In Situ Modeling of the ERL Injector

Michael Rosenman
Mellon College of Science, Carnegie Mellon University, Pittsburgh, PA 15213
(Dated: August 8, 2006)

The following details a Java- and Matlab-based interface between EPICS and the beam simulation code Parmela for the Energy Recovery Linac (ERL) injector. The purpose of this interface is to allow testing of beam codes against actual beam measurements, the results of which will aid in the design and operation of the future ERL. At present, the interface is fully functional and ready for use once the beamline is operational.

I. INTRODUCTION

The Energy Recovery Linac (ERL) is a future X-ray source being built at Cornell University. The ERL is intended to accelerate high-energy electrons to produce X-rays and then deliberately decelerate them to recovery their energy. It is hoped that this design will produce X-rays of higher quality for new type of experiments.[1]

For the ERL to run properly, its injector must produce intense, high-brightness beams. Our ability to construct and operate such a machine depends heavily on our understanding of beam dynamics. Although beamlines can be tested with simulation codes, these codes are only as good as our understanding of the beam. Therefore, a means of testing simulation codes against an actual beam is needed.[2]

The goal of this project was to construct a programming interface between the Experimental Physics and Industrial Control System (EPICS) and the beam simulation code Parmela. This interface would allow researchers to set up a virtual injector and then test it against actual beam measurements. What follows is a description of this interface.

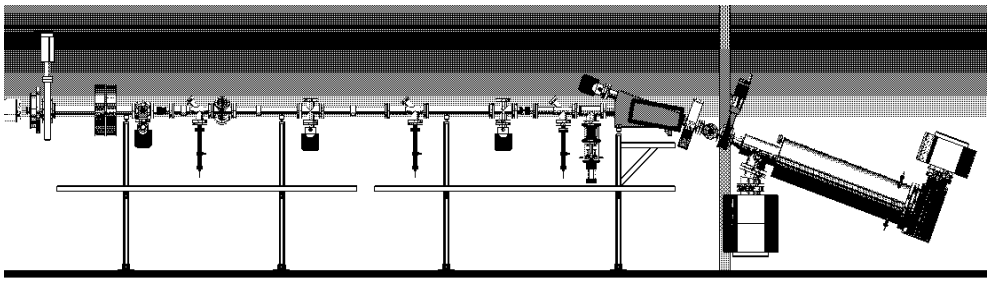

FIG. 1: The ERL Injector

II. OVERVIEW

For this project, we investigated several beam simulation codes and finally decided to use the Los Alamos National Laboratories program Parmela, due to its ability to accurately model the injector's bends and correctors.[3] The interface was intended to be as modular as

possible, so that a new simulation code could easily be substituted for Parmela. Unfortuately, Parmela itself made this goal difficult to fully realize.

The interface consists of the programs EPICSgui and PlotGUI, written in Matlab, and the programs RunParmela, InputConverter, and ProcessOutput, written in Java. The flow of control goes as follows:

EPICSgui is the main user interface. It allows the user to set the parameter values via sliders and text fields and to read data from/write data to EPICS with the *Read EPICS* and *Put Params* commands, respectively.

When the user commands a simulation to be run, EPICSgui generates an input file (whose name may be default, or may be chosen by the user), which lists the name of each parameter, followed by its current value. It then calls the Java program RunParmela.

RunParmela begins by calling InputConverters *convert* method, which parses the input file and a beamline template file, creating a new file called ParmelaExecutable.acc, in which the template file's parameter values have been replaced by those in the input file. RunParmela then commands the Parmela simulation to run, but does not call the post-processor Pargraf. Instead, it invokes the *process* method of ProcessOutput, which uses the data in the Parmela-generated file TAPE3.TXT and generates a data file (whose name may also be default, or chosen by the user).

Control then returns to EPICSgui, which calls PlotGUI and passes it data read from the data file. The process can then begin again. See Figure 1 for a graphical depiction of the control flow.

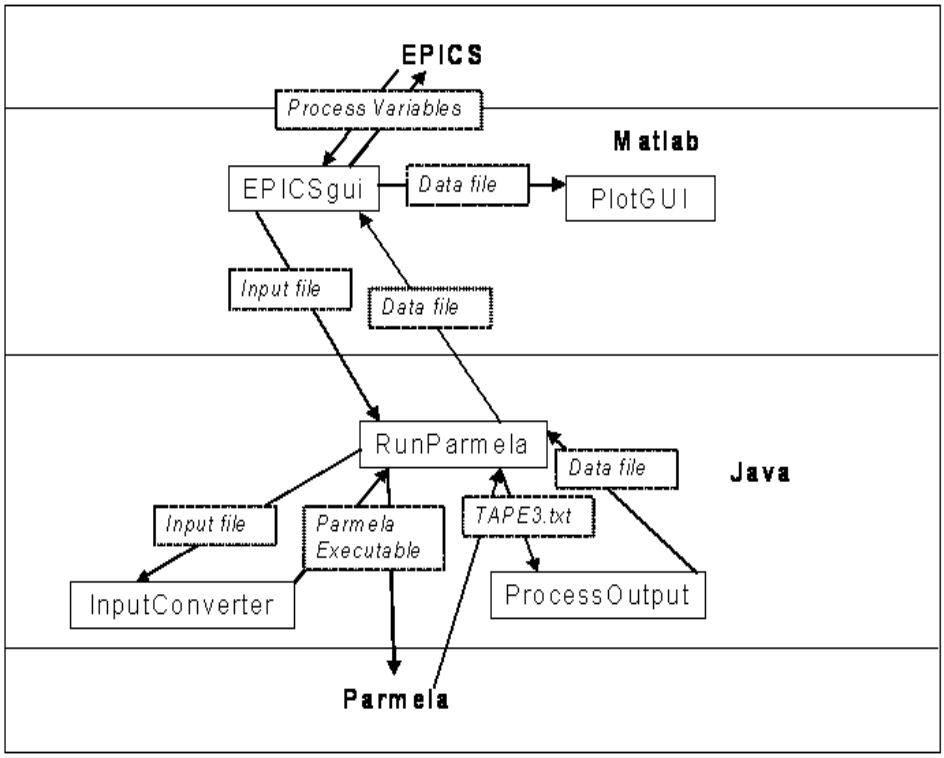

FIG. 2: Flow of Control

III. PROGRAM ELEMENTS

A. Matlab Programs

These are programs the user directly interacts with.

1. EPICSgui

EPICSgui is a Matlab graphical user interface that coordinates the interaction between EPICS and Parmela. The user uses EPICSgui to enter simulation parameters, read from/write to EPICS, and run the simulation. EPICSgui uses the labca package to interface with EPICS. It includes two buttons, *Read Epics* and *Put Params*, which read the values of certain parameters from EPICS or write the current values of those parameters to EPICS, respectively.

The *Run Simulation* button begins the process of running the simulation. First, the GUI creates an input file where the parameters are represented in the following format

name = value

The name of each parameter is hardwired into the GUI. These names have been selected to match the parameter names given in the Parmela template file. These names must match in order for the InputConverter program to function properly. (See below for information on InputConverter). In addition, each parameter is multiplied by a corrective factor to transform EPICS-useable parameters (such as solenoid current) into Parmela-useable ones (such as solenoid field).

The GUI then calls the Java program RunParmela, passing it the input file, a template file (Template.acc), the name of the Parmela executable (ParmelaExecutable.ascii), and the name of the data file. (See below for a description of RunParmela.) When RunParmela is finished running, the data file will have been created and finalized. EPICSgui then loads the data from this file into a *data* field. EPICSgui also has data fields known as *prevData* and *lastRunData*, for use in plotting difference orbits. Both initially are empty. Before each run of the simulation, *lastRunData* is changed to the data from the last simulation. If the user clicks on the *Reset Orbit* checkbox, EPICSgui will also update *prevData* to the values stored in *lastRunData*. If this box is unchecked, *prevData* will not be changed.

EPICSgui then calls PlotGUI, passing it *prevData* and *data*. This completes the simulation. See below for more information on PlotGUI.

2. PlotGUI

PlotGUI is used to display plots of data generated by the simulation. PlotGUI cannot be called on its own; it must be activated by EPICSgui. When EPCISgui calls PlotGUI, it passes to it the data contained in the fields *data* and *prevData*. These are stored and used as the basis for plots.

PlotGUI can generate 18 different plots from the data field. In each case, the values are plotted against the distance Z down the beamline. (See Table I for list of plots). For each plot, PlotGUI scales the output according to default scales hardwired into the GUI. The plot can be scaled as desired using the four text fields and *Rescale* button below the axes.

TABLE I: Plots generated by PlotGUI

Value	Formula	Units
<X>	$\Sigma(X)/numValues$	m
<Y>	$\Sigma(Y)/numValues$	m
<PX>	$\Sigma(PX)/numValues$	eV/c
<PY>	$\Sigma(PY)/numValues$	eV/c
<PZ>	$\Sigma(PZ)/numValues$	eV/c
Xrms	$\sqrt{\Sigma(X- < X >)^2/numValues}$	m
Yrms	$\sqrt{\Sigma(Y- < Y >)^2/numValues}$	m
Zrms	$\sqrt{\Sigma(Z- < Z >)^2/numValues}$	m
PXrms	$\sqrt{\Sigma(PX- < PX >)^2/numValues}$	eV/c
PYrms	$\sqrt{\Sigma(PY- < PY >)^2/numValues}$	eV/c
PZrms	$\sqrt{\Sigma(PZ- < PZ >)^2/numValues}$	eV/c
X Emittance	$\sqrt{(Xrms^2 * PXrms^2) - (\Sigma(X * PX)/numValues)^2}$	m
Y Emittance	$\sqrt{(Yrms^2 * PYrms^2) - (\Sigma(Y * PY)/numValues)^2}$	m
Z Emittance	$\sqrt{(Zrms^2 * PZrms^2) - (\Sigma(Z * PZ)/numValues)^2}$	m
X'	$< PX > / < PZ >$	rad
Y'	$< PY > / < PZ >$	rad
KE	$(\sqrt{MC2^2 + < PZ >^2} - MC2)/1e6$ [a]	MeV
NumParticles	NA	NA

[a] MC2 is the value 510999.06, or mc^2 for electrons in eV/c. The correction factor 1e6 is used to convert the result into MeV.

If the *Difference Orbit* checkbox is checked, PlotGUI will plot a difference orbit, in which the values from *prevData* are subtracted from the values from *data*, and the difference is plotted.

B. Java Programs

The Java programs run the Parmela simulation and return the data to EPICSgui. The user does not interact with the Java programs directly, only through EPCISgui.

1. RunParmela

RunParmela coordinates the actions of the Java programs and communicates with EPICSgui. EPICSgui passes it the input file name, the name of the template file (Template.acc), the name of the Parmela executable (ParmelaExecutable.ascii), and the name of the data file.

RunParmela first calls the convert method of the program InputConverter, passing it the input file, template, and Parmela executable names. See below for more information about InputConverter. Once it has finished running, the Parmela executable file will be ready for use by Parmela.

RunParmela then creates a batch file," ATF.bat", which is used to run the Parmela

simulation. (Parmela executables cannot be called directly: they must be called via a batch file). In particular, RunParmela commands Parmela to run but does not call the Parmela postprocessor Pargraf. Instead, it calls the process method of the Java program ProcessOutput (see below), passing it the name of the data file from EPICSgui. RunParmela then terminates.

2. InputConverter

InputConverter generates Parmela executables based on the EPICSgui input file, which lists the parameters and their values, and a Parmela template file, Template.acc.

Template.acc is itself a functional Parmela input file which details the structure of the virtual beamline. For the program to run properly, the template file must have certain properties. First, each input line must be preceded by a comment line (indicated by a "!" in Parmela), which lists the name of each parameter in the line and its position, with 1 indicating the first element after the name of the input line. There must also be a blank line between each pair of lines. Hence a typical pair of lines in Template.acc has the form

```
!Param1 1  Param2 2  Param3 3
NAME # # #
```

where the # signs stand for parameter values (which can be strings or numbers). It is important to note that the names given for the parameters must match those stored in EPICSgui and the input file it generates, or the program will not work. Any changes to the beamline should be made by changing the file Template.acc, as this is the only file in the program that records the structure of the beamline.

InputConverter first reads in the input file generated by EPICSgui. Each line of this file will have the form.

name = value

where "name" is the parameter name given in EPICSgui. InputConverter reads and parses these lines, storing the names in an *params to change* list and the values at the same index of a *values* list.

InputConverter then steps through Template.acc, parsing the input lines. The NAME of each input line is written out to ParmelaExecutable, and then InputConverter analyzes each parameter following the NAME. If a parameter at a specific index is in the *params to change* list, InputConverter replaces its value with that given in the *values* list and writes the result to ParmelaExecutable. If the parameter is not in the list, InputConverter simply copies the parameter's value from Template.acc to ParmelaExecutable.acc.

Once InputConverter is finished, RunParmela runs a Parmela simulation using the file ParmelaExecutable.acc.

3. ProcessOutput

ProcessOutput processes the output files generated by Parmela into a form suitable for plotting and analysis. For ProcessOutput to work, the InputType parameter of the INPUT line must be set to 9 in ParmelaExecutable.

When input type 9 is used, Parmela will generate a file called TAPE3.txt, which lists the positions and momenta of each particle at the end of each beamline element. These data are broken into groups by beamline element, and each group begins with a line representing the reference particle.

For each beamline element, ProcessOutput reads in the positions and momenta of the particles, skipping over the reference particle, and averages them, writing these averages to the data file specified by EPICSgui and passed to it by RunParmela. It then uses these averages to calculate several quantities, such as rms values, emittances, divergence, and kinetic energy. All of these values are written out on the same line of the data file. Hence each line of the data file represents these values at the end of each beamline element. The data file can then be used for plots by PlotGUI.

IV. MODULARITY ISSUES

Although the interface was desgined to be modular, and allow easy substitution of new simulation codes, several attributes of Parmela have complicated this.

First, Parmela does not permit comment lines everywhere. In particular, there are certain input namelists (POISSON and TITLE) that must be followed immediately by an input string. Hence InputConverter has been designed to look for these lines and deal with them as a special case. While this does not make it incompatible with other programs, it requires extra processing steps that could slow down overall execution.

Second, Parmela may only be called through a batch file. RunParmela therefore generates a batch file and uses it to run the simulation. This would have to be changed to run a different program.

Finally, the Parmela output file, TAPE3.TXT, has a very specific structure that ProcessOutput is designed to recognize. As other simulation programs are likely to produce output files with a different structure, ProcessOutput would have to be reconfigured to work with them.

V. CONCLUSIONS

The interface is currently functional and can be used for beamline testing. In addition, full documentation has been written.

There remains one aspect of the program that requires further work. The exact values for the corrective multipliers used to translate EPICS parameters into Parmela parameters are not known, and must be experimentally determined and set once the the injector test beamline is operational. Otherwise, the interface is ready for use.

VI. ACKNOWLEDGMENTS

I would like to acknowledge Ivan Bazarov and John Dobbins of Cornell University, who oversaw this Research Experience for Undergraduates project and assisted me with my work. This work was supported by the National Science Foundation REU grant PHY-0552386 and

research co-opertaive agreement PHY-9809799.

---

[1] For more information on the ERL, see

- Eds. S.M. Gruner, M. Tigner; I. Bazarov, et.al. "Study for a proposed Phase I Energy Recovery Linac (ERL) synchrotron light source at Cornell University", CHESS Technical Memorandum 01-003, aka JLAB-ACT-0104, 2001
- D.H. Bilderback, I.V. Bazarov, et.al."Energy-recovery linac project at Cornell University", Journal of Synchrotron Radiation, Vol. 10, Part 5 (2003) 346-8

[2] For more information on the ERL injector, see

- I.V. Bazarov, C.K. Sinclair, "Multivariate optimization of a high brightness dc gun photoinjector", Physical Review Special Topics: Accelerator Beam, Vol. 8, 034202 (2005)

[3] For more information about Parmela, see

- Young L.M., PARMELA, Los Alamos National Laboratories, LA-UR-G6-1835, Los Alamos, NM (2000)