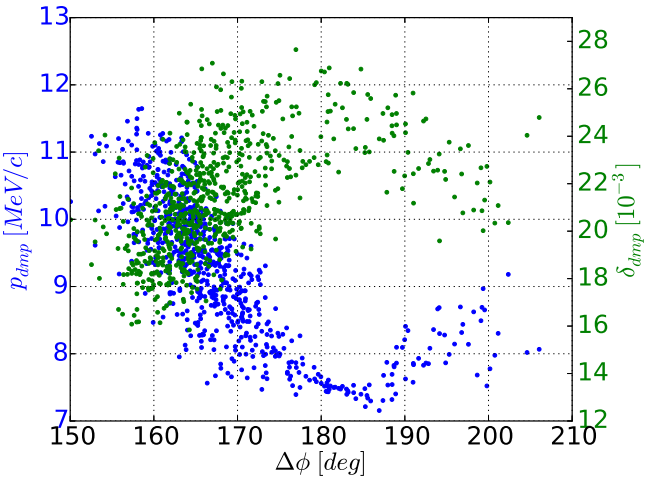

Figure 5.15: The change in phase between the two linac passes $\Delta\phi$ is very important for energy recovery. The optimal $\Delta\phi$ for energy recovery is 180°, where the plot shows almost precise recovery to the desired 7.5 MeV/c, but also making it bad for energy compression. Shows 20% of the population.

Fig. 5.16 shows the tradeoff between parameters during deceleration, forming a Pareto front between energy recovery and energy spread. For maximum energy recovery, we wish for the final momentum to be 7.5 MeV/c. Correspondingly, the minimum achievable relative energy spread is 0.021. Interestingly, the figure also shows that it is theoretically possible to recover more beam energy than the linac put in, with some individuals of the optimization population achieving less than 7 MeV/c, less than the injection momentum of 7.5 MeV/c. If energy spread at the dump becomes a pressing issue, it is possible to sacrifice energy recovery to achieve smaller energy spread.

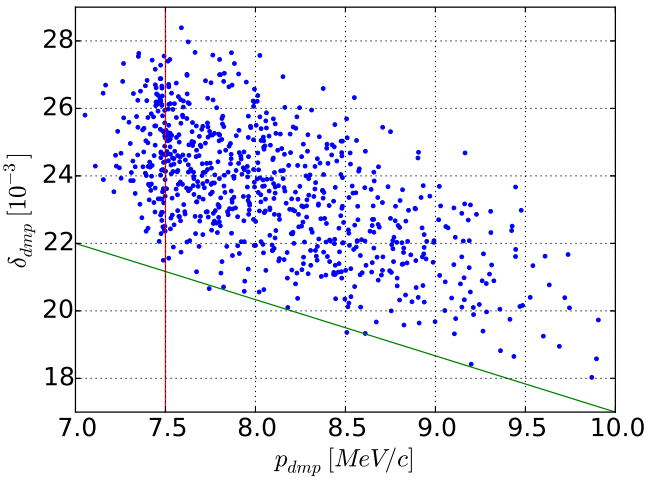

Figure 5.16: A Pareto front forms between energy recovered momentum p_{dmp} and energy spread at dump δ_{dmp}. For energy recovery, we wish to move left towards the vertical red line, which shows the desired 7.5 MeV/c, equal to the injection momentum. At this momentum only a 0.021 minimum δ_{dmp} is achievable. To lower δ_{dmp}, we wish to move to the right. The two parameters are fighting against each other because they both depend on the RF phase. Showing same set of points as fig. 5.14.

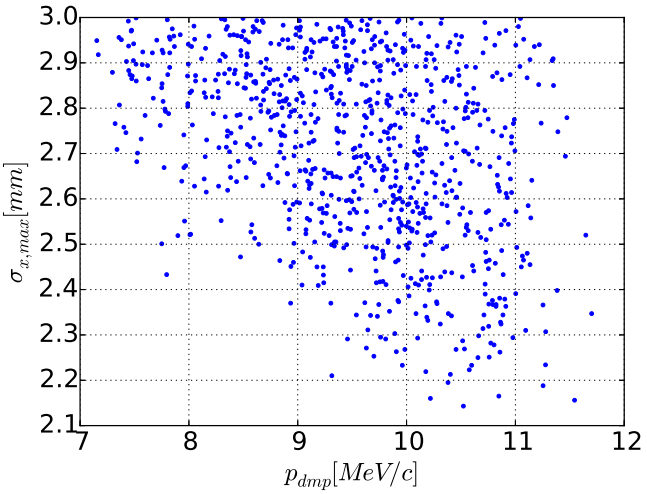

Figure 5.17: A Pareto front forms between final momentum and EDBT beam size. Maximizing energy recovery increases the energy spread, which leads to an increase in beam size.

Since the effect of a large energy spread is beam loss via beam scraping in EDBT, we directly look at the relationship between energy recovery and beam size. Fig. 5.17 shows the energy recovered momentum p_{dmp} and the maximum horizontal beam size in EDBT $\sigma_{x,max}$. A clear bound can be seen, where beam size is larger as we approach the optimal 7.5 MeV/c energy recovery point. To obtain optimal energy recovery, the deceleration pass must be driven at close to the RF crest, where less RF slope exists for energy compression, leading to higher energy spread and thus larger beam size. This demonstrates that having the best energy recovery requires sacrifices in beam control, and vice versa.

Tradeoff Between the Recirculation Loop Time-of-Flight and Energy Recovery

Fig. 5.18 illustrates the effect of acceleration phase ϕ_1 on energy recovery. Three slices were taken in the recirculation loop time-of-flight Δt: Δt that brings the bunch back to the linac with $\approx 180^\circ$ phase change (red), $< 180^\circ$ (blue), and $> 180^\circ$ (black). These represent bunches which arrive at the linac with perfect 180° timing, early, and later, respectively. Interestingly, when the energy recovered momentum p_{dmp} and ϕ_1 are plotted for the three

slices, we find that all three slices cross 7.5 MeV/c, which is the injector momentum and represents our objective of $E_{dmp} = E_{in}$ for energy recovery.

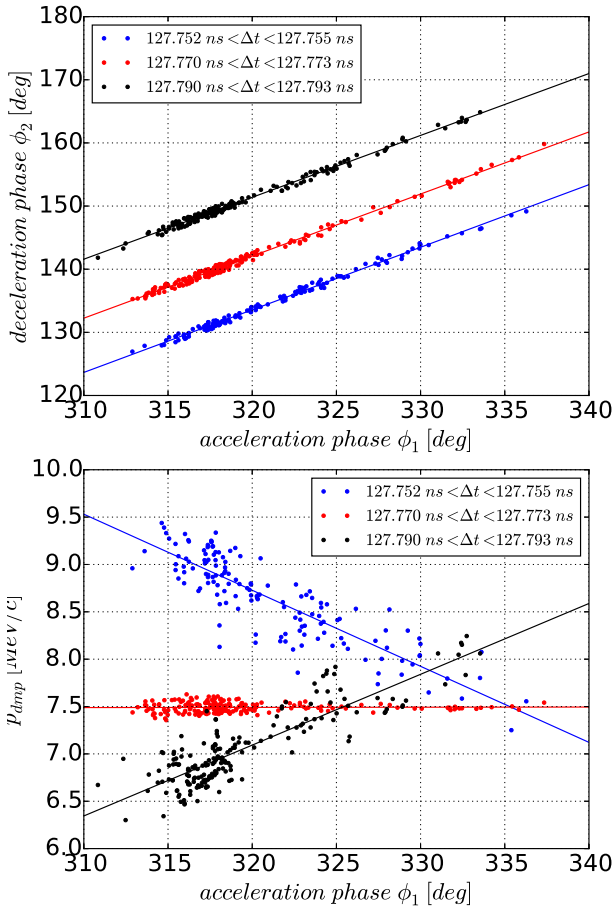

Figure 5.18: Top: correlation between linac acceleration phase ϕ_1 and deceleration phase ϕ_2. Each set represents solutions where the recirculation lattice has a particular range of time-of-flight Δt. The red solutions represent a phase difference of $\Delta\phi = \phi_2 - \phi_1 \approx 180°$, whereas the blue solutions arrive earlier than 180° and black solutions arrive later than 180°. Bottom: energy recovery as a function of ϕ_1, for different slices of Δt. Note that for all three slices, solutions exist to satisfy the energy recovery objective $E_{dmp} = E_{in}$ at certain phases. The slice represented by red is phase independent and corresponds (see left figure) to the theoretically optimal $\Delta\phi_{optimal} = 180°$. The blue and black slices show that satisfying $E_{dmp} = E_{in}$ can still be achieved at other phase settings.

Δt and its effect on energy recovery is shown in fig. 5.11. The role of ϕ_1 becomes clear once Δt is fixed. If Δt transports the beam such that ϕ_1 is

180° phase flipped from the deceleration phase ϕ_2, optimal energy recovery is achieved regardless of the initial ϕ_1 due to the exact cancellation of the RF acceleration and deceleration curves, as shown by the red solutions in fig. 5.18:

$$\begin{aligned} V_2 &= -V_1 \\ \sin\phi_2 &= -\sin\phi_1 \\ \phi_2 &= \phi_1 + 2\pi(n+0.5) \end{aligned} \tag{5.6}$$

where the phases are independent of the recirculation time-of-flight Δt. If a non-180° $\Delta\phi$ is used, $E_{dmp} = E_{in}$ is still possible at specific ϕ_1, as illustrated by the blue and black solutions. The reason is as follows: ϕ_1 and ϕ_2 are coupled by Δt, which determines the offset between the acceleration and deceleration curve (fig. 5.19). Perfect cancellation occurs if the curves are maximally out-of-phase (180° offset), or if they deviate such that ϕ_1 is coupled to ϕ_2 on the opposing side of the RF trough, shown by the diagonally dashed line. The horizontally dashed line indicates $\Delta\phi$ needed for this particular instance. For $E_{dmp} = E_{in}$, Δt is expected to be linearly related to ϕ_1 by

$$\begin{aligned} \phi_{2'} - \phi_1 = 2\pi f \Delta t &= 2d\phi + 2\pi(n+0.5) \\ &= 2(1.5\pi - \phi_1) + 2\pi(n+0.5) \\ \Delta t &= \frac{2(1.5\pi - \phi_1) + 2\pi(n+0.5)}{2\pi f} \\ &= -2/(2\pi f)\phi_1 + \text{const} \\ &= -0.00427 \text{ ns/deg} \times \phi_1 + \text{const} \end{aligned} \tag{5.7}$$

where the prime indicates the opposite side of the crest/trough and $2d\phi$ is the difference between the primed and unprimed phases.

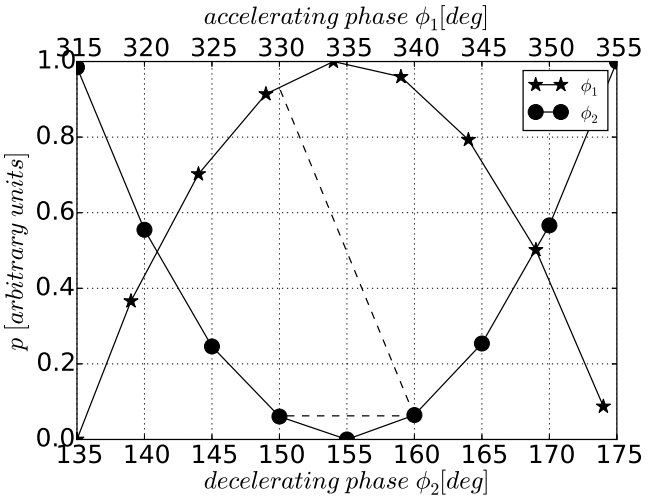

Figure 5.19: The data in fig. 5.18 can be explained by the matching of linac phases. Top: acceleration and deceleration by ϕ_1 and ϕ_2. Momentum units are arbitrary; 1 is maximum acceleration for ϕ_1 and 0 is maximum deceleration for ϕ_2. Data points are extracted from Empirical Model interpolation tables, which are created from Astra tracking. Acceleration and deceleration are reflections of each other with 180° flip, with the cavity imparting maximum effects at 335°/155°. The configuration corresponding to the red solutions is pictured, where the two curves are perfectly out-of-phase and RF effects cancel at all phases for optimal energy recovery. If the loop produces $\Delta\phi \neq 180°$, $E_{dmp} = E_{in}$ can be achieved only at the point at the mirror opposite of the RF crest/trough, illustrated by the diagonally dashed line. In this scenario, $\Delta\phi$ for energy matching depends on the how far ϕ_1 is off-crest. The bottom of fig. 5.18 illustrates this; the black and blue solution sets are non-180° offsets, and crosses the optimal 7.5 MeV/c only at specific ϕ_1.

This suggests a correlation between ϕ_1 and Δt for optimal energy recovery. Eqn. 5.7 is plotted against the solution set with $E_{dmp} = E_{in}$ in fig. 5.20. $E_{dmp} = E_{in}$ is possible at all RF phases if the recirculation loop time-of-flight is tuned correctly.

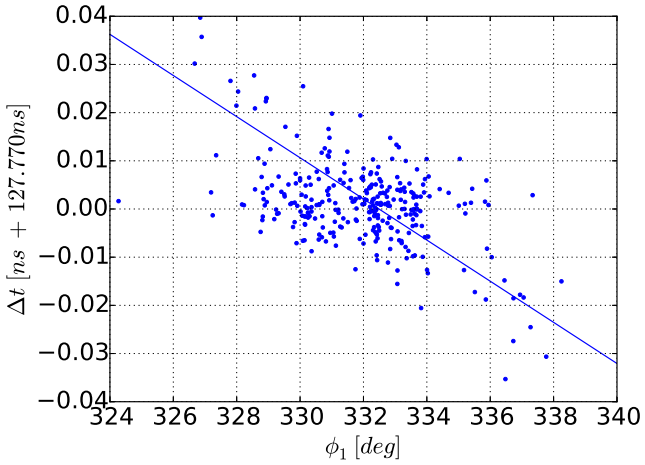

Figure 5.20: Relationship between ϕ_1 and Δt for energy matching to 7.5 MeV/c. $E_{dmp} = E_{in}$ is possible for all initial phases. The fit is given by eqn. 5.7, with the slope being -0.00427 ns/deg. Included are points with p_{dmp} in [7.45, 7.55] MeV/c.

Fig. 5.20 seems to suggest that we can optimally energy recovery at non-180° phase changes.

It is important to note that the optimization was carried out for beam dynamics, and does not include RF loading effects. A previous study [63] has shown that non-180° phase matching has serious consequences for the RF system, including the need for higher RF power, due to beam vectors not cancelling and therefore causing beam loading. Another interesting avenue to pursue is that the linac supports three passes: ERL acceleration and deceleration, and RIB acceleration. 180° may be optimal for ERL operation, but may not be globally optimal when operating simultaneously with RIB. The amount of deviation from 180° that can be tolerated by the system depends on the selection of the loaded Q-value and other RF parameters, and requires further studies beyond the scope of this document.

The optimization platform is capable of finding 180° solutions optimal for RF, as illustrated by the red solution set in fig. 5.18. We intentionally chose not to set a constraint to enforce exactly 180° phase change in the optimization setup, as we wanted to test the capabilities of the platform when we give it minimal guidance, as well as explore the repercussions of non-optimal phase matching, which indeed was illustrated in the tradeoff between energy recovery and energy compression (fig. 5.16). It is interesting that the platform can find unsuspecting dynamics (fig. 5.20) within

the system and demonstrates the advantage of the global approach over piecewise.

Undulator Conditions for Maximizing Gain

We study the effects of beam conditions on the undulator gain. The theoretical gain in an undulator is given by [47]

$$g = -4\sqrt{2}\pi^2 \frac{I_p}{I_A} \frac{K^2[JJ]^2}{(1+K^2/2)^{3/2}} N_u^2 \sqrt{\frac{\lambda}{\lambda_u}} h(\chi) \tag{5.8}$$

where I_A =17045 A is the Alfven current, K is the undulator parameter, $[JJ] = J_0(K^2/(4+2K^2)) - J_1(K^2/(4+2K^2))$ is a Bessel combination, N_u the number of undulator periods, λ the radiation wavelength, λ_u the undulator wavelength, and $h(\chi)$ Madey's function with χ the scaled energy deviation. Of note is the linear dependence on the peak current I_p, which was reproduced by optimization in fig. 5.21. Global optimization shows a maximum bunch compression down to 0.15 mm.

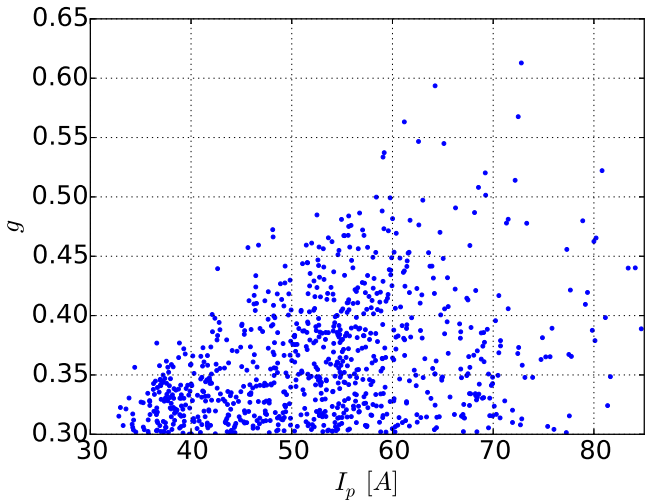

Figure 5.21: Gain increases as peak current I_p increases; this is consistent with FEL theory (eqn. 5.8) and requires compression by the chicane. The upper bound on the gain has the predicted linear functional dependence on I_p. The spread in gain is caused by other beam parameters.

We look at the transverse matching conditions for the undulator, given by [30]. Horizontally, $B_x = 0$ and the undulator resembles a drift, so we

want the incoming beam to be focusing and form a symmetric waist at the undulator center. Vertically, the field is sinusoidal with $B_y \approx B \cos k_u z$. The vertical field can be averaged over the undulator period λ_u to create a section of constant focusing strength $K/(\sqrt{2}\gamma_r \lambda_u)$ where γ_r is the Lorentz factor of the bunch centroid. Thus vertically we'd like a coasting beam. The matching Courant-Snyder parameters are

$$\begin{aligned} \beta_{x,match} &= Z_R + (L_u^2/4Z_R) \\ \alpha_{x,match} &= L_u/(2Z_R) \\ \beta_{y,match} &= \sqrt{2}\gamma_r/(Kk_u) \approx Z_R \\ \alpha_{y,match} &= 0 \end{aligned} \tag{5.9}$$

where Z_R is the Rayleigh length, L_u is the undulator length, and k_u is the undulator wavenumber. Definitions of α and β are presented in Appendix A. Using Z_R =0.5 m, L_u =1 m, the matching conditions can be estimated as

$$\begin{aligned} \beta_{x,match} &= 1 \text{ m} \\ \alpha_{x,match} &= 1 \\ \beta_{y,match} &= 0.5 \text{ m} \\ \alpha_{y,match} &= 0 \end{aligned} \tag{5.10}$$

The matching conditions produced by global optimization can be seen in fig. 5.22, which match closely with values given by eqn. 5.10. 2D plots showing the same matching conditions are shown in fig. 5.23. The global results can be contrasted with results from local (standalone) optimization in fig. 5.24.

Although α and β values agree closely, the gain values suggests an advantage of the global optimization scheme over local optimization. The gain produced by global optimization (fig. 5.22) is higher than the gain from the standalone undulator optimization (fig. 5.24). Global optimization does not impose a limit on the peak current I_p, which is obtained from RF and arc tracking.

On the other hand, local optimization starts at the undulator without the preceding RF and arc sections, thus we require an initial search range for I_p. The search range must be estimated and can impose an arbitrary limit. In particular, the values used to produce fig. 5.24 were based on the Peking University FEL [30]. The TRIUMF arc resulted in a higher I_p, and if the PKU number was used, we would have arbitrarily limited the performance of the undulator.

This is a demonstration of the superiority of the global scheme. If the arbitrary upper limit for I_p is too high, the upstream transport can never produce it. If the limit is low, the performance is suboptimal. Global optimization solves the problem by letting the system dynamically choose the correct I_p for gain and transport.

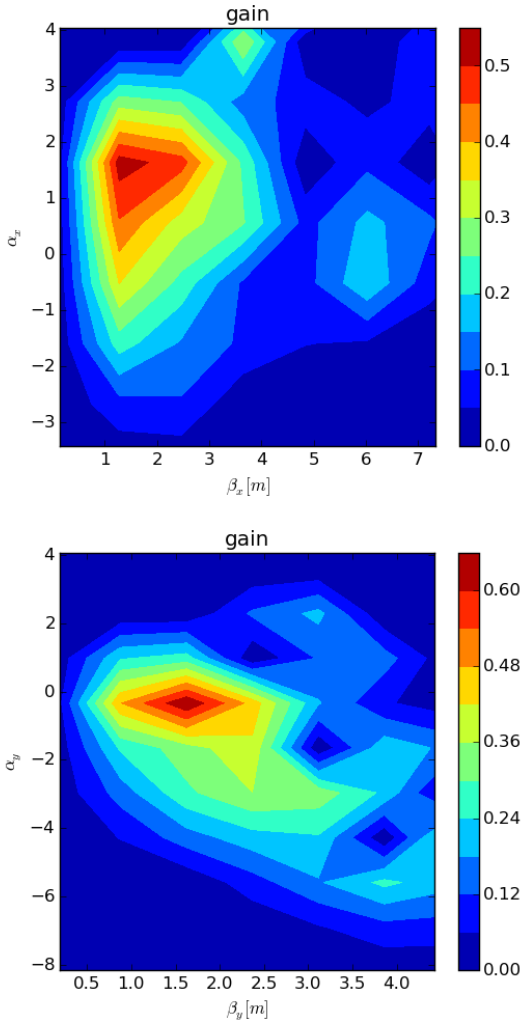

Figure 5.22: Matching conditions of the undulator in horizontal x (top) and vertical y (bottom), as produced by global optimization. The results can be contrasted with the standalone undulator optimization (fig. 5.24), which match closely. The results also match with the theoretical conditions defined by eqn. 5.9.

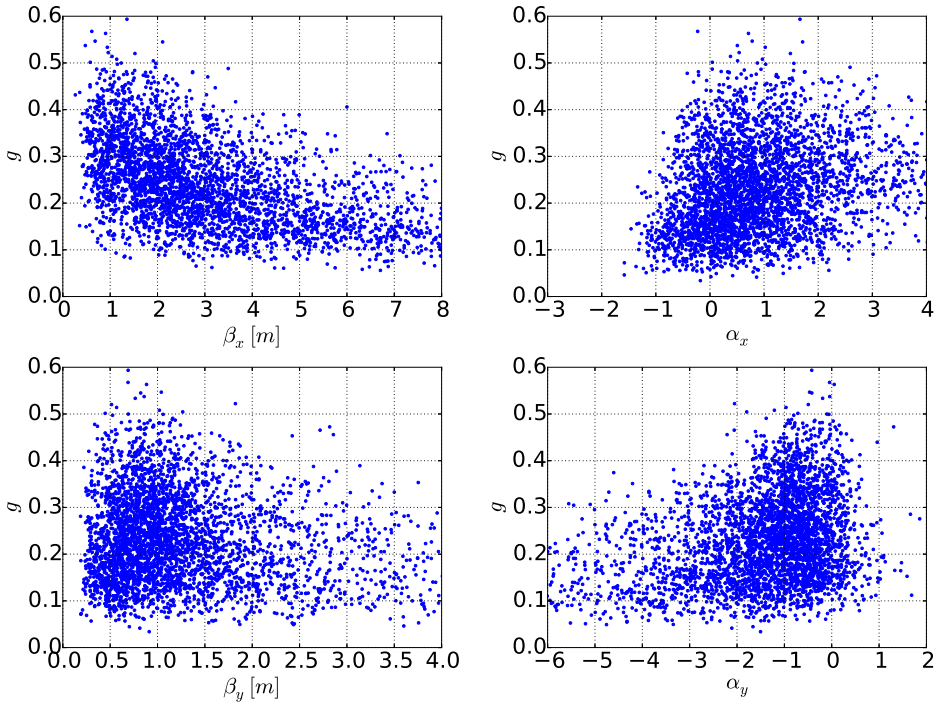

Figure 5.23: Another view of the undulator matching conditions. The data match very well with theory ($\beta_x = 1$ m, $\alpha_x = 1$, $\beta_y = 0.5$ m, $\alpha_y = 0$).

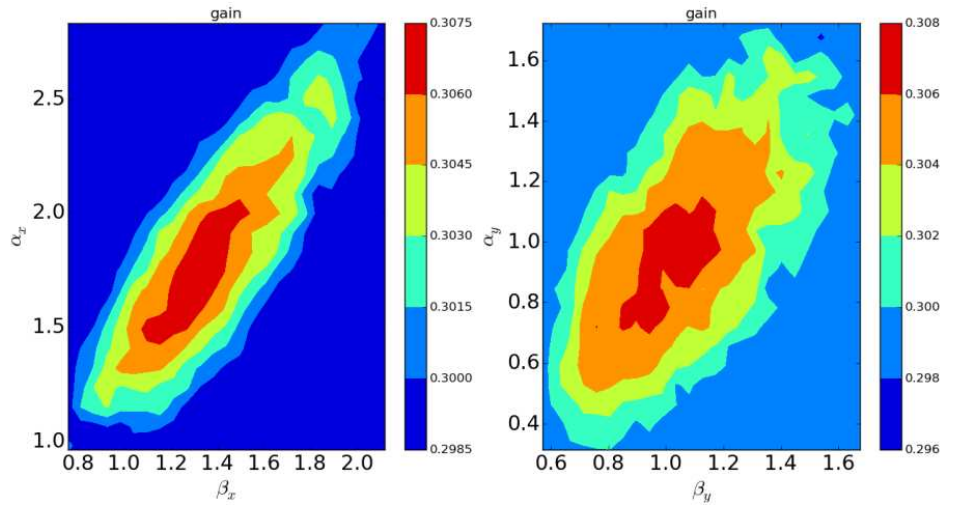

Figure 5.24: Matching conditions of the undulator in horizontal x (left) and vertical y (right), as produced by local (standalone) optimization of Genesis. The results can be contrasted with global optimization (fig. 5.22), which match closely.

The dependence on small β for high gain also sets the focusing conditions of the quads in the FEL matching section. Fig. 5.25 shows strong gradients in the two quads immediately upstream of the undulator. All quads on the lattice are given bipolar freedom and their strengths left to the optimization platform to choose.

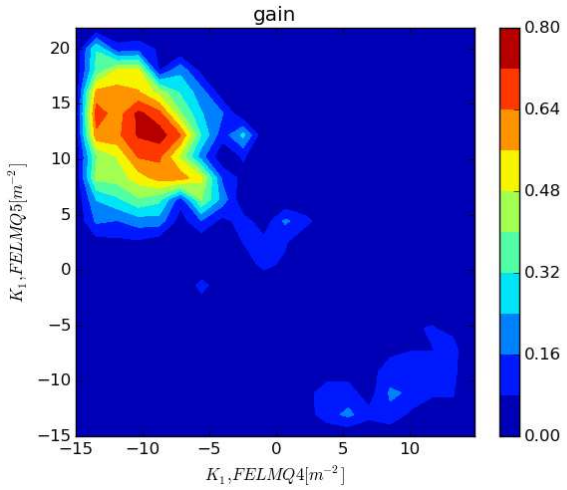

Figure 5.25: Achieving a small beam size requires strong focusing in the FEL matching section. Plotted is K_1 for the matching section quads Q4 and Q5, the two quads prior to the undulator. Global optimization chose Q5 to be horizontally focusing, supporting the theory that the best match for the undulator is a horizontally converging beam (eqn. 5.10). Q4 is horizontally defocusing to create a D0F0 section.

Transverse Dynamics

We look at how transverse objectives such as symmetry can impact beam transport.

The beam size after acceleration is much larger vertically than horizontally (fig. 5.26). As a reminder, we require less than 3 mm RMS beam size at all points.

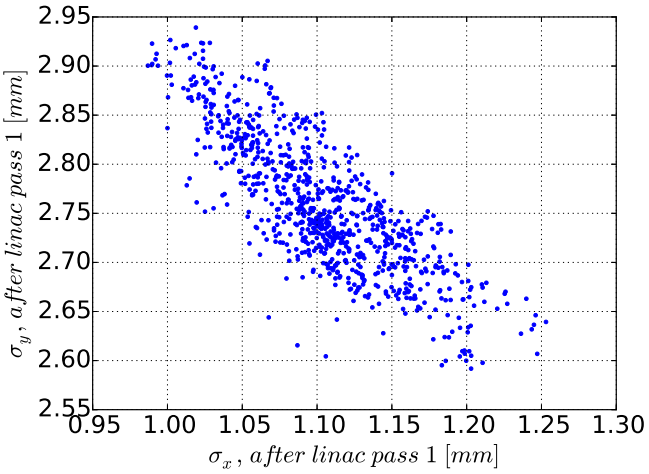

Figure 5.26: Horizontal and vertical RMS beam size after linac pass 1. The vertical size is significantly larger and requires immediate focusing. Points with $g > 0.3$ shown.

This requires the first quad after the linac, EHATQ1, to be vertically focusing (negative K_1). Without immediate vertical focusing, the beam size can easily increase above the acceptable limit of 3 mm (fig. 5.27).

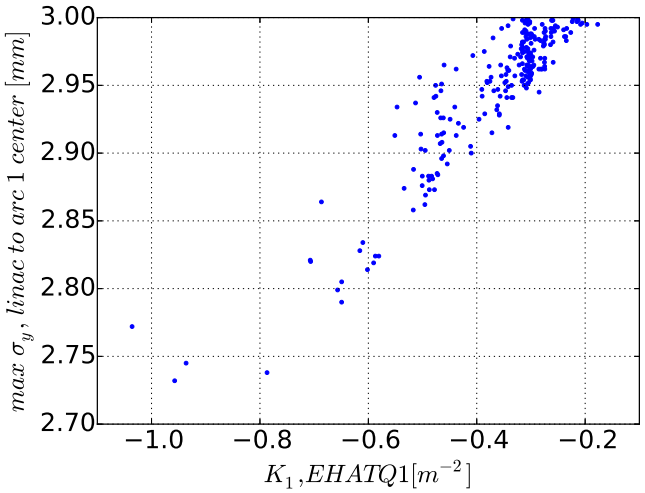

Figure 5.27: Effect of EHATQ1 on vertical beam size in the arc. The Y-axis shows the maximum vertical size from the linac exit to the center of the first arc. Points with $g > 0.3$ shown.

The dump section EDBT consists of the quad EHATQ1, a horizontal bending dipole to kick the beam to the dump transport, followed by a quad doublet (fig. 4.7). Control of horizontal beam size σ_x is critical to proper beam disposal. As was shown previously, the energy spread increases dramatically after deceleration, which is converted to horizontal size. Therefore, other contributions to beam size must be minimized. EHATQ1 cannot be strongly horizontal defocusing. Otherwise, a large beam going into the dipole becomes even larger due to energy spread, and easily violates the beam size condition. Fig. 5.28 shows the effect of EHATQ1 on maximum EDBT σ_x. A bound forms (blue line), in which the more horizontal defocusing EHATQ1 is, the more difficult limiting beam size becomes.

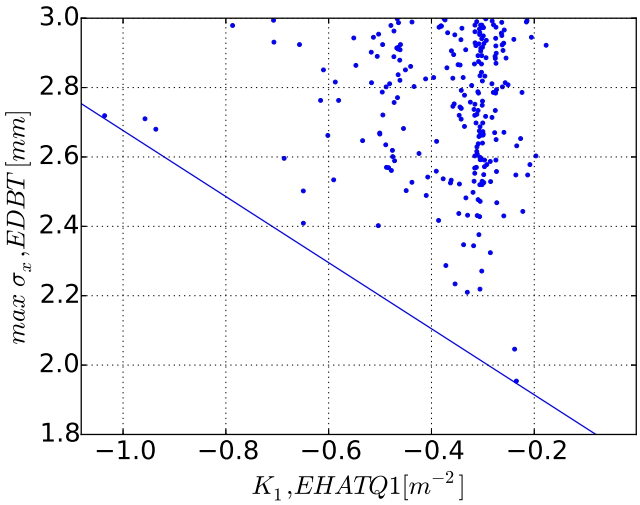

Figure 5.28: EHATQ1 vs max horizontal beam size in EDBT. Too much horizontal defocusing going into the dipole causes problems because we expect the beam size to get even larger after the dipole due to the high energy spread. Points with $g > 0.3$ shown.

The above shows competing effects on the quad EHATQ1.

1. The vertical beam size is large coming out of the linac first pass. Immediate beam size control is required. For vertical focusing, EHATQ1 is preferred to have a negative K_1.

2. For proper dump transport, the beam cannot be too large horizontally. Otherwise, along with the increase in horizontal size from dispersion,

the beam becomes difficult to control. For this reason, EHATQ1 is preferred to have a positive K_1.

The competing objectives between arc beam size and dump beam size forms a Pareto front shown in fig. 5.29. EHATQ1 is pulled in both directions and the optimization platform settles on EHATQ1 being close to neutral. Both figs. 5.27 and 5.28 show that EHATQ1 is concentrated near 0, with a slight negative lean, showing that the arc beam constraint is more urgent.

This tradeoff in beam size should not cause a problem for beam transport as there exist solutions in which the beam size is under 3 mm RMS for both arc 1 and EDBT.

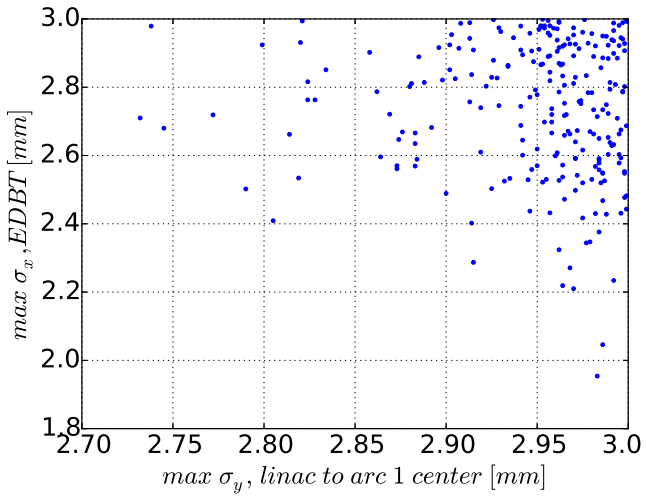

Figure 5.29: Beam size control in arc 1 vs EDBT. Arc 1 vertical beam size forms a Pareto front with EDBT horizontal beam size due to the opposing requirements on the shared quad EHATQ1. Points with $g > 0.3$ shown.

The large vertical beam size out of the linac also has an effect on symmetry. We represent the β_y symmetry by $\alpha_y = 0$ at the arc center. Fig. 5.30 shows that if we want M_{56} to be near the desired value of ≈ -0.1 m for bunch compression, we have to take away from symmetry. To explain this refer to the sample lattice in fig. 5.31. Symmetry requires the points A and A' to be equal. This is impossible, because if A' was made to be as large as A, then it would immediately be made even large by the vertically defocusing chicane, and the beam size constraint is violated.

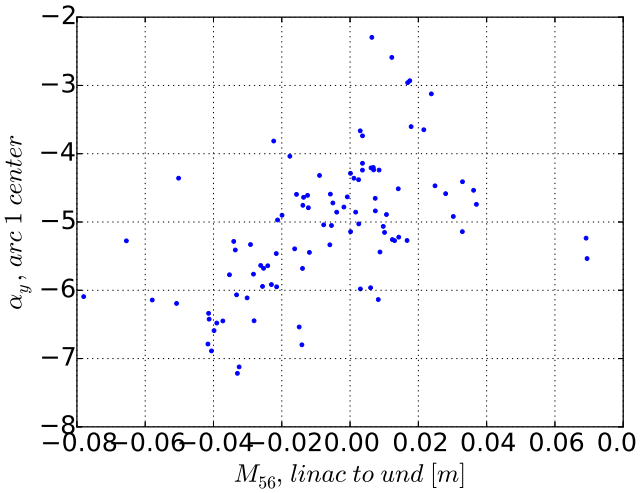

Figure 5.30: M_{56} vs vertical beta function symmetry.

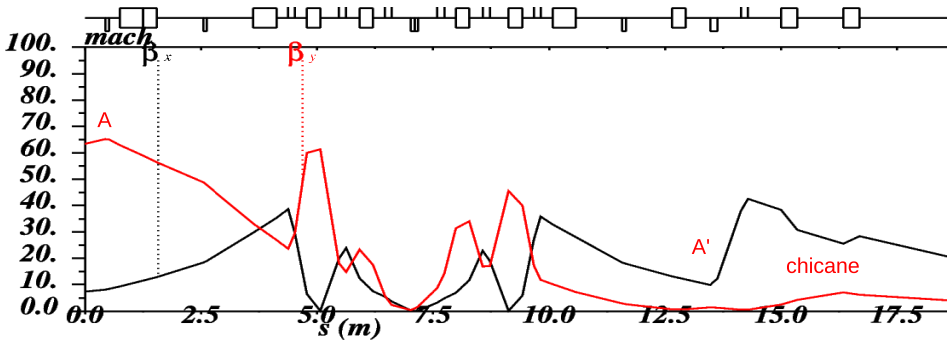

Figure 5.31: β_y (red) symmetry is constrained because of the large initial vertical beam size. If point A' is made the same size as A, the beam size constraint is violated because the chicane is vertically defocusing, thus increases the already large beam size.

This tradeoff between M_{56} and β_y symmetry leads to a Pareto front between β_y symmetry and peak current (fig. 5.32). For high gain operation, vertical symmetry has to be broken. This is acceptable because symmetry is not required for the machine to function (besides, breaking one out of four symmetries is not bad). In principle, this can be fixed by tuning the injector transport to create a different set of beam conditions at the arc.

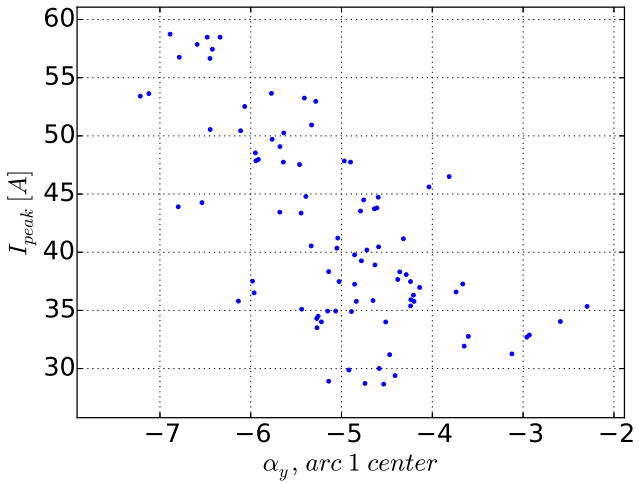

Figure 5.32: A Pareto front forms between the vertical beta function symmetry and peak current at the undulator.

Higher Order Transport Effects

Higher order terms can contribute to linac transport due to the nonlinear RF shape. Fig. 5.33 shows the bunch longitudinal phase space after acceleration with two different energies, 7 MeV and 40 MeV. Neither cases show appreciable higher order effects, and justify using first order envelope tracking in the cavities.

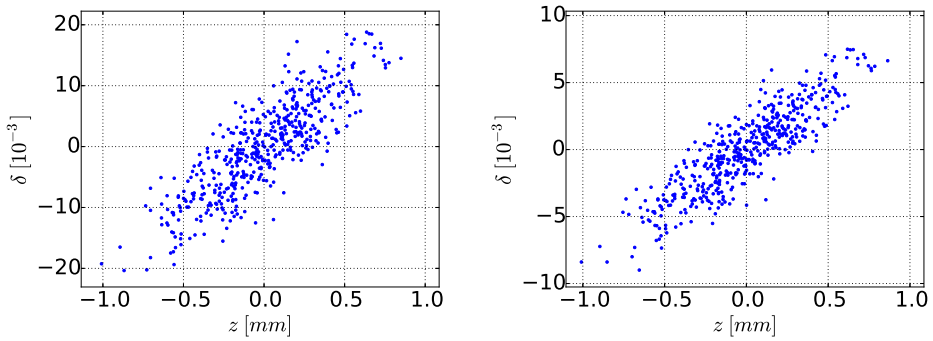

Figure 5.33: Longitudinal phase space after acceleration in cavities 1 (left) and 4 (right). Higher order effects are minimal, therefore first order envelope tracking is sufficient. Tracking performed using the Empirical Model. Note: shown here are the phase spaces for a particular ERL design, not the population.

The effects of transport map element T_{566} is of concern due to dipoles. In arc 1 and the chicane, we are concerned with bunch compression. To compare first and second order contributions to bunch length, define the metric

$$r_{56} = \left| \frac{T_{566}\delta}{M_{56}} \right| \tag{5.11}$$

where δ is the energy spread out of the linac. Fig. 5.35 shows the effect of T_{566}. Fig. 5.34 shows peak current I_p, or inversely, bunch length, as a function of the linear term M_{56}. Maximum I_p, and thus gain, occurs near $M_{56} \approx -0.08$ m, therefore we limit the analysis near this region. This avoids the region $M_{56} = 0$, where eqn. 5.11 breaks down.

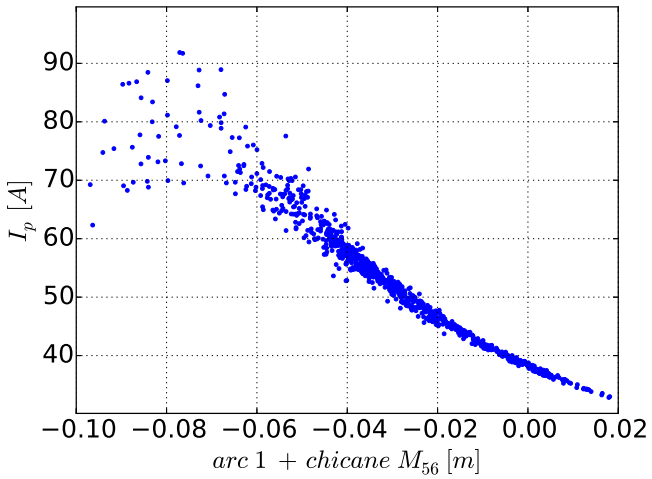

Figure 5.34: Peak current I_p occurs near $M_{56} \approx -.08$ m. The highest gain therefore occurs in the same range. Points shown have $g > 0.3$.

Fig. 5.35 shows that r_{56} is at maximum 0.1, thus linear transport is at least an order of magnitude stronger than the second order term. Translated into bunch length, the second order contribution $T_{566}\delta^2$ is at most 0.026 mm, roughly 20% of the bunch length at the undulator (optimization shows the smallest bunch length is 0.15 mm). Although small, the contribution is not negligible.

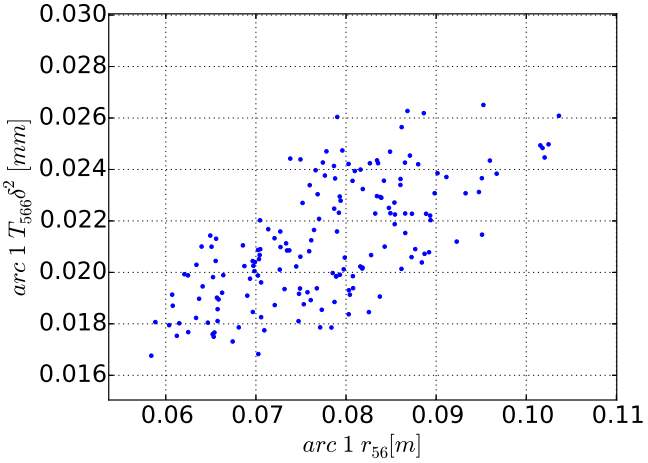

Figure 5.35: The horizontal axis shows the metric r_{56} as defined by eqn. 5.11. Small bunch length is important for achieving high gain (fig. 5.21); the vertical axis shows the second order bunch length contribution $T_{566}\delta^2$. In the M_{56} range [-.1,-.07] m where maximum gain occurs, the maximum value of r_{56} is 0.1, demonstrating that in the optimal gain regime, the linear term dominates. $T_{566}\delta^2$ contributes up to 20% of the bunch length at the undulator (optimization shows compression down to 0.15 mm). Therefore the second order effect, while not dominating, is non-negligible. To isolate the second order effects, points shown have $-.1$ m $< M_{56} < -.07$ m.

Fig. 5.36 shows the direct effect of T_{566} on gain. The plot is bounded by an inverse relationship; large T_{566} leads to larger bunch length, therefore lower gain. Minimizing T_{566} is desirable, but unnecessary, since requesting maximum gain automatically force global optimization to minimize T_{566}.

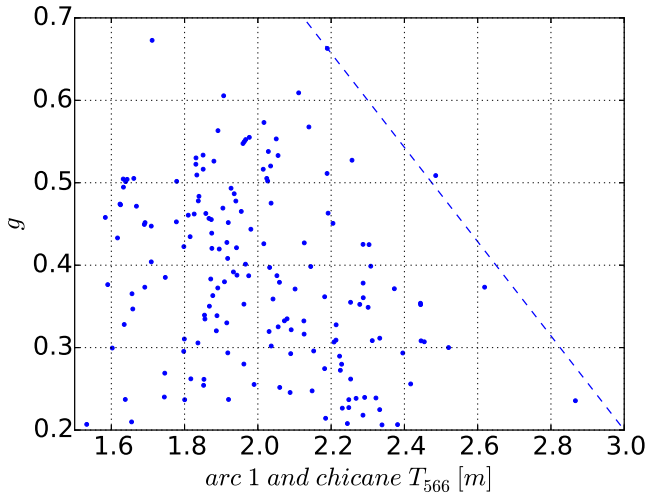

Figure 5.36: Direct effect of T_{566} on gain. Although bunch length at the undulator is primarily determined by M_{56}, T_{566} provides non-negligible contributions. The figure shows gain is bounded by an inverse relationship (albeit fuzzy due to the dependence of gain on other parameters). High T_{566} is not accommodating to high gain. Therefore a criterion for high gain is minimizing T_{566}. This is automatically enforced in the optimization by the gain maximizing objective. Points shown are the same as in fig. 5.35.

In arc 2 we are concerned with the effects of T_{566} and its effect on energy recovery. Fig. 5.37 shows that neither the linear nor the second order term has a dominating effect on energy recovery, i.e. optimal energy recovery is possible for a range of transport schemes. The top figure shows the linear effect. Energy recovery near $M_{56} = 0$ is possible, and eqn. 5.11 is not a useful metric in this scenario.

Energy recovery is not dependent on T_{566}; therefore higher order transport in arc 2 is not a concern. This is in agreement with previous results, which showed that the primary role of arc 2 is timing the bunch to RF phase, the phase being the primary determinant of optimal energy recovery (fig. 5.11).

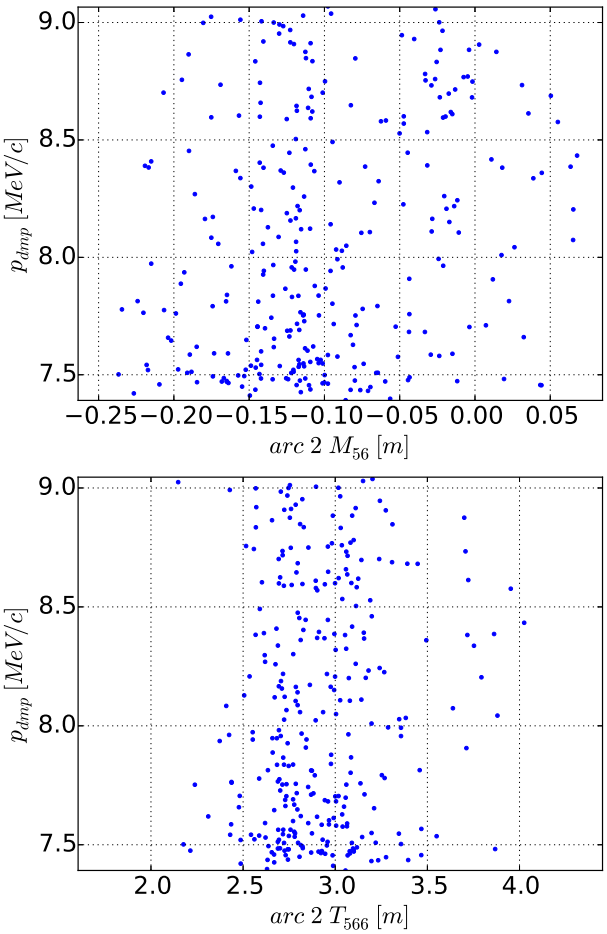

Figure 5.37: The objective of arc 2 is optimal energy recovery. Unlike arc 1, which needed a specific M_{56} to achieve high peak current for lasing, arc 2 can have a range of M_{56} and still achieve optimal energy recovery. Top: M_{56} can cross 0, thus using the metric r_{56} from eqn. 5.11 is not useful. Bottom: effects of T_{566} on energy recovery. Optimal energy recovery is possible for all values of T_{566}, thus T_{566} should not be a concern. The range of T_{566} values shown in the figure spans the entire range of T_{566} explored by the optimization platform, i.e. possible combinations given the initial optimization parameters. There is no indication of a parasitic T_{566} value.

Higher order terms do have an effect on energy spread. Fig. 5.38 shows the effects of arc 2 optics on energy spread δ in the dump line. The top figure displays the typical RF curve, demonstrating that M_{56} affects RF

matching. The bottom figure shows large T_{566} can negatively impact beam disposal by mismatching the bunch to the RF curve, resulting in less than optimal energy compression.

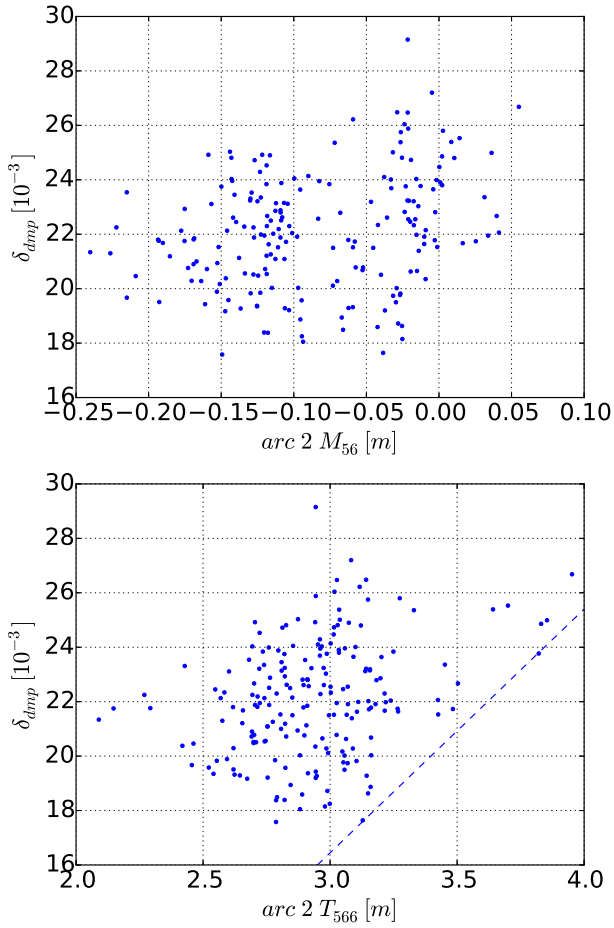

Figure 5.38: Arc 2 transport can shape the energy spread δ at the dump. M_{56} (top) and T_{566} (bottom) shapes how the bunch is matched with the RF curve and therefore influence energy spread compression in the linac. Points with $g > 0.2$ are shown.

Evolution of the Optimization Population

Here we show how the population evolved over the course of the optimization run. This is useful in determining whether the algorithm is working correctly

and finding better Pareto fronts. The state of the run is measured in number of iterations, or generations.

Fig. 5.39 shows the evolution of several key parameters: gain, energy recovered momentum, maximum beam size in EDBT, and the dump energy spread. The run at three different generations are compared. Generation A is near the start of the run, B some time after, and C is the latest generation, or the generation we terminated the run at.

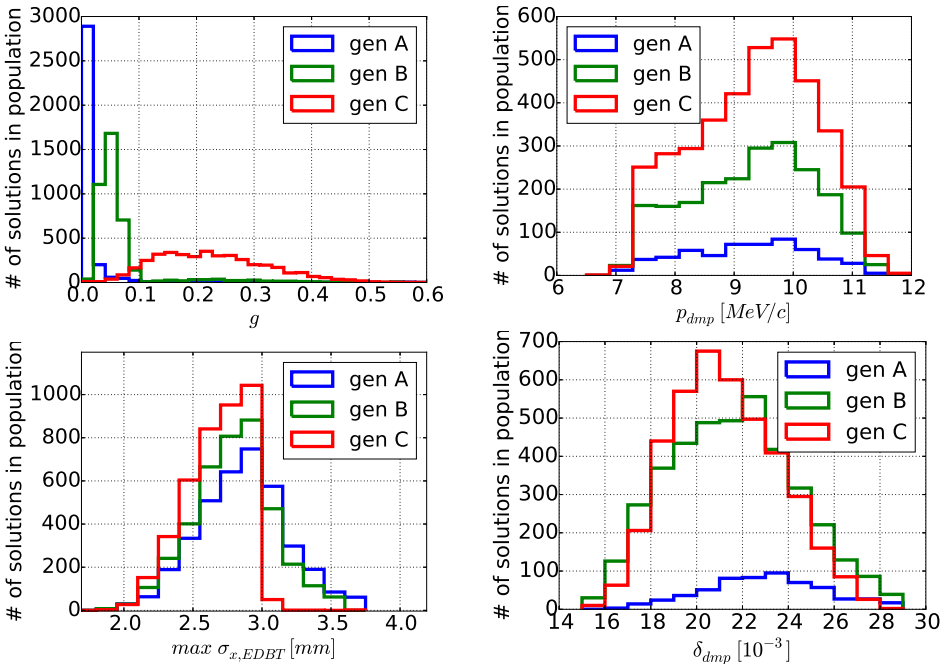

Figure 5.39: Parameter evolution in the optimization population. From top left and clockwise, the parameters are gain, dump momentum, dump energy spread, and maximum horizontal beam size in EDBT.

The plot of gain shows that the population starts with almost all low gain individuals, and the population evolving towards maximizing the gain.

The plot of maximum beam size $\sigma_{x,EDBT}$ shows designs evolving towards the 3 mm boundary in order to satisfy the beam size constraint. In generation A, many designs are larger than the acceptable 3 mm. By generation C almost all are within the 3 mm limit.

The energy spread δ_{dmp} moves toward smaller values in response to the EDBT beam size constraint.

The plot of p_{dmp} shows that the number of designs from $\approx$ 7 MeV/c to

$\approx$ 12 MeV/c are increasing. The number of designs near the optimal 7.5 MeV/c point is increasing because this best satisfies the energy recovery objective. The number of designs above optimality is also increasing due to the tradeoff between p_{dmp} and δ_{dmp}, as fig. 5.15 demonstrates. High p_{dmp} designs also tend to have low δ_{dmp}, thus more likely to satisfy the $\sigma_{x,EDBT}$ constraint.

The optimization run can be completed in 30 days running over 60 computing nodes. The total number of ERL evaluations is roughly one million.

Chapter 6

ERL Baseline Design

This chapter outlines a particular solution taken from the optimization population. The solution obeys all beam transport constraints, taking into account the effects of energy spread and beam scraping in EDBT. The choice was primarily dictated by high gain, followed by energy recovery. Symmetry in Courant-Snyder functions are a nice-to-have.

Design of the ERL beamline is described. Specific information on beam parameters, layout, and coordinates table can be found in Appendix D. The baseline comprises of the following sections (refer to the layout diagram fig. 4.1):

1. cryomodule EACA pass 1 (containing cavities 1 and 2)
2. EABT pass 1
3. cryomodule EACB pass 1 (containing cavities 3 and 4)
4. high energy transport EHAT, including RF separator and septum
5. first arc ARC1
6. mirror septum SEP2
7. chicane CHI
8. undulator matching FELM
9. free electron laser FEL
10. arc 2 matching A2M
11. second arc ARC2
12. merger MERG
13. cryomodule EACA pass 2
14. EABT pass 2
15. cryomodule EACB pass 2
16. ERL beam dump EDBT
17. RIB transport EHAT (alternate path after septum)

The purpose for these sections of the E-linac is to transport and accelerate a 100 pC/bunch ERL beam through EACA-EABT1-EACB from 7.5 MeV to $\approx$ 45 MeV. The beam is transported through the SEP-ARC1-CHI-SEP2-FELM sections to arrive at the undulator FEL. Lasing follows along with slight energy loss. Beam transports through A2M-ARC2-MERG to arrive at the linac. Pass 2 EACA-EABT-EACB decelerates the beam to 7.5

MeV and beam is disposed in EDBT.

A separate 16 pC/bunch beam accelerates through EACA-EABT-EACB-SEP-EHAT to RIB photofission target from 10 MeV to ≈ 50 MeV.

Optimization Parameters for the Baseline

Top level parameters are listed below. Gain is defined as $(\mathrm{d}P/\mathrm{d}z)/P$, where P is the radiation power.

Table 6.1: ERL baseline parameters.

Parameter	Value
Gain	0.5 m^{-1}
Initial momentum	7.5 MeV
EDBT momentum	7.7 MeV
σ_x	$\leq$ 3 mm everywhere
σ_y	$\leq$ 3 mm everywhere
EDBT energy spread	0.029
EDBT max σ_x	3.0 mm
EDBT max σ_y	1.9 mm
Dump σ_x	5.5 mm
Dump σ_y	6.0 mm
Beam loss	$\leq 10^{-5}$

The design has a gain of 0.5 m^{-1}. This is near the top of the optimization search space for gain and satisfies our maximize lasing objective.

The EDBT max σ_x is within our constraints, demonstrating that energy spread is contained and should not be an issue.

Beam Transport

Here we show the baseline optics functions (see Appendix A for definitions) of the ERL recirculation lattice. The transport solution obeys beam size constraints everywhere, including beam loss $\leq 10^{-5}$. A detailed lattice layout coordinates table is shown in table D.8 of Appendix D.

Optics functions of EHAT to FEL is shown in fig. 6.1. From FEL to MERG is shown in fig. 6.2. EDBT is shown in fig. 6.3.

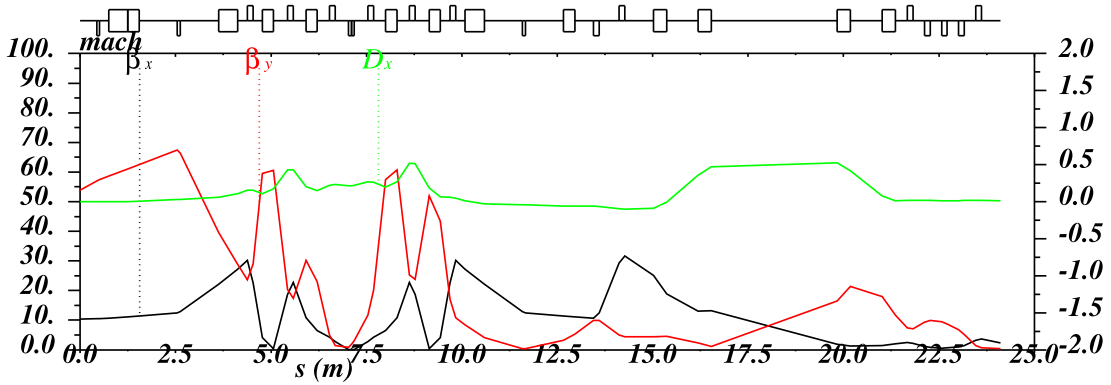

Figure 6.1: Linac to undulator transport. Left axis is beta functions (m), right axis is dispersion (m).

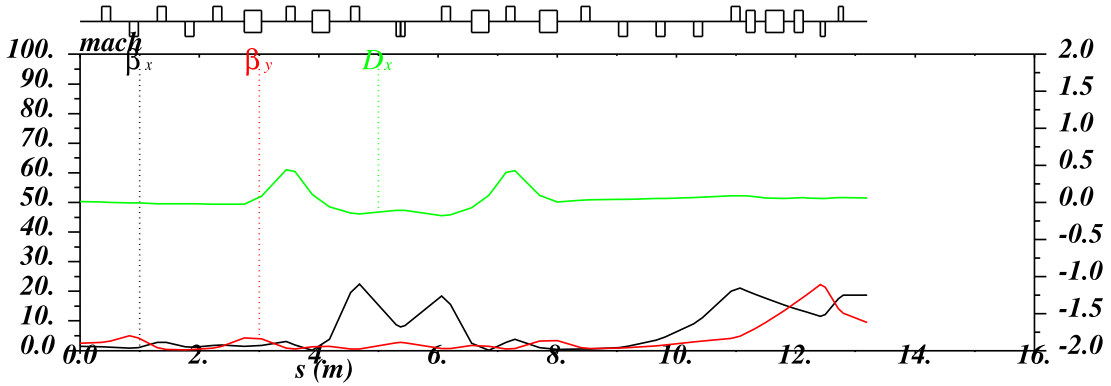

Figure 6.2: Undulator to linac pass 2 transport. Left axis is beta functions (m), right axis is dispersion (m).

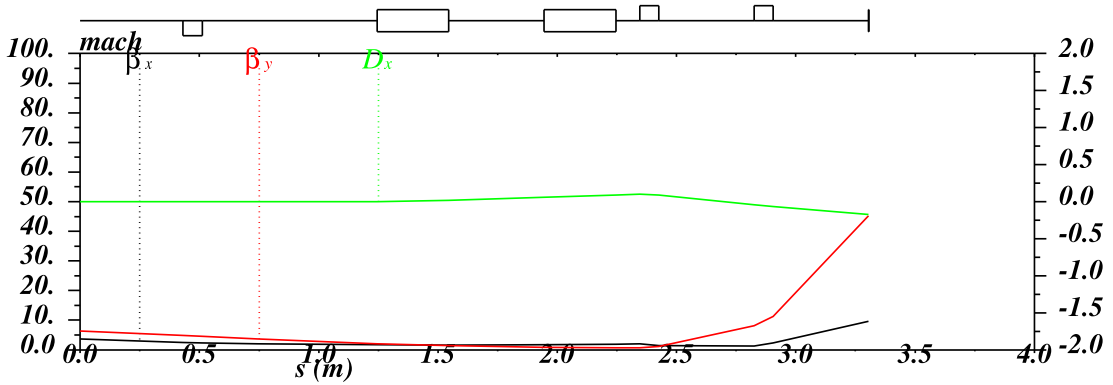

Figure 6.3: EDBT transport. Left axis is beta functions (m), right axis is dispersion (m).

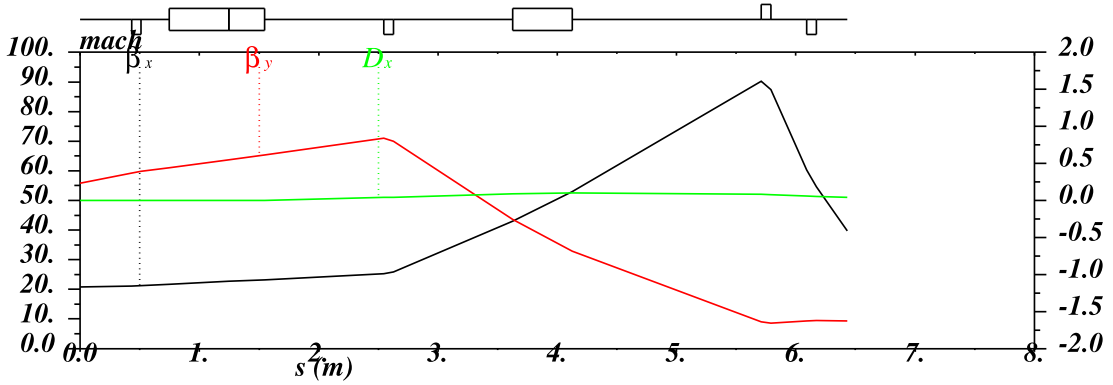

Figure 6.4: EHAT transport for RIB. Left axis is beta functions (m), right axis is dispersion (m).

ERL Compatibility with RIB

ERL operations do not interfere with RIB operations. The simultaneous RIB transport solution in EHAT is shown in fig. 6.4.

ERL Compatibility with Energy Doubling

The first phase of ERL construction may see the possibility of building a bare recirculation lattice without the undulator. We wish to know whether the lattice can be used in energy doubling mode, labeled Recirculating Linear Accelerator (RLA) mode, before the undulator is added and full FEL operation begin. This was not part of the original optimization requirements, thus no provisions were taken to include it in the optimization. In principle, RLA can be added to the optimization in a similar manner as RIB.

RLA operates by transporting the bunch to the linac pass 2, but with 0° phase change from pass 1, for acceleration in both passes. In contract, ERL requires 180° phase change. Therefore, compatibility with energy doubling requires that the ERL recirculation lattice can provide half an RF wavelength λ of freedom in path length.

We begin by examining whether the chicane can provide this freedom. We add $0.25\lambda \approx 6$ cm to each of the first and third drift lengths L (see fig. 6.5). This results in a horizontal offset of 14 cm in the second drift between the ERL and RLA beam trajectories. This is too much for the beam pipe and thus the chicane cannot provide the path length difference.

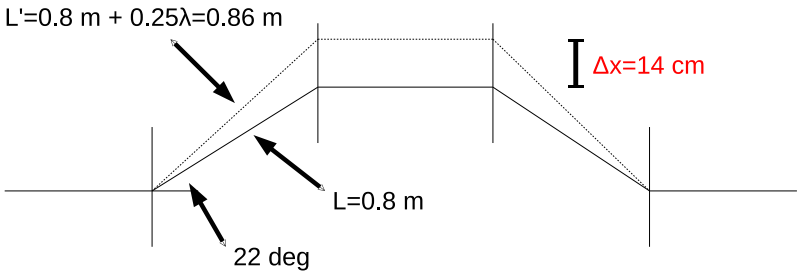

Figure 6.5: Chicane in ERL mode (straight line) and RLA mode (dashed line).

We examine a second possibility of replacing the chicane with a drift. The path length reduced is 2×6 cm $\approx 0.5\lambda$ (see fig. 6.6), which is roughly

the amount required for the phase change. Further small changes can be performed by tuning M_{56} of the arcs. Thus if RLA operation is desired before the full ERL is operational, we recommend leaving out the chicane in the first phase of the recirculation loop construction and replacing it with a drift.

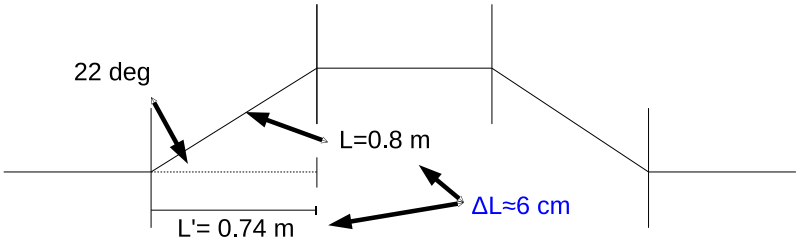

Figure 6.6: Effect of removing chicane on path length.

Chapter 7

Conclusions

This work resulted in the creation of a generic optimization platform and the creation of a baseline ERL. This is an exciting development for it not only is the first ERL in Canada, but the first combined functions ERL and RIB accelerator in the world, and highlights Canadian achievements in accelerator physics. The baseline design is important for two reasons:

1. A complete list of elements and coordinates of the ERL lattice was provided. It lays the foundation for the upgrade of the TRIUMF E-linac to a light source. Clear relationships and tradeoffs were shown regarding this specific design. Empirical relationships, such as that of energy recovery and dump energy spread were derived and of great value to physicists when the full ERL design is underway. In addition, all optical elements were designed in accordance with existing magnet designs. This simplifies the design and does not require new studies in alternative magnets. The baseline also conforms to engineering constraints. Space is set for support technology and diagnostic devices to be inserted.

2. The wider importance of the work is that it offers a study of a general class of ERLs. The physical processes of gain, recirculation, energy recovery, and beam disposal are present in all ERLs in the world. Even physically larger ERLs with more energetic beams encounter the same issues, as can be observed in the Continuous Electron Beam Accelerator Facility (CEBAF) at Jefferson Lab. The interplay between these varied objectives were empirically demonstrated in the first start-to-end ERL optimization, and can be applied to any machine with a linac driver, a recirculation loop with a free electron laser, and energy recovery.

Some dynamic relationships that arose from the start-to-end ERL study are:

1. Lasing increases energy spread, and therefore complicates beam disposal.

2. Tradeoff between energy recovery and energy spread. We cannot have the best of both worlds.

3. Energy recovery is affected by the interplay between RF phase and recirculation time. We showed that the commonly held assumption of a 180° phase change is not the only solution for optimal energy recovery.

4. Lasing complicates beam transport in the return arc, and can result in beam loss. In the TRIUMF ERL layout, this beam loss can be mitigated by good arc optics.

5. ERL operation is compatible with RIB operation.

6. High demand on optics in certain areas, such as EHAT.

Some of the listed items are conceptually known processes, but the significance of the optimization results is that they showed quantitatively how much these constraints and limitations matter, and whether they can be circumvented or avoided in the TRIUMF machine. The question of whether a transport solution exists for the given E-linac ERL layout was answered and presented. The lattice layout proved sufficient and demonstrated that the optimization platform is a viable way of accelerator study and design.

Additionally, the optimization platform was demonstrated to be a valuable computational tool for physicists, both as an optimization tool and as a tool for studying global dynamic relationships in machines that was not possible previously. The ability to add or change modeling engines for studying different processes contributes to its versatility and has never been performed before in the field of accelerator physics. No precedence for this type of tool existed. Either genetic optimizations were performed in specific sections, such as injector optimization in the case of APISA, or an engine needed to be created to encompass all the necessary physics. The TRIUMF platform allows for global optimization with only the choice of the right modeling engines for the different physical processes needed to be included, and many engines exist in accelerator physics, each catering to its niche.

The optimization platform was built with rigorous software engineering standards with the intentions of versatility and reusability. Indeed, application to other problems is already on the way. Collaboration with S. Dechoudhury from the Variable Energy Cyclotron Centre (VECC) in Calcutta is in progress to optimize an injector linac in India. The problem requires two modeling engines, one is the accelerator code TRACK, and the other is a custom written Fortran engine. The custom written engine calculates

drift lengths from initial cavity phases, and then passes the drift lengths to TRACK for tracking. Neither engines were used for the TRIUMF ERL optimization, but were easily implemented into the platform for the VECC problem. This would have been impossible without the extensibility of the platform or its multi-engine feature, and mitigated the need for VECC to write a custom optimizer or tracking tool.

The VECC problem was set up within two hours and put on a computational cluster. The ease of transiting the platform into a parallel environment is a great advantage, as large optimization problems are computationally intensive. The software design includes flexible and powerful exception handling mechanisms, which makes the platform start-and-forget, allowing the physicist to carry on with other works without the need for constant intervention and monitoring.

Another work in progress is the optimization of a compression chicane for the Deutsches Elektronen-Synchrotron facility in Germany. The project models energy loss in a chicane for a 1 GeV electron beam, using the engine CSRtrack, with a second custom built engine that automatically generates a bunch distribution given initial envelope parameters. A third vertex using MADX runs in parallel to the first two, to calculate the transport map elements of the chicane. This is another demonstration of the versatility of the optimization platform.

The creation of the Empirical Model for ERL linac modeling is an incidental benefit. The design of the software makes it very useful for applications outside of the optimization platform, where its fast running time is suitable for online applications. A port of Empirical Model was integrated into XAL, a Java framework for creating high level applications for accelerators. XAL was already used for the first rounds of E-linac commissioning and proved to be invaluable. This demonstrates value of the optimization project outside of optimization.

The optimization was designed as a beam dynamics study. Not included are RF effects such as beam loading. Certain results, such as changing the arc path length, and therefore time-of-flight, for energy recovery, need to be checked for compatibility with the RF system. When the acceleration and deceleration phases are not 180° apart, less than optimal klystron operation follows. In principle, RF loading can be added as another vertex in the optimization. While optimal RF operations preferring 180° phase difference, a non-180° phase difference could be better for beam dynamics and energy compression. The results can be interesting and warrants further study.

A limitation of the platform is its inability to perform discrete optimization. For example, what is the optimal number of dipole magnets in an arc?

The produced baseline design can claim to be a good design with four-dipole arcs, but cannot claim to be unequivocally better than all designs with three-dipole arcs. Although the choice of four dipoles is intuitively better than three dipoles and the decision is physically sound, future questions may arise regarding discreteness that are not as easily answered.

Genetic algorithms can only perform optimization on continuous variables. Therefore, answering questions regarding discreteness require either algorithmic changes, or new features implemented into the platform. One such feature that was discussed but not realized was the "prototype" feature. This allows the comparison of top level parameters such as final energy and emittances, while allowing for different arc layouts, or prototypes. We feel this is the next stage in computational optimization in the field of accelerator physics.

Bibliography

[1] Boost C++ Libraries. http://www.boost.org/. Accessed: 2015-09-04.

[2] CSRtrack. http://www.desy.de/xfel-beam/csrtrack/.

[3] Git. https://git-scm.com/. Accessed: 2015-09-04.

[4] Madx Quadrupole. MAD-XUserGuide-Quadrupole.

[5] NumPy. https://www.numpy.org. Accessed: 2015-09-05.

[6] The XML C Parser and Toolkit of Gnome libxml. http://www.xmlsoft.org/. Accessed: 2015-09-04.

[7] VariantCalc. http://boost-spirit.com/repository/. Accessed: 2015-09-05.

[8] WestGrid. https://www.westgrid.ca. Accessed: 2015-09-03.

[9] *Users Guide to the Program DIMAD*, 2004. SLAC-R-285.

[10] Open XAL. http://xaldev.sourceforge.net/, 2015.

[11] R. Baartman. The Buckley Quads: Hysteresis, Calibration. *Unpublished*, Dec 2013.

[12] I. Bazarov. bi - Beam Instability BBU Code. http://www.lepp.cornell.edu/~ib38/bbucode/src.

[13] I.V. Bazarov and C.K. Sinclair. Multivariate Optimization of a High Brightness DC Gun Photoinjector. *Phys. Rev. ST Accel. Beams*, Mar 2005.

[14] Bazarov, I. APISA. http://www.lepp.cornell.edu/~ib38/apisa/. Accessed: 2015-09-04.

[15] Ben-Zvi, I. and Kayran, D. and Litvinenko, V. High Average Power Optical FEL Amplifiers. In *Proceedings of The 27th International Free Electron Laser Conference*, Aug 2005.

[16] Bertsimas, D. and Tsitsiklis, J.N. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[17] Vyacheslav A. Buts, Andrey N. Lebedev, and V.I. Kurilko. *The Theory of Coherent Radiation by Intense Electron Beams*. Springer, 2006.

[18] Brown K.L. Carey, D.C. and Rothacker F. *THIRD-ORDER TRANSPORT*, 1995.

[19] David C. Carey. *THE OPTICS OF CHARGED PARTICLE BEAMS*. 1987.

[20] CERN. MAD - Methodical Accelerator Design. `http://madx.web.cern.ch/madx`, 2015.

[21] Y.C. Chao. Prib injector distribution in z.

[22] Y.C. Chao. RIB Injector Distribution in X. Unpublished.

[23] Y.C. Chao. RIB Injector Distribution in Y. Unpublished.

[24] Y.C. Chao. Baseline Pass 2 ERL Design. E-linac Optim transport tune, 2012.

[25] Y.C. Chao. Baseline Pass 2 ERL Design. E-linac Optim transport tune, 2012.

[26] Y.C. Chao. Baseline RIB Extraction Design. E-linac Optim transport tune, 2012.

[27] Chao, Y.C. *Empirical Model Interpolation Tables*, 2013. Unpublished.

[28] Chao, Y.C. *Recipe for Empirical Model Propagation*, 2013. Unpublished.

[29] Chao, Y.C. E Linac EMBT-EABT-EHAT Phase One Major Components and Layout. *Design Note TRI-DN-12-03*, Mar 2014.

[30] Chao, Y.C. *Wiggler Parameters and Matching Conditions*, 2014. Unpublished.

[31] Chao, Y.C. et al. Low-Beta Empirical Models used in Online Modeling and High Level Applications. *Proceedings of the 2011 International Particle Accelerator Conference*, Sep 2011.

[32] J. A. Clarke. *The Science and Technology of Undulators and Wigglers*. 2004.

[33] S. Dechoudhury. Optimizing the VECC Linac. 2015.

[34] Y.T. et al. Ding. Study on the planar undulator scheme with focusing properties for PKU-FEL. In *Free electron lasers. Proceedings, 26th International Conference, FEL04, and 11th FEL Users Workshop*, Aug 2004.

[35] Dohlus, M and Limberg, T. CSRtrack: faster calculation of 3D CSR effects. 2004.

[36] Bleuler S. et al. PISA — A Platform and Programming Language Independent Interface for Search Algorithms. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, pages 494 – 508, Berlin, 2003. Springer.

[37] K. Floettmann. A Space Charge Tracking Algorithm. `http://www.desy.de/~mpyflo/`, 2015.

[38] R. Garoby. Beam Loading in RF Cavities. In *Frontiers of Particle Beams: Intensity Limitations*, pages 509–541. Springer, 1992.

[39] Gong, C. *CSR Optimization in Chicane*, 2013. Unpublished.

[40] Gong, C. *Empirical Model Design Document*, 2013. Unpublished.

[41] Gong, C. and Chao, Y.C. The TRIUMF Optimization Platform and Application to the E-linac Injector. *Proceedings of ICAP12*, Aug 2012.

[42] Hassan, R. et al. A Comparison of Particle Swarm Optimization and the Genetic Algorithm. *Proceedings of the 46th Structures, Structural Dynamics, and Material Conference*, 2004.

[43] Herman, W. Fourth Generation Light Sources. May 1997.

[44] Georg H. Hoffstaetter and Ivan V. Bazarov. Beam-Breakup Instability Theory for Energy Recovery Linacs. *Phys. Rev. ST Accel. Beams*, 7, May 2004.

[45] Holder, D. J. et al. The Status of the Daresbury Energy Recovery Linac Prototype. *Particle accelerator. Proceedings, 11th European Conference, EPAC08*, Jun 2008.

[46] Jones, K. Comparison of Genetic Algorithm and Particle Swarm Optimisation. *Proceedings of CompSysTech2005*, 2005.

[47] Kwang-Je Kim, Zhirong Huang, and Ryan Lindberg. *Introduction to the Physics of Free Electron Lasers*. Jun 2010. Unpublished.

[48] P. Kolb. private communication, 2015.

[49] P. Kolb. HOM Data. private communication, 2015.

[50] Kolb, P. HOM Considerations for the TRIUMF eLINAC. Presented at the HOM Workshop 2012, 2012.

[51] S. Koscielniak. ARIEL and E-linac Conceptual Design Report. 2008.

[52] R. Laxdal. private communication, 2015.

[53] M. et al. Liepe. Pushing the Limits: RF Field Control at High Loaded Q. In *Proceedings of the 2005 Particle Accelerator Conference*, May 2005.

[54] C.Y. Liu, K. Geoffrey, and G.M. Wang. Performance Evaluation of Undulator Radiation at CEBAF. *Proceedings, 1st International Particle Accelerator Conference (IPAC10)*, 2010.

[55] H. H. et al. Lu. Research on the undulator used for PKU-FEL. In *Free electron lasers. Proceedings, 26th International Conference, FEL04, and 11th FEL Users Workshop*, Aug 2004.

[56] M. et al. Marchetto. Beam Dynamics Optimization of the TRIUMF Elinac Injector. In *Particle accelerator. Proceedings, 23rd Conference, PAC'09*, May 2009.

[57] G. Marcus. private communication, 2015.

[58] Merminga, L. Presented at the 2002 Electron Ion Collider Accelerator Workshop, Feb.

[59] Merminga, L. Energy Recovery Linacs. In *Proceedings of the 2007 Particle Accelerator Conference*, Jun 2007.

[60] B.D. et al. Muratori. Space Charge Effects for the ERL Prototype Injector Line at Daresbury Laboratory. In *Proceedings of the 2005 Particle Accelerator Conference*, pages 1676–1678, May 2005.

[61] Onuki, H. and Elleaume, P. *Undulators, Wigglers and Their Applications*. CRC Press, 2003.

[62] P. Piot, D.R. Douglas, and G.A. Krafft. Longitudinal Phase Space Manipulation in Energy Recovering Linac-Driven Free-Electron Lasers. *Physical Review Special Topics-Accelerators and Beams*, Oct 2003.

[63] Powers, T. and Tennant, C. Implications of Incomplete Energy Recovery in SRF-Based Energy Recovery Linacs. In *Proceedings of the 41st Advanced ICFA Beam Dynamics Workshop on Energy Recovery Linacs*, May 2007.

[64] R. E. Rand. *RECIRCULATING ELECTRON ACCELERATORS*. 1984.

[65] S. Reiche. *GENESIS 1.3 User Manual*, 2004.

[66] T. Satogata. More Lattice Optics and Insertions. USPAS Accelerator Physics, 2011.

[67] Smith, S.L. The Status of the Daresbury Energy Recovery Linac Prototype ERLP). *Proceedings of ERL07*, May 2007.

[68] Planche T. ARIEL S34 Dipoles design note. *Unpublished*, Apr 2013.

[69] Planche T. ARIEL 'Y30' Dipoles design note. *Unpublished*, Jul 2013.

[70] Thompson, N. Introduction to Free-Electron Lasers, 1997.

[71] Vu, V. T. A Comparison of Particle Swarm Optimization and Differential Evolution. *International Journal on Soft Computing*, Aug 2012.

[72] G.M. et al. Wang. Energy recovery transport design for PKU FEL. *Particle accelerator. Proceedings, 22nd Conference, PAC07*, Jun 2007.

[73] L.Y. Yang, Y.J. Li, W.M. Guo, and S. Krinsky. Multiobjective Optimization of Dynamic Aperture. *Phys. Rev. ST Accel. Beams*, May 2011.

[74] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report, 2001.

[75] E. et al. Zitzler. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evol. Comput.*, 8(2):173–195, June 2000.

Appendix A

Notations

Accelerator physics notations used in this document conform to TRANSPORT notations [18].

A particle is defined by the six coordinates x, x', y, y', z, and δ, where $x' = p_x/p$ is the x-angle defined by the ratios of electron momentum to bunch centroid momentum, $y' = p_y/p$ is similarly the y-angle, and $\delta = (p_z - p)/p$ is the energy deviation of the particle to the centroid. $z = -ct$ is the time difference compared to the time of the bunch centroid, scaled by the negative speed of light $-c$. Positive z denotes that the particle arrives at a location before the centroid, in the centroid frame. The six coordinates can also be labeled from 1 to 6, or x_i where $i = 1, ..., 6$. The RMS values of the coordinates are defined by σ, for example, σ_x.

The particle energy is represented by $E = \gamma mc^2$, where γ is the particle's Lorentz factor and m is the electron rest mass. The bunch centroid energy is represented by E_r and correspondingly γ_r.

RMS bunch Courant-Snyder (CS) parameters are defined in the usual manner:

$$\begin{aligned} \varepsilon_{x,rms} &= \sqrt{\langle x^2\rangle\langle x'^2\rangle - \langle xx'\rangle^2} \\ \beta_{x,rms} &= \langle x^2\rangle/\varepsilon_{x,rms} \\ \alpha_{x,rms} &= -\langle xx'\rangle/\varepsilon_{x,rms} \end{aligned} \tag{A.1}$$

where $\langle\rangle$ denotes the mean, ε is the emittance, β is related to the beam size, and α is related to the beam tilt. Similar equations follow for y and z.

The transfer map of a lattice follows typical accelerator convention. The second order transport is given by the Taylor expansion

$$x_i = \sum_j M_{ij} x_j^0 + \sum_{j,k} T_{ijk} x_j^0 x_k^0 \tag{A.2}$$

where x_i^0 denotes the coordinates before the map is applied, and x_i after. The first order map elements are given by $M_{ij} = \partial x_i/\partial x_j^0$, and the second order elements are given by $T_{ijk} = (1/2)\partial^2 x_i/\partial x_j^0 \partial x_k^0$. For example, the map element M_{12} denotes how a beamline section affects the x coordinate given the x' coordinate of the particle at the start of the section.

Dispersion η_x is defined as, for an off-momentum particle with energy deviation δ, $\eta_x\delta$ is the x deviation of the off-momentum closed orbit from the reference orbit. Similarly for dispersion prime η'_x, $\eta'_x\delta$ is the x' deviation. Identical definitions exist for y and y'.

RF phases are defined using ϕ_{ij}, where i is the linac pass number, and j is the cavity number. i can range from 1 to 3, with 1 and 2 representing the acceleration and deceleration passes of the ERL beam, and 3 representing the acceleration pass of the rare isotope beam. j can range from 1 to 4, representing the four 9-cell cavities of the linac. If only one subscript is displayed, e.g., ϕ_2, the subscript represents the index i, with the cavity number j assumed to be 1.

The phase ϕ_{ij} is the phase (in degrees $^\circ$) of the jth cavity when the bunch centroid is at the jth cavity entrance during pass i. We choose to use the cavity entrance phase so as to conform to the Empirical Model convention (see section 3). Sometimes, ϕ_{ij0} is used, denoting the initial phase of the jth cavity at the beginning of machine modeling, for pass i.

FEL literature [47] typically replaces the particle coordinate z with the ponderomotive phase $\theta = (k + k_u)r - w\tau + \text{const}$, where k and w are the wavenumber and angular frequency of the FEL radiation, k_u is the wavenumber of the undulator periodicity, r is the position in the undulator, and τ is the time an electron arrives at r. The other five coordinates remain identical.

Quad strengths are given in units of K_1 in units of [m^{-2}], defined by [4, 19]

$$K_1 = \frac{1}{B\rho}\frac{\partial B_y}{\partial x} \tag{A.3}$$

where $B\rho$ is the beam rigidity and $\partial B_y/\partial x$ is the quad gradient.

Appendix B

Software Design of the Optimization Platform

Introduction

The goal of the TRIUMF optimization project is to study the TRIUMF ERL design, although the input format is flexible enough to be easily extended to other optimization problems. The code executes on parallel machines and can easily switch between different physics engines.

The TRIUMF optimization framework is based upon APISA [14], which is a realization of PISA [36], an interface designed for the global optimization of multiple objectives based on pseudo-random sampling algorithms. The genetic algorithm SPEA2 [74] is chosen, which has precedence of prior applications to accelerator physics [13, 73]. A genetic algorithm is a population based algorithm which solves the optimization problem using analogues to processes from evolutionary biology:

1. Generate random starting population, e.g., a set of ERL designs each with different design parameters.

2. Evaluate fitness of each individual, or ERL design, in the population. An ERL design which better satisfies the objectives and constraints is more fit.

3. Stochastically choose individuals (weighed by fitness), mutating or recombining them to form new individuals.

4. Throw away some individuals (designs with worst fitness more likely to be thrown away). Add new individuals added to the population. Repeat from step 2.

SPEA2 uses two processes for creating new individuals: mutation, where a copy of a parent is made, then each parameter of the child copy has a chance to change, i.e. mutate. The second is binary crossover where two parents are chosen, and each parameter of the child is randomly taken from

one parent or the other. The specifics of the algorithm can be found in [74]. The algorithm ranks fitness based on Pareto dominance. An example is shown later in this section.

The algorithm iterates until the terminating condition is satisfied, i.e. maximum number of generations (iterations) is reached.

PISA conceptually separates the software into two components, Variator and Selector. Each program is compiled to its own binary and executed separately. Data is exchanged between the two through text files.

Program Description

The optimization platform is written in C++ in a 64-bit Linux environment.

Requirements

The TRIUMF optimization platform must have the following:

1. Allow multiple engines to be included into the same optimization problem. This allows for a global setup and discovers dynamic relationships typically not possible with local optimization.
2. Parallel capable - moving from local to global optimization necessitates higher computing resources.
3. Easily extensible to add new engines.
4. A generic input file capable of describing all optimization problems described by section B.
5. Good exception handling mechanism in preparation for the multitude of errors that can occur with the inclusion of different simulation engines.
6. Handle unit conversions between different engines.
7. Create python codebase for common post-processing operations, such as extraction of energy and emittances.

Prerequisites

The input file for the optimization program is defined in XML format. The libxml2-2.7.8.13 [6] XML parser was used. All C++ projects using libxml must include the flags `-lxml2 -lm` in the linker options.

The Boost libraries [1] are required, in particular `boost_filesystem`, `boost_spirit` and `boost_regex`.

The following environmental variables are required:

- `LD_LIBRARY_PATH` - directory containing the compiled optimization libraries.
- `PYTHONPATH` - directory containing the python post-processing code. This path is optional if the included python code is not used.

Configuration Management

The development environment `gongc.triumf.ca` uses Git [3] for version control.

All design documentation are also version controlled.

Deliverables

In build order:

1. libDiagnostics.so
2. libAccessibility.so
3. libEvaluator.so
4. spea2 (executable)
5. Variator (executable)
6. Minion (executable)
7. OptimizationMain (executable)

The executable OptimizationMain is the entry point for running optimization.

Top Level Logic

The three major components of the optimization program are (fig. B.1):

- Variator - the optimization base. Controls mutation/crossbreeding, giving information to Selector on the current generation. Passes information to Evaluator on what runs to make.
- Selector - part of the PISA framework that evaluates fitness of individuals. The SPEA2 algorithm used here is almost unchanged from the original.

- Evaluator - wrapper framework for the physics engines. Gets information from Variator on what to run, runs the engines, then sends the information back to Variator. Evaluator is completely new and does not appear in PISA or APISA.

Figure B.1: Optimization top level logic.

Variator and Selector are not coupled to each other programmatically, but rather run as separate programs that communicate by reading/writing text files. They run as state machines with the state codes passed via the text files. Evaluator is a part of Variator, and thus could be called directly.

We make a distinction between global and local problems. A global problem is the overall ERL optimization. Given a population of individuals (i.e. ERL designs), we select the best individuals in the hope to maximize/minimize some parameters. Each optimization problem is a single global problem.

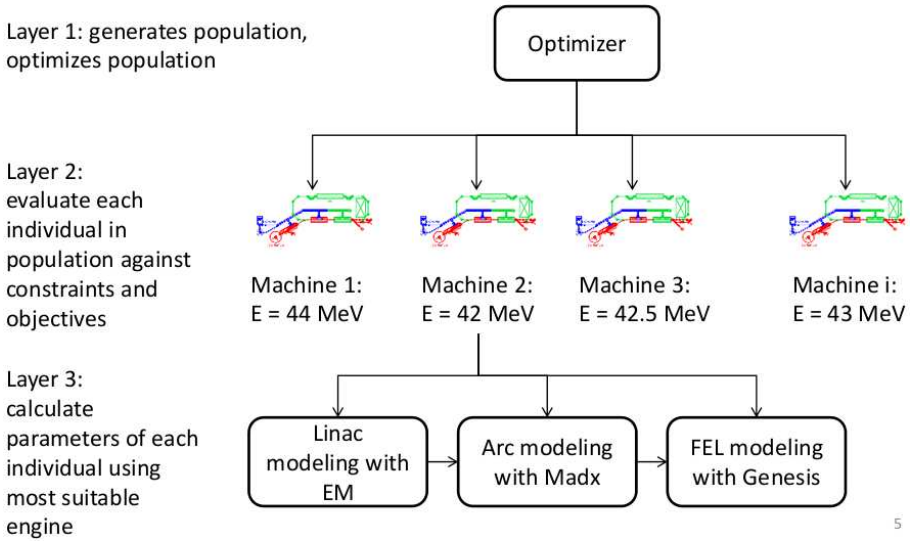

Figure B.2: Layers of the optimization problem. The top layer is the global problem, in which the optimum individual(s) are selected from a population. The lower layer is how each section of the ERL is produced.

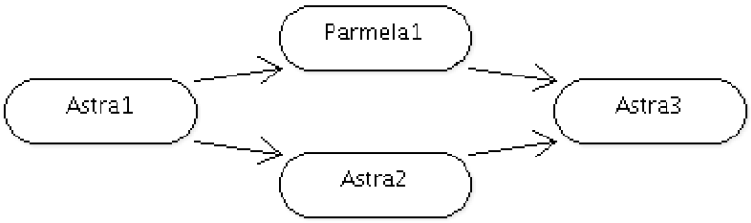

Figure B.3: Example topology. Simulation engines can be linked together in parallel or serial for flexibility.

A local problem is the production of a single individual. Figure B.2 illustrates the relationship between global and local problems. Each individual is a simulation of the beamline. We do this through multiple local problems; the output of each local problem is piped to the next local problem to form a continuous and complete simulation. It is convenient to represent the local problem as a graph (fig. B.3). Each process is a call to a simulation program, such ASTRA, PARMELA, or a custom written program. Evaluator links the output of one process to the input of the next. The graph object in fig.

B.3 is referred to as a *topology*, and each process is a *vertex*. Local problems do not always have to represent a physical section of a beamline. Example 1: GPT to GPT2ASTRA to ASTRA. Here the local problem GPT2ASTRA is a conversion program, which takes the output of GPT and massages it to a format suitable for the subsequent ASTRA run.

Selector is purely concerned with the global problem, Variator with both global and local, and Evaluator purely with local. The top level architectural design is shown in fig. B.4.

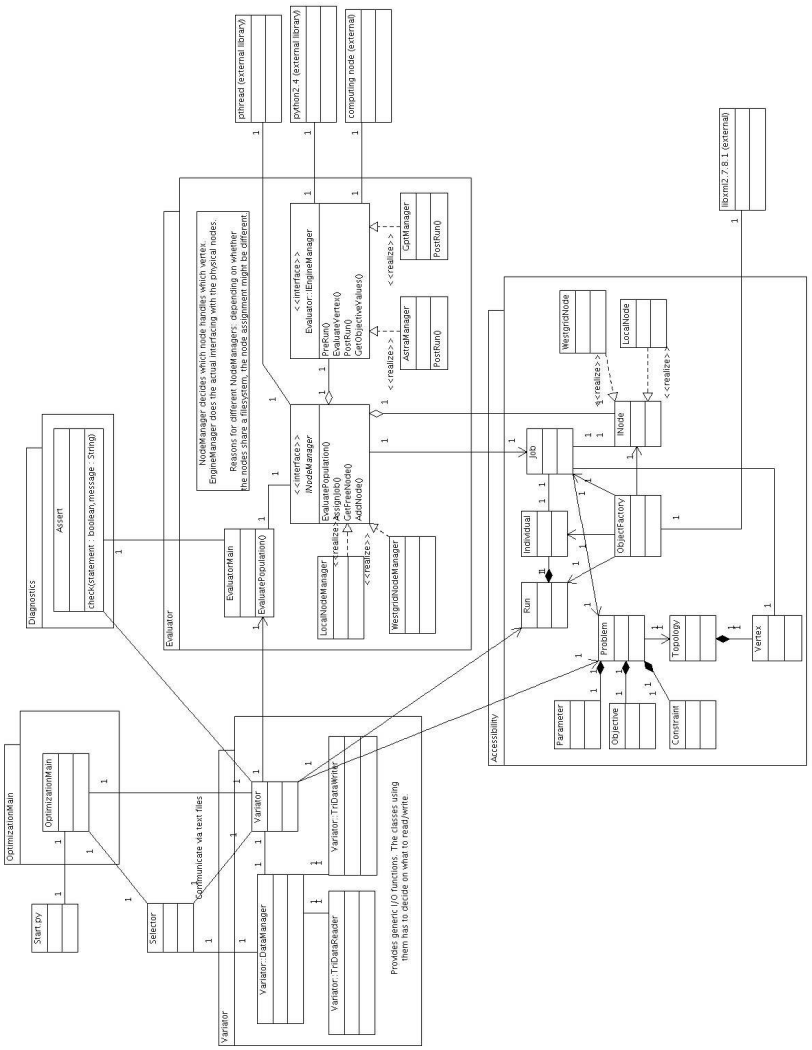

Figure B.4: Optimization platform class diagram.

Variator

Variator is written entirely in C and C++. The initial Variator codebase is Ivan Bazarov's APISA [14] framework, which in turn is based on PISA. APISA was written entirely in C, with the Variator section consisting of all global variables and global functions. Additionally, APISA works only with ASTRA and runs locally. The upgrade from APISA to the TRIUMF PISA framework consisted of the following stages:

1. In the near term, encapsulate the APISA Variator in a namespace "Variator". The code should be usable at this point, but does not fit perfectly within the object-oriented framework.

2. In the long term, modify the APISA Variator to be object-oriented. The reason is the exponential increase in the scale of the software. Moving to a multithreading, multiple engine setup significantly increases the complexity of the software, which is why the modularity of the OO-framework is desired. This is a difficult and time-consuming. Good OO-design requires that each class serves an explicit purpose. The global functions of the APISA Variator, which are split into multiple files, does not fit this requirement.

Variator acts as an interface in the creation of new generations and new individuals. It does not do the real work for either processes, but directs the flow. Variator communicates with Selector to determine fitness, and communicates with Evaluator to produce offsprings. It is a state machine (see fig. B.5).

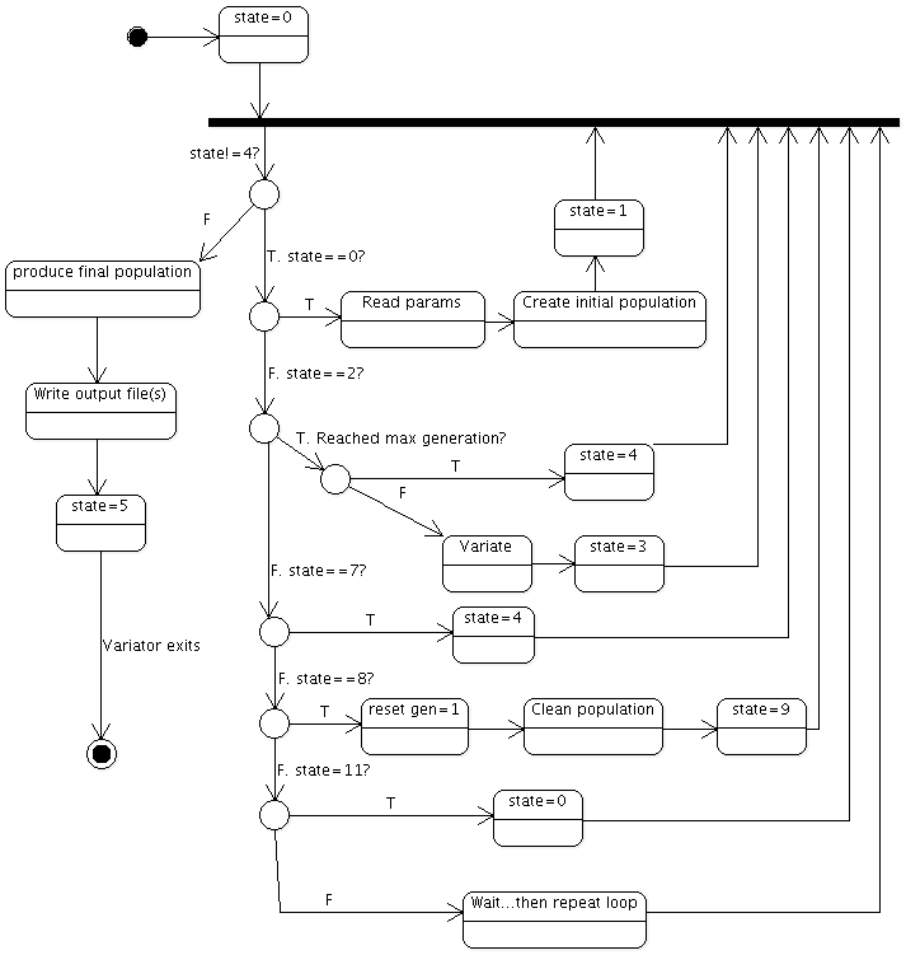

Figure B.5: Variator state chart.

Selector

Selector is the component that evaluates the fitness of each individual, and then determines the individuals of the population that are selected as parents for the next generation. The fitness evaluation algorithm is SPEA2 [74].

The Selector state chart is shown in fig. B.6. APISA's implementation of SPEA2 is used as the codebase. Refactoring was performed on the codebase. Otherwise, algorithmic changes were not done. All SPEA2 settings are moved to the optimization input XML file. This makes the XML file a unified location for all optimization input settings.

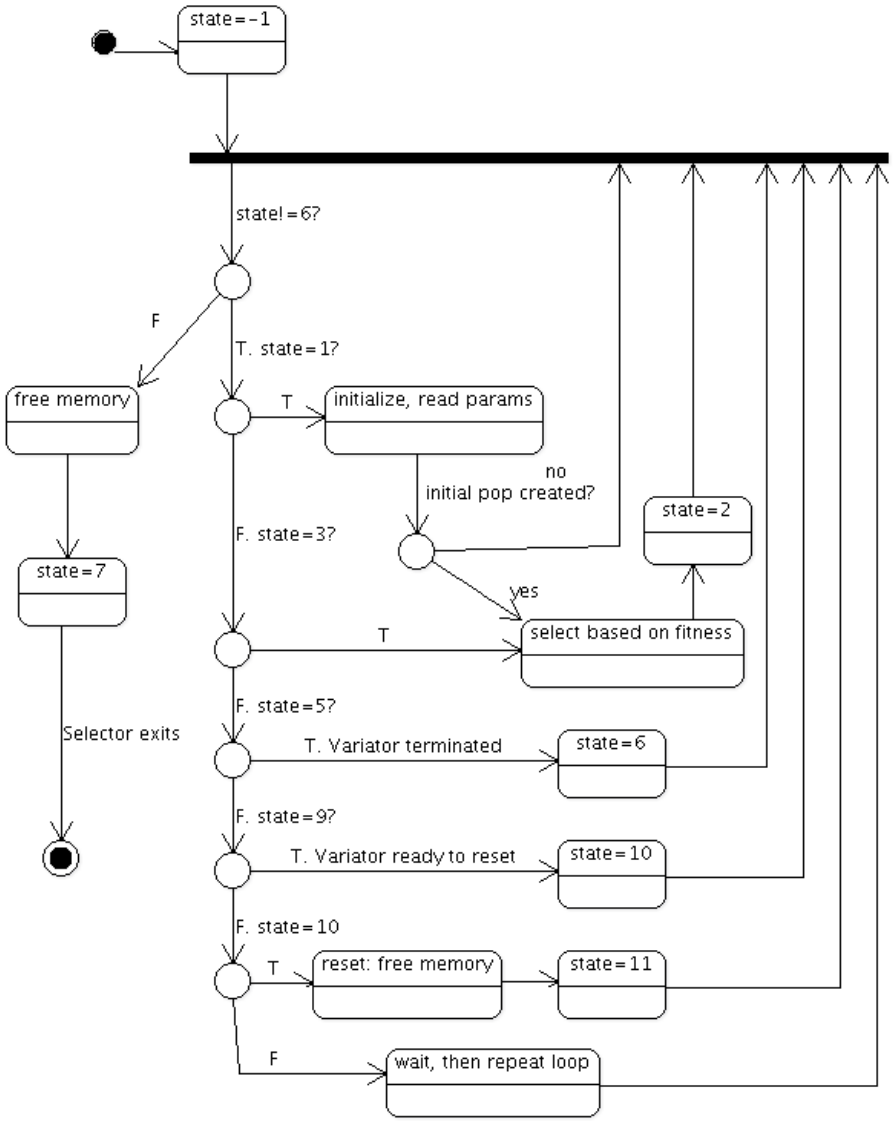

Figure B.6: Selector state chart.

Evaluator

Evaluator controls the physics engines. For each individual of the population, Variator calls Evaluator to run the engines, after which numerical values such as final emittances are produced. Each individual is obtained

via a sequence of engine processes. Evaluator does not exist in APISA, so will be written from scratch following rigorous object-oriented design principles. Evaluator also handles the multithreading using the POSIX threading library.

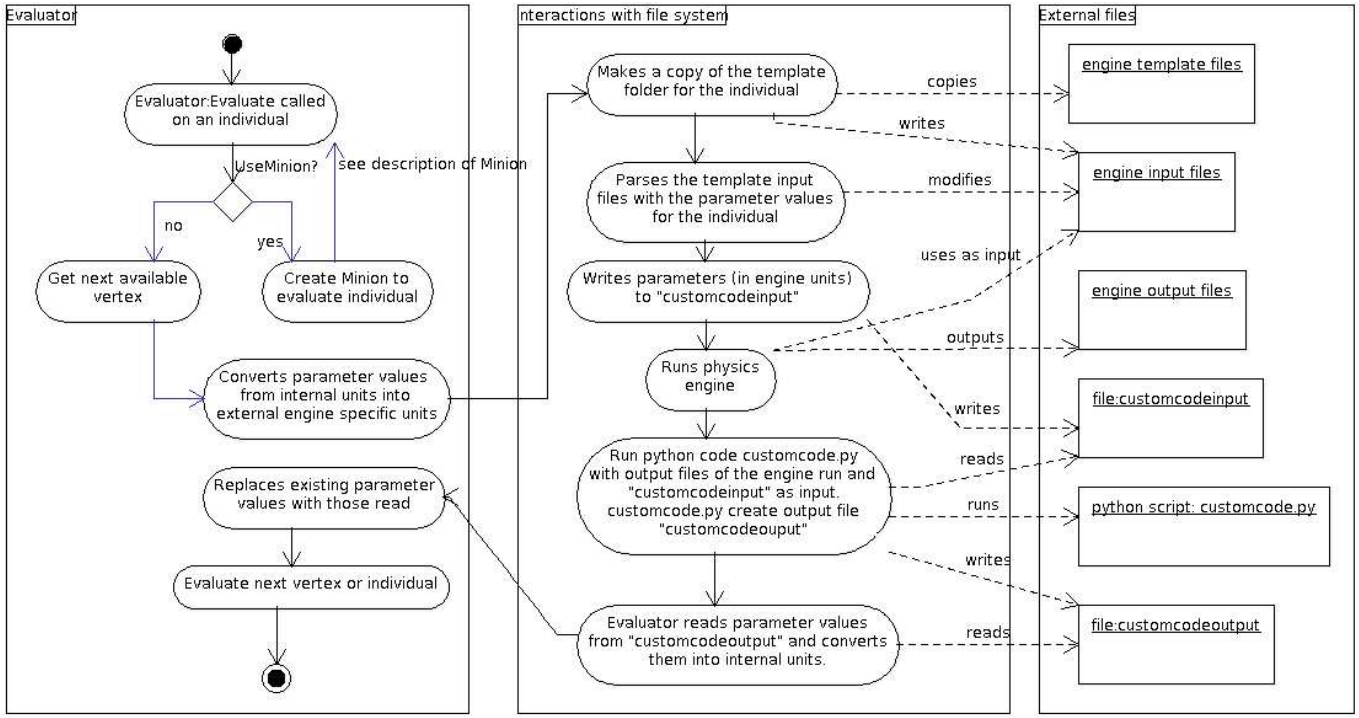

Figure B.7: Evaluator execution flowchart.

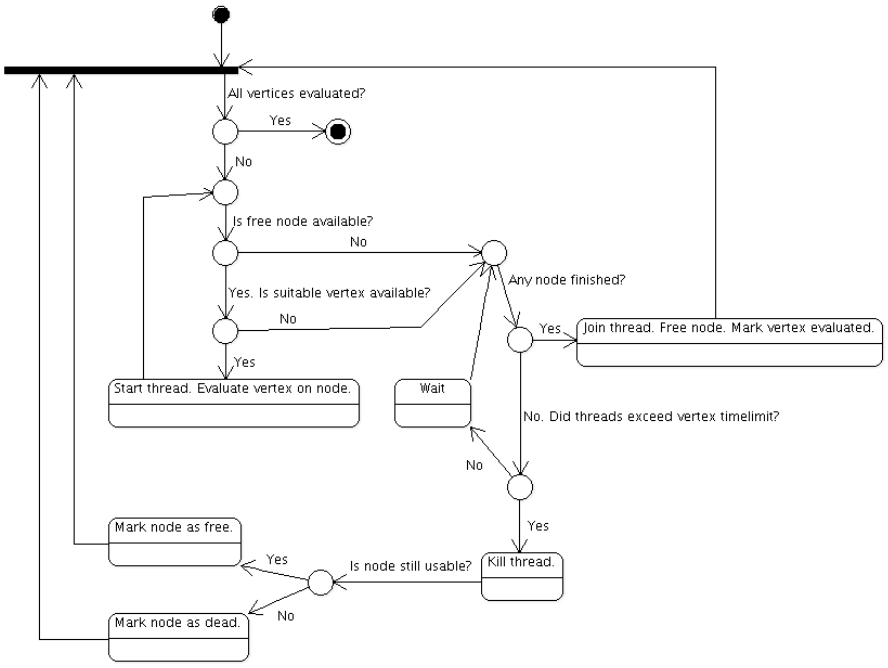

Figure B.8: Evaluator multithreading flowchart.

Fig. B.7 shows the process of evaluating each individual, and the interactions between Evaluator and the external files and physics engines. The Evaluator state diagram (fig. B.8) shows the algorithm Evaluator uses to handle parallel computing. To take advantage of parallel-processing, a thread is created for each vertex to be executed. Each thread uses `ssh` to access a node, and executes the engine instructions. The number of threads should never exceed the number of available nodes.

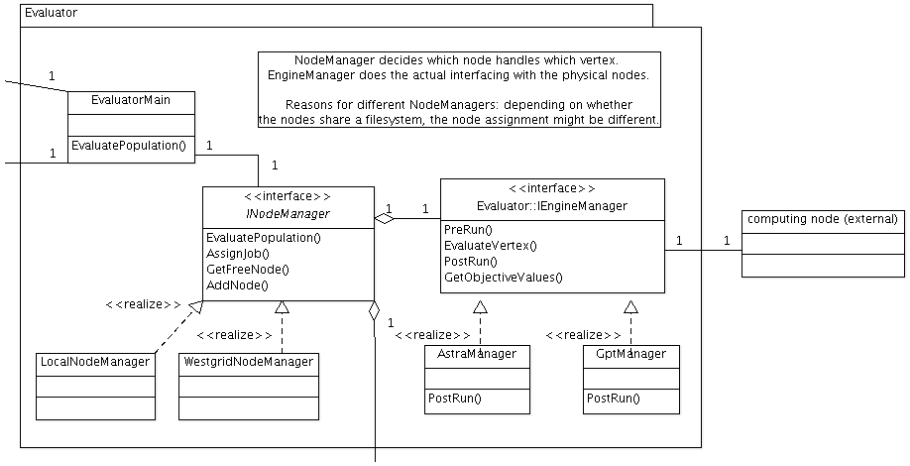

Figure B.9: Evaluator class diagrams.

Evaluator consists of two pieces. The backbone is written in C++ and acts as a mediator between the frontend and the Variator, and as a wrapper for the physics engine. The frontend allows user-injected python code, useful for post-processing. The class design of Evaluator is shown in B.9. Evaluator contains three major classes and interfaces: EvaluatorMain, INodeManager, and IEngineManager.

- EvaluatorMain - this singleton class is the interface between Evaluator and Variator. Variator passes data of individuals to this class to be evaluated. For the sake of modularity, this is the only entry point available to Evaluator.

- INodeManager - interface for classes that handles computing nodes. EvaluatorMain creates a NodeManager depending on the environment, e.g., `WestgridNodeManager` if the optimization is run on WestGrid. The NodeManager determines which vertex is assigned to which node. Reasons for different NodeManager for different environments include

  - Different environments have different ways of retrieving the list of nodes available to do work
  - Different filesystem: Westgrid nodes all share the same filesystem, whereas a cluster of individuals do not. This introduces limits on which vertex can be executed on which node. A vertex which requires output files of previous vertices must be executed on the

same filesystem as those previous vertices, to avoid copying files between systems. See below for more information.

The NodeManager also handles thread creation, monitoring, and termination.

- IEngineManager: this is the interface between Evaluator, which treats nodes and engines as abstract objects, and the actual nodes and engines which executes the vertices. The EngineManager ssh into the working node and runs the simulation engine.

Assigning Jobs to a Node

We distinguish between vertex-based assignment and individual-based assignment:

- Vertex-based - used for both local and Westgrid executions. Vertices belonging to the same individual can be executed by any different nodes. This is designed for systems where nodes share the same filesystem.
- Individual-based - designed for systems where nodes do not share the same filesystem, for instance, a network of different computers. All vertices of the same individual must be executed on the same node. Since vertices might exchange information with each other, this avoids having to copy files and data between nodes.

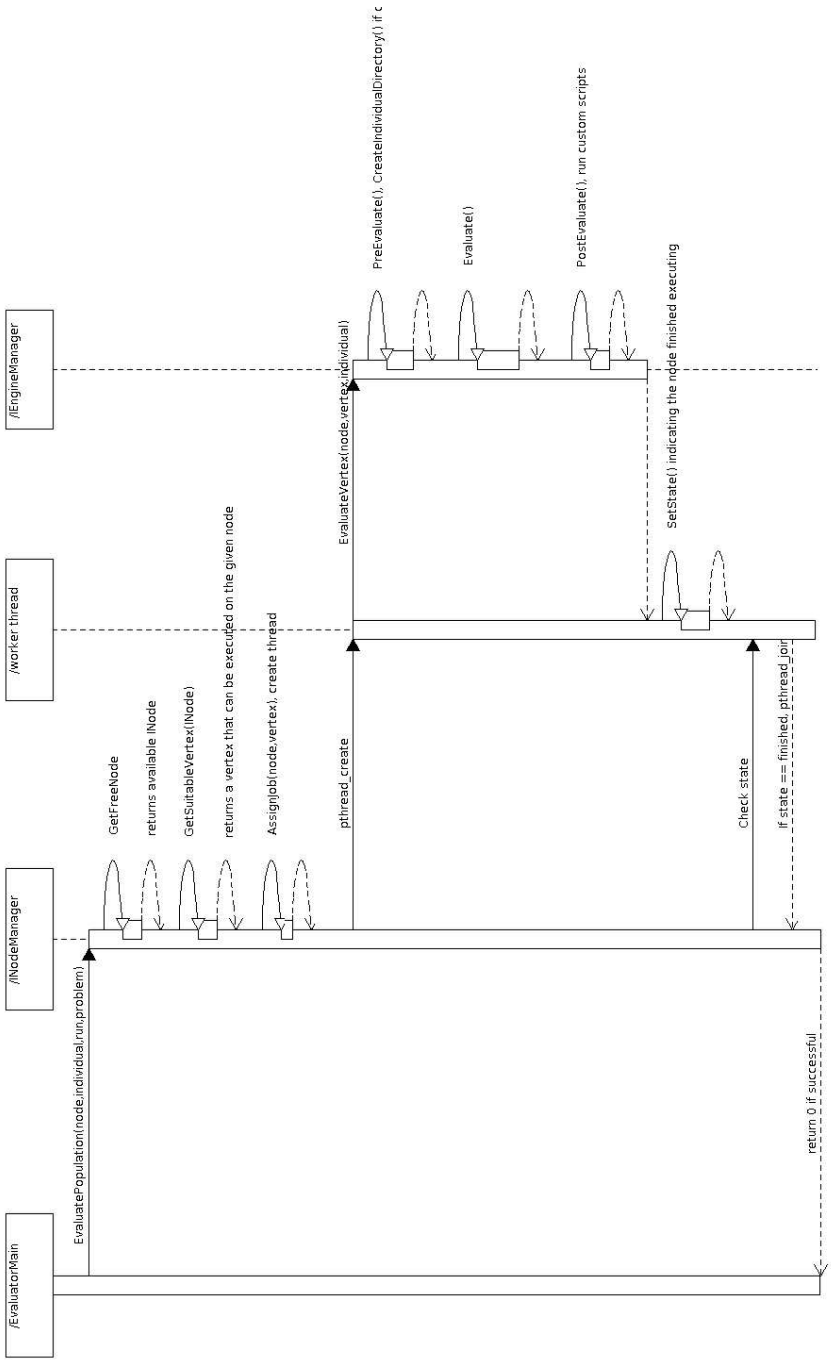

Figure B.10: Population evaluation statechart.

The optimization platform assigns each individual to a node for evaluation. A node represents a single processor, which may contain multiple cores. Multiple threads are created, with each thread evaluating a vertex of the individual's topology. The relationship between the number of threads and the number of cores in the node can be many-to-one. The evaluate procedure is shown in fig. B.10.

Process Monitoring

One of the biggest additions to the TRIUMF platform is process oversight. The increase in software scope also resulted in an increase in exception production. Process monitoring refers to exception handling in the multi-threading, parallel computing scheme. When the main node assigns a job to a worker node via `ssh`, the worker process needs to be monitored, i.e. we need to know when the worker node finishes the job, so we can analyze the results and free the worker node for additional jobs. There are two methods of process monitoring used in the optimization platform: process vs processless.

1. Process-based monitoring - requires `Environment=WESTGRID` and `ProcessLess=NO`, or `Environment=LOCAL`. Each thread creates a monitor process. The steps are shown in fig. B.11:

   (a) Create monitor on head node.

   (b) Monitor assigns job to worker via `ssh`.

   (c) Monitor does not return until job finishes.

   The advantage of such a system is the monitor takes care of tracking every aspect of the job. The disadvantage is for large optimizations, there are many monitor processes on the head node. For systems such as Westgrid, the administrators will think you are a resource hog, even though the monitors are not computationally intensive.

2. Processless job monitoring - requires `Environment=WESTGRID` and `ProcessLess=YES`. The steps are shown in fig. B.12:

   (a) Create monitor on head node.

   (b) Monitor assigns job to worker via background `ssh`, which returns the PID. Monitor returns immediately.

   (c) Thread polls PID on worker node to check if engine finished running.

This is the preferred way of running because it does not overload the head node with monitoring threads. More details can be found below B.

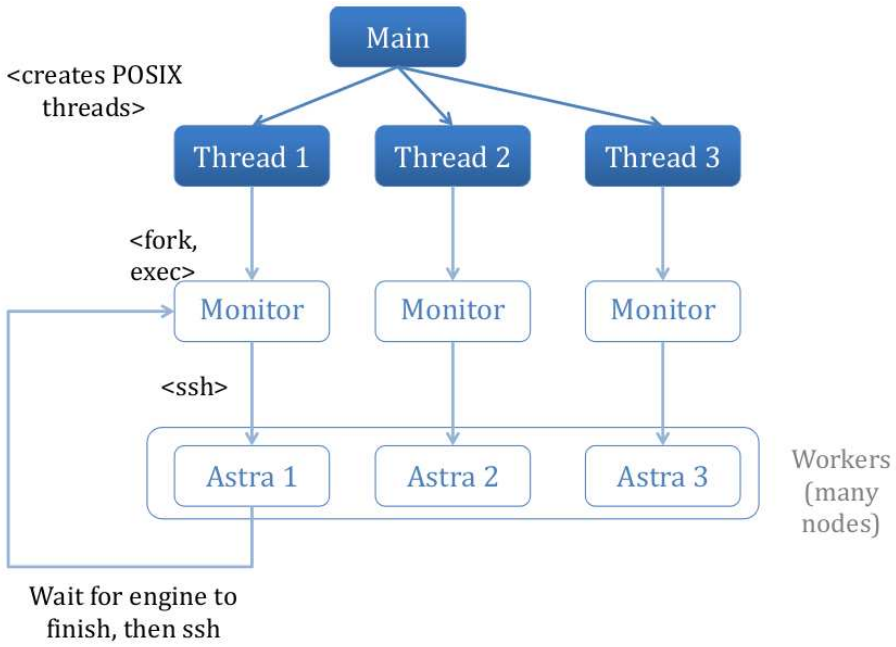

Figure B.11: Process monitoring with monitoring thread.

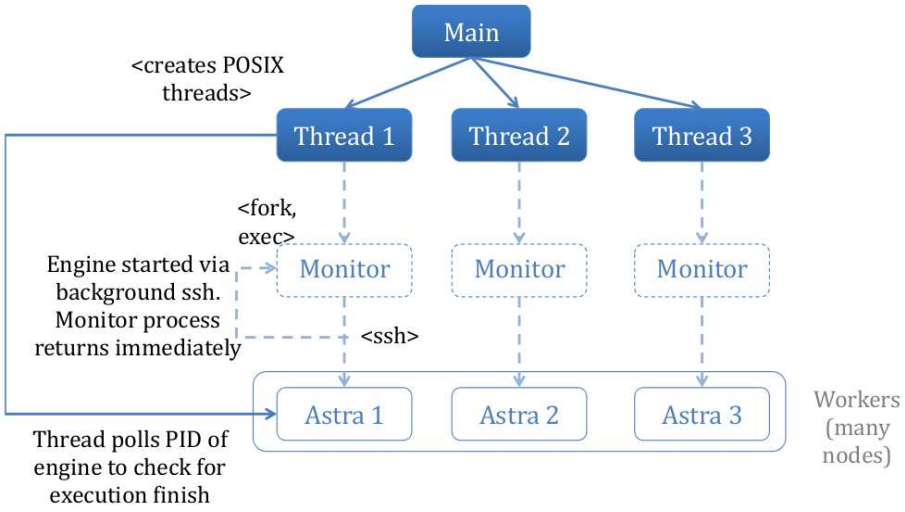

Figure B.12: Process monitoring without monitoring thread.

Minion: Local Population Evaluator

For processless monitoring 2, the head node (where Variator runs) does not submit individuals to nodes to be ran. Instead, the run is outsourced to Minion. Variator starts Minion on a given node, and passes individuals to be evaluated. Minion then runs in process monitoring mode and evaluates the individuals on its local host node. Variator and Minions exchange data through text files. The interaction flowchart is shown in fig. B.13.

The purpose of such a scheme is that the POSIX threading system, while powerful, is not designed for extreme numbers of threads. Large optimization problems often make use of dozens of worker nodes. If one thread is assigned to each node, terrible performance is caused by the large number of threads due to virtual memory limitations, context-switching overhead, and scheduler overhead. Globally, Variator circumvents this threading problem by executing Minion as a detached process. Variator then uses a single thread, the main thread, to monitor the status of all Minions. Locally, Minion is allowed to assign a small number of threads to take advantage of parallel computing on a multicore node, but not too many to cause virtual memory problems. This combination of multiprocessing and multithreading is an efficient way of executing a genetic algorithm.

To allow Minion to operate correctly, set the XML parameter `UseMinion=YES`. Minion works as a detached process, so it is necessary for the

executing environment it runs in (which is different from the environment that Variator is running in) to contain the correct variables. Set the variables `LD_LIBRARY_PATH` and `PYTHONPATH` in the `.bashrc` file.

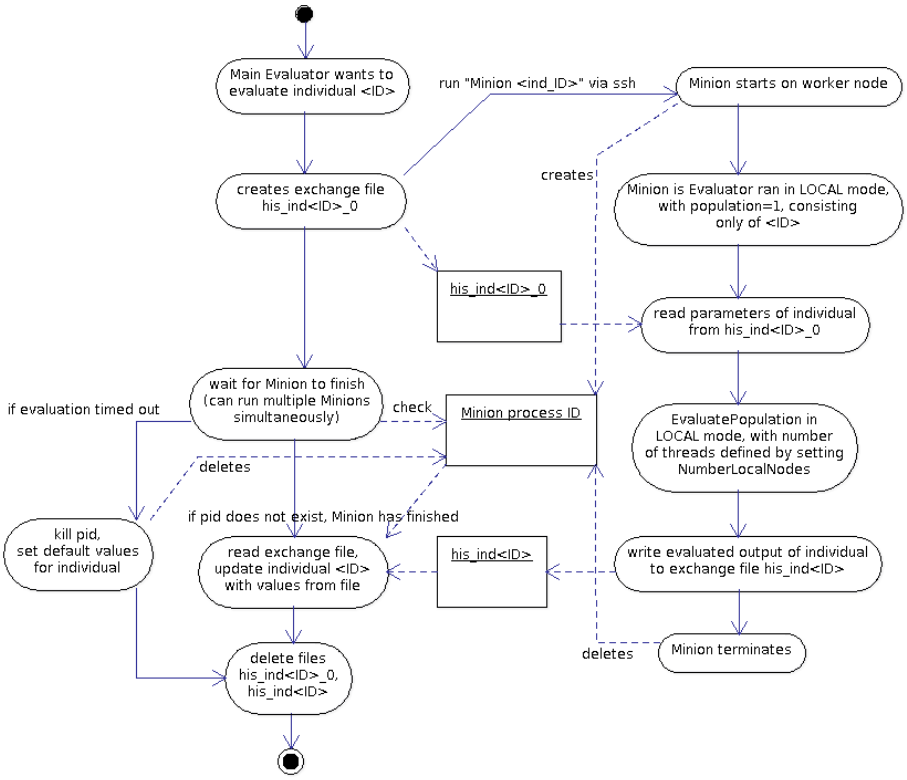

Figure B.13: Minion flowchart.

Custom Python Code

Custom code allows the user to write problem-specific code in a convenient manner. In the template directory of each local problem is a file, `customcode.py`, with the function `RunCustomCode`. A list of running variables is passed to this function, which the user can use to calculate new variables and write them to file. The user can also read from program output files to obtain values to assist in the calculation of new quantities.

Many typical functionalities for engines exist in included python files, e.g., `astra.py`, `madx.py`, to calculate and extract common accelerator physics variables such as emittances, Courant-Snyder functions, and transport map

elements. The location of these files should be stated in a global shell variable `PYTHONPATH`. By default, the path is `$INSTALLPATH/python`, where `$INSTALLPATH` is the location of the optimization engine binaries.

Input and Output Formats

The TRIUMF optimization platform uses two types of input files:

1. Optimization input (global) - input for the optimization program
2. Physics engine input (local) - each modeling engine has its own input files and format, e.g., MADX uses input files based on TRANSPORT notation

The global input format is presented first, followed by the local input format.

Global Input File Problem.xml

Global input is defined in the XML file Problem.xml. This file includes

1. Problem description - parameters, constraints, objectives, and topology
2. Settings - variables related to the execution of the optimization program, such as output folder path and seed

The XML file problem.xml includes everything needed to define the current optimization problem. The XML file must begin with the tag `<Problem>`. The following blocks must be included inside the problem tag (in order):

1. SettingsList
2. UnitsList
3. DefaultUnits
4. AutoNDParamLists
5. ParamList
6. NDParamList
7. ConstraintList
8. ObjectiveList
9. Topology

Settings Input

The `<Settings>` tag contains attributes related to the run:

```
<SettingsList>
  <Setting name="RunName" value="sample_run" />
  <Setting name="BaseDir" value="./" />
  <Setting name="ExchangeDir" value="exch" />
  <Setting name="TemplateDir" value="template" />
  <Setting name="OutputDir" value="output" />
  <Setting name="OS" value="UNIX" />
  <Setting name="Environment" value="WESTGRID" />
  ...
</SettingsList>
```

All name and value pairs are case-sensitive. The list of settings can be found in the spreadsheet accompanying this document.

Units and Default Units Input

The `<UnitsList>` tag contains the list of available units. Information on units is provided in section B.

Template for Automatic Lists Input

Often we want to retrieve the same set of variables at many points on the beamline. For example, we might want x, p_x, y, p_y at the points 'start', 'middle', and 'end' of the lattice. We could create all the variables like such

```
<NDParamList>
  <NDParam name="x_start" ... />
  <NDParam name="px_start" ... />
  <NDParam name="y_start" ... />
  <NDParam name="py_start" ... />
  <NDParam name="x_middle" ... />
  <NDParam name="px_middle" ... />
  <NDParam name="y_middle" ... />
  <NDParam name="py_middle" ... />
  <NDParam name="x_end" ... />
  <NDParam name="px_end" ... />
  <NDParam name="y_end" ... />
  <NDParam name="py_end" ... />
```

```
  ...
</NDParamList>
```

The alternative is to add a block after `UnitsList`:

```
<AutoNDParamLists>
  <AutoListDefinition name="beam">
    <Item name="x" unit="m" />
    <Item name="px unit="MeV/c" />
    <Item name="y" unit="m" />
    <Item name="py" unit="MeV/c" />
  </AutoListDefinition>
</AutoNDParamLists>
```

Later in the `NDParamList` section:

```
<NDParamList>
  <NDParam ... />
  <AutoList listname="beam" label="_start" />
  <AutoList listname="beam" label="_middle" />
  <AutoList listname="beam" label="_end" />
  <NDParam ... />
</NDParamList>
```

Optimization will automatically expand the `AutoList` block to create all the variables necessary. Auto lists are used only for non-decision parameters. For decision parameters, the upper and lower values must be specified for each, therefore they cannot be automated.

Decision Parameters Input

Decision parameters x (see eqn. 1.1) are defined under the `<ParamList>` tag. These are the free parameters that the platform will try to optimize. Boundaries and units can be specified for each parameter.

```
<ParamList>
  <Param name="Energy" minvalue="40" maxvalue="50"
    unit="MeV" />
  <Param name="Q1" minvalue="0" maxvalue="2"
    unit="T" />
  <Param name="B1Length" minvalue="5" maxvalue="10"
    unit="cm" />
</ParamList>
```

The value of the `unit` attribute must be predefined in the `<Units>` block.

Non-Decision Parameters Input

Non-decision parameters $y(x)$ are defined under the `<NDParamList>` tag. These are the implicit parameters that are not free parameters. They can be (but not have to be) used as constraints and objectives.

```
<NDParamList>
  <NDParam name="energyloss" unit="MeV" />
  <NDParam name="K" unit="dimensionless" />
</NDParamList>
```

In the above example, energy loss is typically not a free parameter predetermined at the start of beamline design, but rather a result of lattice settings.

Constraints Input

Constraints are $F_j(x)$ defined under equation 1.3. They are represented by the `<ConstraintList>` tag. Example:

```
<ConstraintList>
  <Constraint param="energyloss"
    direction="LT" value="1.2" unit="MeV" />
</ConstraintList>
```

The `direction` attribute can be less than (`LT`) or greater than (`GT`).

Objectives Input

Objectives are $g_i(x)$ defined under equation 1.2. They are implemented by the `<Objective>` block as follows:

```
<ObjectiveList>
  <Objective param="emitx_end" direction="minimize" />
</ObjectiveList>
```

The direction attribute can be `minimize`, `maximize`, or `equals`. If the direction is `equals`, the `Objective` element must also contain an additional `value` attribute, for example:

```
  <Objective param="x"
    direction="equals" value="0.5*{mm}" />
```

The above is equivalent to

$$\min \ |x - 0.5 \text{ mm}| \tag{B.1}$$

representing that we want to optimize a parameter such that it is as close to a given value as possible. The `value` attribute is not used and can be omitted if the direction is `maximize` or `minimize`.

Topology Input

The execution order of the engines, or local problems, is defined under the `<Topology>` tag.

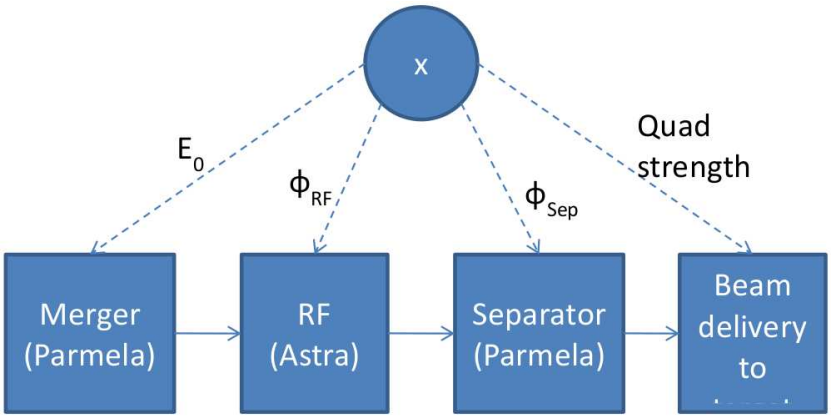

Figure B.14: Optimization topology XML block example.

A topology is a directed graph. Fig. B.14 illustrates a sample topology. The round vertex x, represents the input parameters. The square vertices represent local problems. To produce an individual, Variator selects starting values for the global optimization parameters x. The beam runs through the local problems in the defined order. The merger section is modeled first, using Parmela. Then the beam passes through the RF, and so on. Any local problem can make use of x. For example, we can choose to optimize the RF phase in the cavities, hence the ASTRA input file for the RF local problem can include the parameter phi_{RF}. The illustrated topology is defined in XML as

```
<Topology>
  <Vertex name="Merger1" type="PARMELA"
```

```
    inputfolder="m1" prereqs="" timeout="60" />
  <Vertex name="RF1" type="ASTRA"
    inputfolder="rf1" prereqs="Merger1" timeout="120" />
  <Vertex name="S1" type="PARMELA"
    inputfolder="s1" prereqs="RF1" timeout="600" />
  <Vertex name="BD1" type="ASTRA"
    inputfolder="bd1" prereqs="RF1,S1" timeout="60" />
</Topology>
```

Each Vertex node must have a `type` attribute. This can be ASTRA, PARMELA, etc. The Process must also have an `inputfolder` attribute, which states the relative location of the local input files. For instance, the location of an ASTRA Process A1 must contain the ASTRA input file `astra.in`.

The `prereqs` attribute is a comma-delimited list of vertices that must complete before the current vertex can execute. In the above sample, RF1 will not start until Merger1 finishes. Merger1 and S1 can run in parallel, since they are independent from each other. NOTE: if vertex A2 uses A1 as a prerequisite, A1 must be listed before A2 in the XML.

The `timeout` attribute denote the maximum wall-time (in seconds) the vertex is allowed to run for. The engine run is cancelled when this limit is reached, and the vertex must be processed again for the particular individual. The timeout is a safety feature to prevent a software hang-up from destroying the entire optimization run.

Often, one engine is used to produce a file that will be used as an input to another engine. For instance, GPT can be used to produce a particle distribution file, which can then be used as the input file for ASTRA. Copying files can be performed as follows:

```
  <Vertex name="v1" type="GPT" inputfolder="gpt1"
      prereqs="" timeout="600" />
  <Vertex name="v2" type="ASTRA" inputfolder="astra1"
      prereqs="v1" timeout="120">
    <FileRequest from="v1"
name="out/dist" newname="in/dist" />
  </Vertex>
```

In the above code, when the optimization engine begins to execute the engine v2, it will first copy the file `out/dist` from the directory of the engine v1 into the directory of the engine v2, and renames the file to `in/dist`. To make sure the file exists, it is a good habit to put vertex v1 as a pre-requisite for the vertex v2.

Local Engine Input Files

The vertex input files, or local input files, are files necessary for an engine to run. They are organized into template directories. An example engine might have an input file that takes in energy as a parameter. The vertex template directory could therefore contain the input file with energy as a parameter:

```
Run beam at ${energy} MeV.
```

When this vertex is evaluated for a particular individual, a copy of the template is made and parsed to contain:

```
Run beam at 10.2 MeV.
```

where 10.2 is the optimization generated value of the energy for the particular individual.

Another example is an RF section of the linac modeled by ASTRA. We must provide the optimization program the path to the ASTRA input file. In the topology definition in Problem.xml, we specified an `inputfolder` attribute for each Process, i.e. local problem. This input folder contains template files. As an example, the RF model has one template file, `astra.in`. Within the template file `astra.in`, we can find the line

```
MaxE(1)=${MaxE1}
```

where `MaxE1` is the name of a parameter (within x or $y(x)$). When Variator requests a new individual to be created, Evaluator makes a directory for the individual and copies the ASTRA template file to the directory. If for a particular individual, a value of 10 MeV for `MaxE1` is selected. Variator makes the substitution in the copied file

```
MaxE(1)=10
```

The copied Astra input file can now be read by the program ASTRA. Evaluator then starts ASTRA for the current local problem with the newly generated input file.

Local Engine Output

The output of a local problem adheres to the conventions of the physics engine used in the problem. ASTRA problems create output files according to the convention described in the ASTRA manual. In order to read data from a local output, the global framework must first check the type of process

of the local problem (whether ASTRA or PARMELA, etc.), then follow the folder and file convention of that process type.

When the process of a local problem finishes running, we want to extract useful information from the output data. Each modeling engine implements `IEngineManager` and a post-run function. Within this function, we can extract commonly used information from the output files. As an example, we have ASTRA designed to model the RF section. During post-run, we can extract the energy of the beam and the size of the beam. These extracted values belong to the set of non-decision parameters $y(x)$, and are passed back to Variator to be used, for instance, to check whether a constraint on beam size is satisfied.

Custom Code for Local Problems

Custom code allows the user to manipulate the optimization parameters to suit their programming needs. This is convenient, for instance, if for an Astra vertex, the user wants to extract the emittance only of particles that are in front of the beam center.

Each vertex contains a file `customcode.py` (see the folder structure section). `IEngineManager` executes the following algorithm:

```
EvaluateVertex
  Run vertex engine
  Writes parameter list to customcode.in
  Run customcode.py
    customcode.py reads from customcode.in
    Manipulates/extracts parameter values as needed
    Outputs modified values to customcode.out
  Reads customcode.out
  Updates parameter values for this individual
End
```

The python code is executed by `system()`. Embedded Python was ruled out as an option since the Global Interpreter Lock (GIL) allows only one Python object to be accessed at a time. This effectively makes the optimization program single-threading. Custom codes are anticipated to have much I/O interaction with the file system, so can be very computationally costly. Using System makes this process multi-processing, so bypassing the single-threading issue.

Global Optimization Output

The optimization platform writes the population to a history file `his`. Format of the file is

```
Ln 1:   # comment
Ln 2:   # comment
Ln 3:   id1 x0 x1 ... y0 y1 ... g0 g1 ... F0 F1 ...
Ln 4:   id2 x0 x1 ... y0 y1 ... g0 g1 ... F0 F1 ...
...
Ln N+2: idN x0 x1 ... y0 y1 ... g0 g1 ... F0 F1 ...
```

The first two lines are comment lines. The following lines list the individual ID, x, y, F, and g values for each individual of the population. N denotes the size of the population.

With each generation of the genetic algorithm, the history file is overwritten with the new generation. An XML setting `HistoryBackupInterval` can be used to periodically save the history file.

Filesystem Structure

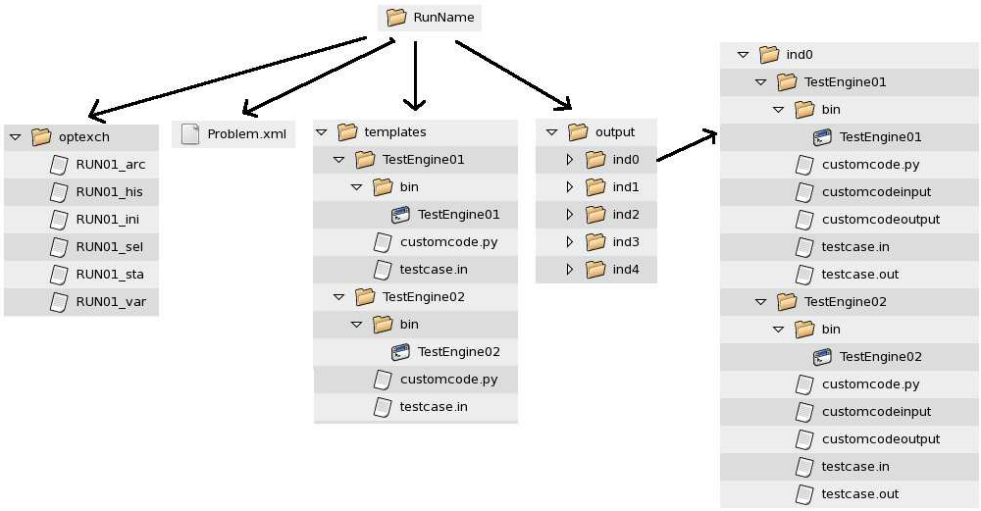

Figure B.15: Optimization filesystem structure.

Fig. B.15 illustrates the folder structure of an optimization problem:

1. Problem directory - base directory (`RunName` in figure) for the current optimization run. The global XML input file is located inside. This directory is unique to each optimization problem.
   (a) Global input file `Problem.xml`.
   (b) `optexch` is the location for exchange files from Variator and Selector, and global output files. The name is carried over from APISA.
   (c) The `templates` directory contains the engine files that define the problem topology. In the above sample, there are two engines: TestEngine01 and TestEngine02. Within the folder of each engine are:
      i. Engine executable.
      ii. Template input files for the engine. Here it is testcase.in.
      iii. A `customcode.py` file, which contains user-written python code to manipulate program output.
   (d) The `output` directory stores run files for each individual. Results for each individual are stored its own directory, named after the individual ID. The contents of each individual directory mimic that of the template directory, but with parameter values specific to that individual. It is best to make sure the output directory is empty at the beginning of an optimization run. Some engines can check for the existence of an output file as a completion condition. A pre-existing output file can cause Evaluator to terminate a thread prematurely.
2. Executables directory (not shown in figure) containing the compiled C++ binaries (see list B). By default `/home/<user>/opt`. This path needs to be added to the environmental variable `LD_LIBRARY_PATH`.
3. Directory (not shown in figure) containing python files for common engine operations. For example, the file `madx.py` contains functions for manipulating MADX output. By default `/home/<user>/opt/python`. This path needs to be added to the environmental variable `PYTHONPATH`.

Multithreading and Parallel Processing Implementation Details

To take advantage of high performance computation clusters such as WestGrid [8], the optimization program uses both multithreading and parallel

processing in its algorithms. The current implementation of the platform requires that the cluster supports parallel computing, not distributed computing, i.e. all processors share the same storage space. The interaction between threads and processes are detailed in fig. B.16.

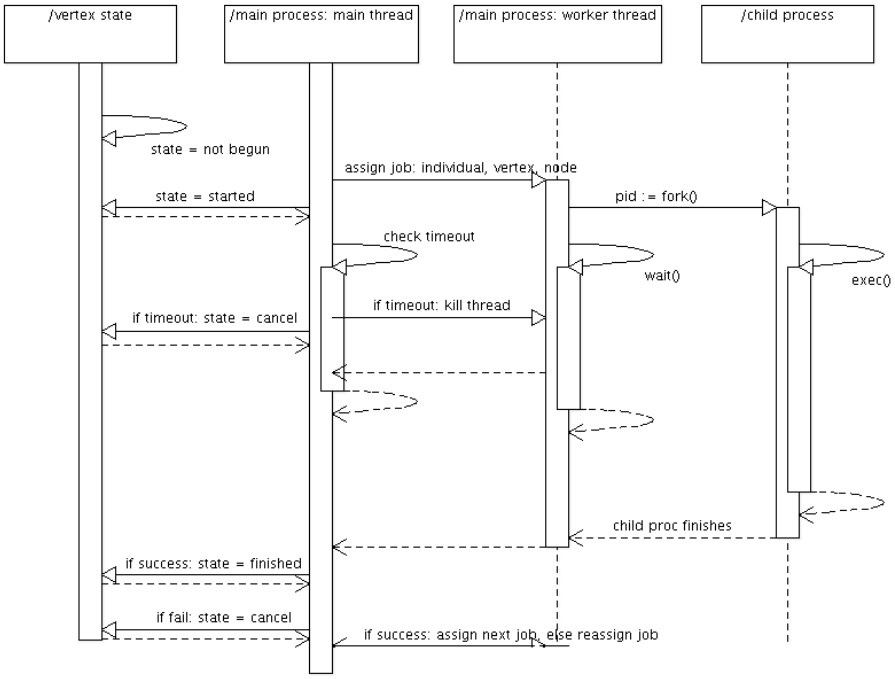

Figure B.16: Optimization multithreading sequence diagram.

The above components involve Variator and Evaluator. Selector is single-threaded. The major components of the scheme in fig. B.16 are

1. Main thread - the main Variator thread. Handles all job assignments. When a worker thread finishes, the thread uses the node occupied by the previous thread, to assign a new job.

2. Worker threads - when the main thread assigns a job, it creates a worker thread to execute this job. Multiple worker threads can run concurrently. On Westgrid, the maximum number of worker threads depends on the number of nodes granted by the Westgrid scheduler. For local runs, the number of nodes can be changed in `Problem.xml`. When a worker thread is created, it calls `fork()` to create two pro-

cesses. The worker thread (parent process) waits for the child to complete, then calls post-run functions (custom code).

3. Child process - the child process is created by the worker thread through `fork()`. The child process calls `exec()` to run the physics engine, and terminates right after. The child process may be terminated prematurely by the main thread if a timeout clause is violated.

4. Vertex state - an internal representation that stores the current status of the job evaluation.

Executing an Engine

A modeling engine such as ASTRA is executed when a child thread is assigned a vertex to evaluate. The thread creates a child process using `fork()`. The child process uses `execvp()` to start the physics engine. For a local run, the command looks like

```
astra input > output
```

Internally, Evaluator tokenizes the command before it is passed to `execvp`:

```
argv = {"astra", "input", ">", "output"}
execvp("astra", argv)
```

For a WestGrid run, since we want to tunnel into another processor, the command is of the form

```
ssh -q nodename cd vertexdir; astra input > output
```

This is converted into

```
argv = {"ssh", "nodename", "cd", ...}
execvp("ssh", argv)
```

The command string is tokenized via `strtok_r()`, the thread-safe version of `strtok`.

Evaluation Timeout

Since the optimization platform allows for a large array of different modeling engines, many exceptions can occur. This can arise from input files with bad combinations of values or bugs with the engine code, and can cause

the engine to hang-up. The platform uses a timeout counter to check for hang-ups.

The main thread keeps a list of process IDs for physics engine processes. When a worker thread creates a physics engine process via `exec()`, the resulting `pid` is added to this list. The job assignment algorithm in the main thread will periodically check whether any process violates the timeout condition. If so, the main thread sends a `SIGKILL` signal to terminate the process. The worker thread which is waiting for the physics engine to finish will also join. The purpose of the timeout is to reap any process that might have encountered an error during execution and hangs as a result.

A job which timed out is given the internal status `CANCELLED`. If the cancellation occurs during the first execution of the job, the job will be re-assigned. If the cancellation occurs during the re-execution, the job is cancelled permanently. The associated individual is given default values for all its non-decision parameters.

The timeout value (in seconds) of a vertex is defined in Problem.xml. It is up to the user to determine an upper bound for each vertex. If the timeout is too short, then it is possible that the physics engine is running properly but is not given the time to finish executing.

WestGrid

A major component of the optimization program is to take advantage of parallel processing on the WestGrid cluster.

Variator controls the distribution of nodes. When Variator assigns a job to Evaluator, it passes a node ID for Evaluator to use. Evaluator then runs the job on this node. Evaluator is in charge of tracking the status of this one node only. If the node dies or fails to respond, Evaluator must return an error code to Variator, and the job is run again.

Python on WestGrid

The python codebase uses Numpy [5]. To take advantage of Numpy in custom code, please adhere to specific policies regarding Numpy for each WestGrid cluster. On the WestGrid system Orcinus, for instance, the shell command `module load python2.4/extra` must be run (command can be added to .bashrc) first. Otherwise, the python line `import numpy` returns an error. The cluster-specific policies can be found at `http://www.westgrid.ca/support`.

A Note on Vertices in Parallel and Updating Individual Values

Consider for instance, two non-decision parameters A1, A2, and two vertices, V1, V2. V1 is charged with updating A1 and V2 for A2. V1 returns values A1=1, A2=NaN. The NaN is because V1 does not know about A2. When the engine finishes running, Evaluator will update the individual with the new values. The individual now has A1=1, A2=NaN. Then V2 finishes with A1=NaN, A2=1. Evaluator updates the individual to the new values. Instead of A1=1, A2=1, the individual now has A1=NaN, A2=1. The correct value for A1 returned by V1 is overwritten by the NaN from V2. This would occur if the update mechanism is indiscriminant.

To prevent good values from being overwritten by bad ones, we have to distinguish which is good. As a convention, an Individual object should start with random values for decision variables, and INFINITY for non-decision variables. The INFINITY value is used as a check when update parameter values. If an engine returns INFINITY for a particular parameter, it means that parameter should updated by a different engine. Therefore, the starting values of the Individual object are integral to the operation.

This means that any customcode function which returns INFINITY should be truncated to some finite value (default 10^{30}), so as to not confuse between a legitimate INF vs a do-not-overwrite INF.

Units

Since the optimization framework is designed to optimize over multiple (perhaps different) physics engines, it has to adapt to the unit convention of each engine when necessary. For example, Empirical Model uses MeV whereas MADX uses GeV. The optimization platform takes care of the frustration of unit conversions between engines.

Parameters and constraints are defined with a unit attribute in the XML input file. Functions related to units and conversion are stored in the optimization platform's Accessibility library. Accessibility keeps a set of units, each with a defined numerical value. For example, the user can define length units "m"=1 and "mm"=0.001. Then if a parameter is defined as

```
<Parameter name="x" min="2" max="5" unit="mm" />
```

In internal units, the above parameter can vary from 0.002 to 0.005.

When parsing template files, the parameter values must be converted into the external units that the engine uses. The template parsing func-

tion in Evaluator must remember to call the necessary conversion functions. Each `IEngineManager` class contains its own set of default units relevant to the particular engine. These default engine units are stored as settings in `Problem.xml`.

Unit conversions occur in the following areas:

- Creating parameters and constraints from the XML file. Need to convert values defined in XML into internal units.
- Parsing template files. Convert parameter values from internal into external engine units.
- Writing `customcode.in` file. Convert into external units.
- Reading from `customcode.out` file. Convert from external engine units into internal units.

The local problem (including `customcode.py`) assumes all units are in the local engine units. The global problem (including the global output file) assumes global units.

Unit Definitions

The list of available units are defined in the XML file under the `<UnitsList>` block. For instance:

```
<UnitsList>
  <UnitType name="length" />
    <Unit name="m" value="1" />
    <Unit name="cm" value="{m}/1e2" />
    <Unit name="mm" value="{cm}/10" />
  </UnitType>
</UnitsList>
```

All the units and types used in the Parameters and Constraints XML blocks must be defined here first. The value attribute for a unit can reference other units, as long as the referenced units are defined first. In the above, if the cm unit is defined first, a parsing error would occur. The unit values are translated from XML strings into floating points by 1) using Boost Regex to parse any references to other units, then 2) using Boost Spirit6 to calculate the value. The units calculator is based on the example VariantCalc [7].

Default Units

The optimization engine has a list of default units (labeled "generic"). In addition, each engine can have its set of default units. If, for example, the XML does not define a default length unit for ASTRA, then the generic length will be used. All custom code inputs are in the units of the engine that created them. When reading from custom code outputs are also assumed to be in the engine default units.

Defaults units are defined in XML as:

```
<DefaultUnits>
  <DefUnit engine="GENERIC" type="length"
    defaultunit="m" />
  <DefUnit engine="GENERIC" type="angle"
    defaultunit="deg" />
  <DefUnit engine="ASTRA" type="length"
    defaultunit="cm" />
</DefaultUnits>
```

Every unit type must have a unit defined under the label GENERIC. This is the unit that the optimization program uses if an engine specific unit cannot be found.

When defining parameters in XML, make sure to set numerically stable values for units. Upper and lower limits should not be smaller than 10^{-7}. E.g., instead of setting parameter x to be between 10^{-7} and 10^{-5} m, define a new unit "um"=m/1e6, and set parameter x to be between 0.1 and 10 µm. This avoids any rounding errors that can occur during computation.

Appendix C

List of Parameters and Constants in the ERL Optimization

Here we detail free parameters and their search ranges, beam parameters, and constants, as implemented in the ERL optimization setup. The objectives and constraints in the optimization are listed in section 4.

Initial Beam Parameters

Initial ERL beam parameters are constants taken from E-linac baseline designs [25, 30], and are listed in table C.1. RIB parameters are taken from baseline designs [26] and injector ASTRA simulations [21–23] and are listed in table C.4.

Table C.1: Initial ERL beam parameters for optimization.

Parameter	Value
Bunch charge Q	100 pC
Horizontal normalized emittance $\varepsilon_{x,n}$	10 µm
Vertical normalized emittance $\varepsilon_{y,n}$	10 µm
Longitudinal normalized emittance $\varepsilon_{z,n}$	47 µm
β_x	4.787 m
α_x	-1.64881
β_y	1.08843 m
α_y	0.91827
Bunch length σ_z	0.0003 m
Energy spread δ	0.019
Electron energy E_{in}	7.5 MeV
η_x	0 m
$\eta_{x'}$	0

Table C.2: Initial rare isotope beam parameters for optimization.

Parameter	**Value**
Bunch charge Q	16 pC
Horizontal geometric emittance ε_x	0.3257 µm
Vertical geometric emittance ε_y	0.3257 µm
Longitudinal geometric emittance ε_z	0.331 µm
β_x	4.8 m
α_x	-1.6
β_y	1.1 m
α_y	0.92
Bunch length σ_z	0.00026 m
Energy spread δ	0.04
Electron energy E_{in}	10 MeV
η_x	0 m
$\eta_{x'}$	0

RF Parameters

Cavity amplitude and phase were optimization variables, with all four cavities operating independently from each other. The initial phases of the RIB and ERL beams can be different to represent the independent phasing of the RIB and ERL injectors. The list of RF parameters is shown in table C.3. The parameter ranges are the maximum ranges as defined by the Empirical Model interpolation tables. The RF phases span 20° to either side of the crest.

Table C.3: RF optimization parameters.

Parameter	**Value**
Frequency	1.3 GHz
Cavity 1 amplitude	[16,20] MV/m
Cavity 2 amplitude	[16,20] MV/m

Continued on next page

Table C.3 – continued from previous page

Parameter	Value
Cavity 3 amplitude	[16,20] MV/m
Cavity 4 amplitude	[16,20] MV/m
Cavity 1 phase	[310°,350°]
Cavity 2 phase	[310°,350°]
Cavity 3 phase	[310°,350°]
Cavity 4 phase	[310°,350°]

Lattice Parameters

Lattice optics are mostly free parameters.

Drifts in the merger, linac, and separator are not variables due to existing design constraints. Certain quads in the arcs are given higher ranges than others due to the excessive demands placed on them. For example, sections with one quad between two dipoles are given extra range to compensate for possible strong dipole effects (see fig. 4.1). Other quads, while capable of reaching the same ranges [11], are not given the freedom because we seek solutions with low to moderate optical demands. Note that the first and last quads of both arcs have a high positive limit. This is to offset the strong vertical focusing of dipoles to either side of the quads. The dump doublet has higher K_1 (see Appendix A for definition) because the beam energy is lower in this section. In addition, all quads in the recirculation loop are allowed to move, i.e. the drifts around the quads can change. Drifts are minimum 25 cm to reserve space for diagnostics and other equipment.

Table C.4: Lattice optics parameters, listed from upstream to downstream. Drifts are labeled 'L'. Arc quads are labeled with the system AxyQz, where x is the arc number, y is the arc drift section (1, 2, or 3), and z is the quad number, so A21Q3 is the third quad in the first drift of arc 2.

Section	Parameter	Value
EABT	EABTQ1,K_1	[-15,15] m^{-2}
EABT	EABTQ2,K_1	[-15,15] m^{-2}
EABT	EABTQ3,K_1	[-15,15] m^{-2}
EHAT	EHATQ1,K_1	[-15,15] m^{-2}
Continued on next page		

Table C.4 – continued from previous page

Section	Parameter	Value
EHAT	EHATQ2,K_1	[-15,15] m^{-2}
EHAT (RIB only)	EHATQ3,K_1	[-15,15] m^{-2}
EHAT (RIB only)	EHATQ4,K_1	[-15,15] m^{-2}
Arc 1 matching	A1MQ1,K_1	[-15,30] m^{-2}
Arc 1 matching	drift L1	0.25 m
Arc 1 matching	drift L2	0.25 m
Arc 1	A11Q1,K_1	[-15,30] m^{-2}
Arc 1	A12Q1,K_1	[-15,15] m^{-2}
Arc 1	A12Q2,K_1	[-15,15] m^{-2}
Arc 1	A12Q3,K_1	same as A12Q1
Arc 1	A13Q1,K_1	same as A11Q1
Arc 1	drift L1	[0.25,0.45] m
Arc 1	drift L2	[0.25,0.38] m
Mirror separator	MSEPQ1,K_1	same as A1MQ1
Chicane matching	CHIQ1,K_1	[-15,15] m^{-2}
Chicane matching	CHIQ2,K_1	[-15,15] m^{-2}
Chicane matching	drift L1	[0.25,0.75] m
Chicane matching	drift L2	[0.25,0.75] m
Chicane	drift L1	[0.8,2.0] m
FEL matching	FELMQ1,K_1	[-15,15] m^{-2}
FEL matching	FELMQ2,K_1	[-15,15] m^{-2}
FEL matching	FELMQ3,K_1	[-15,15] m^{-2}
FEL matching	FELMQ4,K_1	[-15,15] m^{-2}
FEL matching	FELMQ5,K_1	[-15,15] m^{-2}
FEL matching	drift L1	[0.25,0.5] m
FEL matching	drift L2	[0.25,0.5] m
Arc 2 matching	A2MQ1,K_1	[-15,15] m^{-2}
Arc 2 matching	A2MQ2,K_1	[-15,15] m^{-2}
Arc 2 matching	A2MQ3,K_1	[-15,15] m^{-2}
Arc 2 matching	A2MQ4,K_1	[-15,15] m^{-2}
Arc 2 matching	A2MQ5,K_1	[-15,15] m^{-2}
Arc 2 matching	drift L1	[0.25,0.5] m
Arc 2 matching	drift L2	[0.25,0.5] m
Arc 2	A21Q1,K_1	[-15,30] m^{-2}
Arc 2	A22Q1,K_1	[-15,15] m^{-2}
Arc 2	A22Q2,K_1	[-15,15] m^{-2}

Continued on next page

Table C.4 – continued from previous page

Section	Parameter	Value
Arc 2	A22Q3,K_1	same as A22Q1
Arc 2	A23Q1,K_1	same as A21Q1
Arc 2	drift L1	[0.25,0.45] m
Arc 2	drift L2	[0.25,0.67] m
Merger matching	EMBQ0,K_1	[-15,15] m^{-2}
Merger matching	EMBQ1,K_1	[-15,15] m^{-2}
Merger matching	EMBQ3,K_1	[-15,15] m^{-2}
Merger matching	EMBQ5,K_1	[-15,15] m^{-2}
Merger matching	EMBQ7,K_1	[-15,15] m^{-2}
Linac matching	EMBTQ6,K_1	[-15,15] m^{-2}
Linac matching	EMBTQ7,K_1	[-15,15] m^{-2}
EDBT	MQD1,K_1	[-60,60] m^{-2}
EDBT	MQD2,K_1	[-60,60] m^{-2}
-	Arc dipoles edge angle	[15°,30°]
-	Chicane dipoles bend angle	[10°,25°]
-	Variable length drifts	[0.0,0.058] m

A series of variable length drifts are inserted into various points on the lattice to provide further freedom in choosing the recirculation time-of-flight.

Undulator Parameters

The list of undulator parameters is shown in table C.5. Some parameters were derived previously in section 4.

Table C.5: List of undulator parameters.

Parameter	Value
Undulator parameter K	0.7
Beam energy	45 MeV
Beam power	0.5 MW
Bunch charge	100 pC
Transverse emittance, normalized	10 µm
Longitudinal emittance, normalized	80 keV-ps

Continued on next page

Table C.5 – continued from previous page

Parameter	Value
RMS energy spread	0.004
RMS bunch length	1 ps
Undulator period	4 cm
Number of undulator periods	25
Undulator length	1 m
Radiation wavelength	4 µm
intracavity saturated power	8 MW

Appendix D

ERL Major Components and Layout

Parameters and coordinates for the ERL baseline presented in chapter 6 are shown here. The format of this section follows that of the E-Linac Phase One design note [29].

Beam and Laser Requirements

ERL beam properties are listed in table D.1. Rare isotope beam properties are listed in table D.2.

Table D.1: ERL beam parameters.

Parameter	Value
Horizontal normalized emittance $\varepsilon_{x,n}$	10 µm
Vertical normalized emittance $\varepsilon_{y,n}$	10 µm
Longitudinal normalized emittance $\varepsilon_{z,n}$	47 µm
Bunch length σ_z	0.0003 m
Energy spread δ	0.019
Electron energy before acceleration	7.5 MeV
Electron energy after acceleration	45.81 MeV
Electron energy after lasing	45.78 MeV
Electron energy after deceleration	7.7 MeV

Table D.2: RIB parameters.

Parameter	Value
Horizontal geometric emittance ε_x	0.3257 µm
Continued on next page	

Table D.2 – continued from previous page

Parameter	Value
Vertical geometric emittance ε_y	0.3257 µm
Longitudinal geometric emittance ε_z	0.331 µm
Bunch length σ_z	0.00026 m
Energy spread δ	0.04
Electron energy before acceleration	10 MeV
Electron energy after acceleration	48.3 MeV

FEL parameters are listed in table D.3. Gain is defined as $(\mathrm{d}P/\mathrm{d}z)/P$, where P is the radiation power. Undulator geometric parameters are listed in table C.5 and are not repeated here.

Table D.3: FEL parameters.

Parameter	Value
Gain	0.5 m^{-1}
Laser wavelength	3.8 µm

RF Requirements

RF requirements are listed in tableD.4 for the four 1.3 GHz cavities (EACA and EACB each contain two cavities). The cavities are phased independently. ERL and RIB injectors are also phase independently.

Convention: phases labeled "initial" are defined with the bunch centroid at the entry point of EACA. This represents the phases of the cavities at the same time. Phases labeled "entrance" are defined with the bunch centroid at the respective cavity entrance. Under this convention 335° is the crest.

Table D.4: RF requirements.

Cavity	ERL phase (deg)	RIB phase (deg)
EACA:CAV1, initial	328.14	337.27
Continued on next page		

Table D.4 – continued from previous page

Cavity	ERL phase (deg)	RIB phase (deg)
EACA:CAV2, initial	164.32	173.45
EACB:CAV3, initial	228.41	237.54
EACB:CAV4, initial	57.73	66.86
EACA:CAV1, entrance	328.14	337.27
EACA:CAV2, entrance	332.64	340.79
EACB:CAV3, entrance	337.67	345.57
EACB:CAV4, entrance	333.04	340.89

Optics Requirements

ERL quad requirements are listed in table D.5. All quads are assumed to be bipolar.

Table D.5: ERL quadrupole requirements.

Quad	KdL (G)
EABT:QEABTQ1	93
EABT:QEABTQ2	-677
EABT:QEABTQ3	682
EHAT:QEHATQ1	-40
EHAT:QEHATQ2	-410
ARC1:QA1M1	3474
ARC1:QA111	4711
ARC1:QA121	1890
ARC1:QA122	-1315
ARC1:QA123	1890
ARC1:QA131	4711
SEP2:QA1M1	3474
SEP2:QEHATQ2	-410
FELM:QFELM1	2062
FELM:QFELM2	-803
FELM:QFELM3	-468
FELM:QFELM4	-1453
FELM:QFELM5	2385

Continued on next page

Table D.5 – continued from previous page

Quad	KdL (G)
A2M:QA2M1	1019
A2M:QA2M2	-3406
A2M:QA2M3	3370
A2M:QA2M4	-2903
A2M:QA2M5	1477
ARC2:QA211	6563
ARC2:QA221	3205
ARC2:QA222	-2728
ARC2:QA223	3205
ARC2:QA231	6563
MERG:QEMBQ0	2766
MERG:QEMBQ1	-645
MERG:QEMBQ3	-172
MERG:QEMBQ5	-270
MERG:QEMBQ7	1099
MERG:QEMBTQ6	-3395
MERG:QEMBTQ7	2124
EDBT:QMQD1	521
EDBT:QMQD2	-159

RIB quad requirements are listed in table D.6. Contains overlap with ERL quads. All quads are assumed to be bipolar.

Table D.6: RIB quadrupole requirements.

Quad	KdL (G)
EABT:QEABTQ1	93
EABT:QEABTQ2	-677
EABT:QEABTQ3	682
EHAT:QEHATQ1	-40
EHAT:QEHATQ2	-410
EHAT:QEHATQ3	1137
EHAT:QEHATQ4	1012

Table D.7: ERL dipole requirements.

Dipole	Path length (m)	Bend angle (deg)	Entry angle (deg)	Exit angle (deg)	BdL (G-cm)
EHAT:BRF	0.5000	0.27	0.00	0.00	715
EHAT:BBSTR	0.3000	0.99	-0.25	1.18	2633
EHAT:BBSEP	0.5000	8.36	0.00	8.36	22306
ARC1:BA11	0.2906	40.19	29.99	29.99	107183
ARC1:BA12	0.2906	40.19	29.99	29.99	107183
ARC1:BA13	0.2906	40.19	29.99	29.99	107183
ARC1:BA14	0.2906	40.19	29.99	29.99	107183
SEP2:BBSEP	0.5000	8.36	8.36	0.00	22306
SEP2:BBSTR	0.3000	1.26	1.18	-0.25	3348
CHI:BCHI1	0.3500	23.15	0.00	23.15	61747
CHI:BCHI2	0.3500	-23.15	-23.15	0.00	-61747
CHI:BCHI3	0.3500	-23.15	0.00	-23.15	-61747
CHI:BCHI4	0.3500	23.15	23.15	0.00	61747
ARC2:BA21	0.2906	45.00	29.99	29.99	119936
ARC2:BA22	0.2906	45.00	29.99	29.99	119936
ARC2:BA23	0.2906	45.00	29.99	29.99	119936
ARC2:BA24	0.2906	45.00	29.99	29.99	119936
MERG:BMBA1	0.1500	-4.61	-2.30	-2.30	-12281
MERG:BMBA2	0.3000	9.22	4.61	4.61	24563
MERG:BMA3	0.1500	-4.61	-2.30	-2.30	-12281
EDBT:BBSTR	0.3000	6.00	0.00	6.00	2720

ERL dipole requirements are listed in table D.7. All eight arc dipoles (four per arc) have the same geometry. The four arc 1 dipoles operate at lower strengths than the arc 2 dipoles since they do not need to account for the full 180° turn. Some of the turn is absorbed by the separator and mirror separator.

All four chicane dipoles have the same geometry.

Installation Coordinates Table

We provide the list of ERL transport elements and their layout in the TRIUMF E-hall. The list begins with the first linac cavity. In all the following:

- Uses TRIUMF standard Cyclotron Center coordinates (Cyclotron Center is (0,0))
- S: Cumulative path length with S=0 at RIB injector cathode
- X: +:East; -:West in Cyclotron Center coordinates
- Y: +:North; -:South in Cyclotron Center coordinates
- Naming convention: elements beginning with 'O' are drifts; 'Q' are quads; 'B' are dipoles
- Coordinates refer to the end of the element

Table D.8: ERL element coordinates. Layout shown in fig. 4.1.

Element	**S (m)**	**X (m)**	**Y (m)**
EACA:START	9.1595	-36.8470	-3.0114
EACA:CAV1	10.4395	-36.8470	-1.7314
EACA:O12	11.0895	-36.8470	-1.0814
EACA:CAV2	12.3695	-36.8470	0.1986
EABT:ODN04	12.7752	-36.8470	0.6043
EABT:QEABTQ1	12.8552	-36.8470	0.6843
EABT:ODN05A	13.0895	-36.8470	0.9186
EABT:BEABD	13.2395	-36.8470	1.0686
EABT:ODN05A	13.4737	-36.8470	1.3028
EABT:QEABTQ2	13.5537	-36.8470	1.3828
EABT:ODN06	13.7737	-36.8470	1.6028
EABT:QEABTQ3	13.8537	-36.8470	1.6828
EABT:ODN07	14.2594	-36.8470	2.0885
EACB:CAV3	15.5395	-36.8470	3.3686
EACB:O34	16.1895	-36.8470	4.0186
EACB:CAV4	17.4695	-36.8470	5.2986
EHAT:ODN08	17.9009	-36.4156	5.2986
EHAT:QEHATQ1	17.9809	-36.3356	5.2986
EHAT:ODN09	18.2149	-36.1016	5.2986
EHAT:BRF	18.7149	-35.6016	5.2974
EHAT:BBSTR	19.0149	-35.3016	5.2934
EHAT:ODN11	20.0149	-34.3016	5.2715
EHAT:QEHATQ2	20.0949	-34.2216	5.2698
EHAT:ODN12	21.0949	-33.2216	5.2479
EHAT:BBSEP	21.5949	-32.7216	5.2005
ARC1:OA1M1	21.8449	-32.4716	5.1588
ARC1:QA1M1	21.9949	-32.3216	5.1337

Continued on next page

Table D.8 – continued from previous page

Element	S (m)	X (m)	Y (m)
ARC1:OA1M2	22.2449	-32.0716	5.0919
ARC1:BA11	22.5355	-31.7810	4.9508
ARC1:OA111	22.9051	-31.4114	4.6685
ARC1:QA111	23.0551	-31.2614	4.5539
ARC1:OA112	23.3855	-30.9310	4.3015
ARC1:BA12	23.6761	-30.6404	4.0341
ARC1:OA121	23.9970	-30.3195	3.7133
ARC1:QA121	24.1470	-30.1695	3.5633
ARC1:OA122	24.5010	-29.8155	3.2093
ARC1:QA122	24.6510	-29.6655	3.0593
ARC1:OA123	25.0050	-29.3115	2.7053
ARC1:QA123	25.1550	-29.1615	2.5553
ARC1:OA124	25.4758	-28.8407	2.2344
ARC1:BA13	25.7664	-28.5501	1.9670
ARC1:OA131	26.0968	-28.2197	1.7147
ARC1:QA131	26.2468	-28.0697	1.6001
ARC1:OA132	26.6164	-27.7001	1.3177
ARC1:BA14	26.9070	-27.4095	1.1766
ARC1:OA1M2	27.1570	-27.1595	1.1348
ARC1:QA1M1	27.3070	-27.0095	1.1098
ARC1:OA1M1	27.5570	-41.0776	9.4208
SEP2:BBSEP	28.0570	-41.1249	8.9235
SEP2:ODN12	29.0570	-41.1469	7.9237
SEP2:QEHATQ2	29.1370	-41.1486	7.8437
SEP2:ODN11	30.1370	-41.1705	6.8440
SEP2:BBSTR	30.4370	-41.1738	6.5440
CHI:OCHI1	30.9195	-41.1738	6.0616
CHI:QCHI1	31.0695	-41.1738	5.9116
CHI:OCHI2	31.5879	-41.1738	5.3931
CHI:QCHI2	31.7379	-41.1738	5.2431
CHI:OCHI3	32.4875	-41.1738	4.4935
CHI:BCHI1	32.8375	-41.1040	4.1529
CHI:OCHI11	33.6607	-40.7804	3.3961
CHI:BCHI2	34.0107	-40.7106	3.0555
CHI:OCHI21	37.3048	-40.7106	-0.2386
CHI:BCHI3	37.6548	-40.7804	-0.5792

Continued on next page

Table D.8 – continued from previous page

Element	S (m)	X (m)	Y (m)
CHI:OCHI31	38.4780	-41.1040	-1.3360
CHI:BCHI4	38.8280	-41.1738	-1.6766
FELM:OFELM1	39.1439	-41.1738	-1.9924
FELM:QFELM1	39.2939	-41.1738	-2.1424
FELM:OFELM2	39.5912	-41.1738	-2.4398
FELM:QFELM2	39.7412	-41.1738	-2.5898
FELM:OFELM3	40.0386	-41.1738	-2.8871
FELM:QFELM3	40.1886	-41.1738	-3.0371
FELM:OFELM4	40.4859	-41.1738	-3.3345
FELM:QFELM4	40.6359	-41.1738	-3.4845
FELM:OFELM5	40.9333	-41.1738	-3.7818
FELM:QFELM5	41.0833	-41.1738	-3.9318
FELM:OFELM6	41.5960	-41.1738	-4.4445
FEL:UND	42.5960	-41.1738	-5.4445
A2M:OA2M1	42.9576	-41.5354	-5.4445
A2M:QA2M1	43.1076	-41.6854	-5.4445
A2M:OA2M2	43.4235	-42.0014	-5.4445
A2M:QA2M2	43.5735	-42.1514	-5.4445
A2M:OA2M3	43.8895	-42.4673	-5.4445
A2M:QA2M3	44.0395	-42.6173	-5.4445
A2M:OA2M4	44.3554	-42.9332	-5.4445
A2M:QA2M4	44.5054	-43.0832	-5.4445
A2M:OA2M5	44.8213	-43.3991	-5.4445
A2M:QA2M5	44.9713	-43.5491	-5.4445
A2M:OA2M6	45.3460	-43.9238	-5.4445
ARC2:BA21	45.6366	-44.2144	-5.3362
ARC2:OA211	46.0518	-44.6296	-5.0426
ARC2:QA211	46.2018	-44.7796	-4.9365
ARC2:OA212	46.4866	-45.0644	-4.7351
ARC2:BA22	46.7772	-45.3550	-4.4735
ARC2:OA221	47.1327	-45.7106	-4.1179
ARC2:QA221	47.2827	-45.8606	-3.9679
ARC2:OA222	47.8945	-46.4724	-3.3561
ARC2:QA222	48.0445	-46.6224	-3.2061
ARC2:OA223	48.6563	-47.2341	-2.5944
ARC2:QA223	48.8063	-47.3841	-2.4444

Continued on next page

Table D.8 – continued from previous page

Element	S (m)	X (m)	Y (m)
ARC2:OA224	49.1619	-47.7397	-2.0888
ARC2:BA23	49.4525	-48.0303	-1.8272
ARC2:OA231	49.7373	-48.3151	-1.6258
ARC2:QA231	49.8873	-48.4651	-1.5197
ARC2:OA232	50.3025	-48.8803	-1.2261
ARC2:BA24	50.5931	-49.1709	-1.1177
MERG:ODQE	50.9931	-49.5709	-1.1177
MERG:QEMBQ0	51.1431	-49.7209	-1.1177
MERG:OMQ	51.6231	-50.2009	-1.1177
MERG:QEMBQ1	51.7731	-50.3509	-1.1177
MERG:OMQ	52.2531	-50.8309	-1.1177
MERG:QEMBQ3	52.4031	-50.9809	-1.1177
MERG:OMQ	52.8831	-36.8470	-5.9045
MERG:QEMBQ5	53.0331	-36.8470	-5.7545
MERG:OMQ	53.5131	-36.8470	-5.2745
MERG:QEMBQ7	53.6631	-36.8470	-5.1245
MERG:ODD	53.7631	-36.8470	-5.0245
MERG:BMBA1	53.9131	-36.8410	-4.8747
MERG:OD1	54.0922	-36.8266	-4.6961
MERG:BMBA2	54.3922	-36.8266	-4.3965
MERG:OD1	54.5714	-36.8410	-4.2179
MERG:BMA3	54.7214	-36.8470	-4.0681
MERG:ODD2	54.7714	-36.8470	-4.0181
MERG:ODN01	55.0094	-36.8470	-3.7801
MERG:QEMBTQ6	55.0894	-36.8470	-3.7001
MERG:ODN02	55.3103	-36.8470	-3.4791
MERG:QEMBTQ7	55.3903	-36.8470	-3.3991
MERG:ODN03	55.7780	-36.8470	-3.0114
EACA:CAV1	57.0580	-36.8470	-1.7314
EACA:O12	57.7080	-36.8470	-1.0814
EACA:CAV2	58.9880	-36.8470	0.1986
EABT:ODN04	59.3937	-36.8470	0.6043
EABT:QEABTQ1	59.4737	-36.8470	0.6843
EABT:ODN05A	59.7080	-36.8470	0.9186
EABT:BEABD	59.8580	-36.8470	1.0686
EABT:ODN05A	60.0923	-36.8470	1.3028

Continued on next page

Table D.8 – continued from previous page

Element	S (m)	X (m)	Y (m)
EABT:QEABTQ2	60.1723	-36.8470	1.3828
EABT:ODN06	60.3923	-36.8470	1.6028
EABT:QEABTQ3	60.4723	-36.8470	1.6828
EABT:ODN07	60.8780	-36.8470	2.0885
EACB:CAV3	62.1580	-36.8470	3.3686
EACB:O34	62.8080	-36.8470	4.0186
EACB:CAV4	64.0880	-36.8470	5.2986
EHAT:ODN08	64.5195	-36.8470	5.7300
EHAT:QEHATQ1	64.5995	-36.8470	5.8100
EHAT:ODN09	64.8334	-36.8470	6.0440
EDBT:BRF	65.3334	-36.8470	6.5440
EDBT:BBSTR	65.6334	-36.8627	6.8434
EDBT:ODQ40	66.0334	-36.9045	7.2413
EDBT:BBSTR2	66.3334	-36.9359	7.5396
EDBT:ODQ	66.4334	-36.9463	7.6391
EDBT:QMQD1	66.5134	-36.9547	7.7186
EDBT:ODQ40	66.9134	-36.9965	8.1164
EDBT:QMQD2	66.9934	-37.0049	8.1960
EDBT:ODQ40	67.3934	-37.0467	8.5938
EDBT:BDMP	67.3944	-37.0468	8.5948