SLAC-PUB-7642
August 1997

LEGO: A Modular Accelerator Design Code*

Yunhai Cai, Martin Donald, John Irwin and Yiton Yan
Stanford Linear Accelerator Center, Stanford University,
Stanford, CA 94309

Abstract

An object-oriented accelerator design code has been designed and implemented in a simple and modular fashion. It contains all major features of its predecessors: TRACY and DESPOT. All physics of single-particle dynamics is implemented based on the Hamiltonian in the local frame of the component. Components can be moved arbitrarily in the three dimensional space. Several symplectic integrators are used to approximate the integration of the Hamiltonian. A differential algebra class is introduced to extract a Taylor map up to arbitrary order. Analysis of optics is done in the same way both for the linear and non-linear case. Currently, the code is used to design and simulate the lattices of the PEP-II. It will also be used for the commissioning.

Submitted to *1997 Particle Accelerator Conference, 12-16 May 1997, Vancouver, B.C. Canada*

*Work supported by: the Department of Energy under Contract No. DE-AC03-76SF00515 and DE-AC03-76SF00098

LEGO: A Modular Accelerator Design Code *

Y. Cai, M. Donald, J. Irwin and Y.T. Yan
Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94309 USA

Abstract

An object-oriented accelerator design code has been designed and implemented in a simple and modular fashion. It contains all major features of its predecessors: TRACY and DESPOT. All physics of single-particle dynamics is implemented based on the Hamiltonian in the local frame of the component. Components can be moved arbitrarily in the three dimensional space. Several symplectic integrators are used to approximate the integration of the Hamiltonian. A differential algebra class is introduced to extract a Taylor map up to arbitrary order. Analysis of optics is done in the same way both for the linear and non-linear case. Currently, the code is used to design and simulate the lattices of the PEP-II. It will also be used for the commissioning.

1 INTRODUCTION

There were many accelerator design and simulation codes used for designing lattices for the PEP-II[1] largely due to the complexity of the design. It has been always a dream during the design stage to have one code that can handle everything correctly: purposely off-aligned quadrupole inside a solenoid detector, two beams inside a common quadrupole and non-linear chromatic effects with coupling. It is clear that a code with object-oriented design and implementation is the most natural and powerful approach to handle even more complicated modeling efforts during the commissioning and operation of the machines.

We started to design and implement LEGO two years ago to generate an environment to simulate single charge particle dynamics as a primary goal. The first requirement for the design was that all physics calculation directly related to particles shall be handled in a local coordinate system mounted on the accelerator components. The second requirement was to use differential algebra methods to generate maps and analyze beam dynamics whenever appropriated[2].

We also wanted any applications developed in this environment to be applied to real accelerators in the same way as a simulated machine. Finally, we tried very hard to make our design as simple and modular as possible.

2 BASIC CORE LIBRARY

The core library consists of several inter related modules. They are the parser, beamline, processor, integrator and patch. These modules are designed to be used most effectively as parts of a library. However, they can be used independently as well. For example, a beamline can be constructed directly without using the parser module.

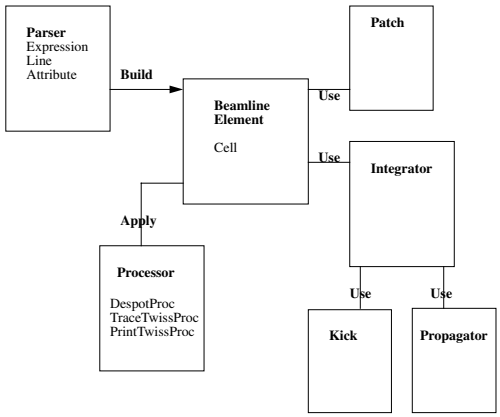

Figure 1: Main Lego Modules

The parser module is for decoding lattice input files. The main function of this module is to read a given input lattice file into tables of parameters, element attributes and symbolic beamlines which then build a beamline with a tree structure for LEGO. The module can be used to parse many common input formats used in the accelerator community, for instance the MAD input. We will discuss some implemented formats later in the section on interfaces.

The beamline module is the core of the library. It defines many components commonly used in accelerators and holds places for the integrators and patches required for physics calculation. It also provides the interface and hook for processors to access elements and travel through the tree-structured beamline sequentially. Together with the processor, they form a visitor pattern[3]. This creates a separation between the beamline and its operations. This is a very desirable feature of a library because additional operations on the beamline can be added using a processor without recompiling the core library.

The processor module is the key of the library. A processor uses the hooks provided by the beamline to manipulate the data of elements and beamline. Most data processing performed on elements, integrators or patches is handled by processors. Applications often use processors to interface with the beamline. One of the most important processors in the module sets up the DESPOT integrators for tracking. Actually, we can replace the engine of the computation

* Work supported by the Department of Energy under Contract No. DE-AC03-76SF00515 and DE-AC03-76SF00098

simply by sending another processor to set up another type of integrators, for example TRANSPORT matrices. Linear and non-linear analysis procedures are unaffected by the swapping of integrators.

The integrator module defines the physics of the beam-transport. An integrator is introduced for the integration of the local Hamiltonian through the body of element including fringe field if needed. Since there are many ways to approximate the integration, the choice of what kind of integrators to use for a given type of element is left for users. In the module, we provide a few processors to set up a consistent set of integrators for instance, DESPOT or TRANSPORT matrices. The integrator makes it possible to separate the description of physical components and how they are used in the calculation of physics. This feature is considered to be one of the major achievements of the library.

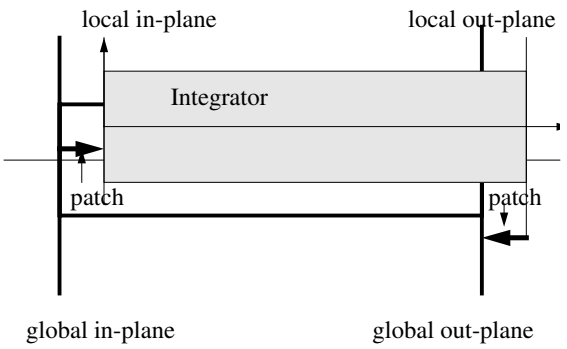

Figure 2: Lego Concept

The patch module handles element or beamline misalignment. Typically, there are two patches for each misaligned component. One is for the entry right before entering the element and another for the exit. Similar to the integrator, the choice of patches can be made by users. A proper selection of the patch allowed us to handle purposely off-aligned quadrupoles inside the solenoid detector.

Many important features have been implemented and tested in the library, for example:

- geometry and survey,
- symplectic integrator,
- synchrotron radiation,
- linear optics,
- element by element tracking,
- non-linear map extraction to arbitrary order,
- fast map tracking,
- non-linear map analysis.

In additional to these closely related modules, we have many independent modules, such as differential algebra, matrix, vector, geometry, fitting and map modules.

3 DESIGN FEATURES

There were many tradeoffs made during the design stage. The most important one is the introduction of the integrator class. In principle, we could carry integrators in a processor while going through a beamline. However, this approach will slow down multi-turn tracking significantly because an integrator needs to be constructed in the element before each calculation. In our design, integrators are saved on the beamline so that setting up integrators only has to be done once.

Similar to the integrator, we also decided to save patches on the beamline for the reason of speed. Actually we gain more than just the speed. Stored patches also allow us to place an element on an existing geometry. This feature is crucial when we are dealing with two beams in the same detector or magnet.

Integrators and patches may be used to carry parameters of the differential algebra. For example, when we study beam-based alignment of quadrupoles the patches for the quadrupole magnets are replaced with parameterized ones so that we can calculate the response of the beam trajectories to their alignemnts. This mechanism of extracting responses is very powerful and conceptually simple.

In the element class, we have built up an inheritance structure that reflects a building-block concept and the perturbative nature of the accelerator theory. The first layer is called "chart" to note the zeroth-order perturbation, namely those elements needed to define geometry and coordinates system. Then the second layer is for components which carry additional attributes for the optics and higher order perturbations.

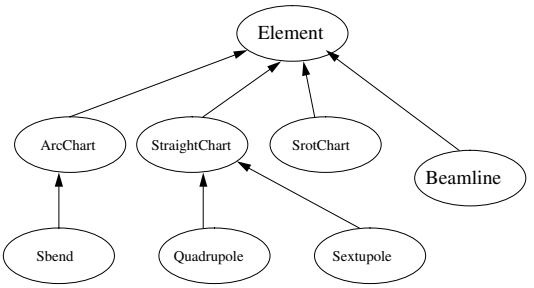

Figure 3: Lego Element Classes

4 APPLICATIONS

Building upon the core library, we have written many useful application programs. They are commonly used to evaluate the performance of the machines when many aberrations are present. Among them, the simplest procedures adjust tunes with two families of quadrupoles and chromaticities with sextupoles. To make a global coupling correction, we implemented a scheme of four families of skew

quadrupoles to zero out the four coupled elements in an one-turn matrix.

For control of the closed orbit, we have implemented the widely used three-bump method. Recently, we added a more powerful scheme correcting orbit and dispersion simultaneously using orbit steering correctors based on eigen-vector decomposition and the MICADO method[4].

Finally, to prepare for the commissioning of the high energy ring(HER) this year, we wrote a beam-based alignment package to determine misalignments of quadrupoles and offsets of beam position monitors. The method of analysis is to fit the beam trajectories for several quadrupole configurations differing by a large percentage in strength while treating the circular accelerator as a single-passage beamline.

Similar to the beam-based alignment scheme, by kicking the beam with correctors and then measuring the beam positions, we can work out the errors of strength in quadrupoles and correctors and gains of beam position monitors. This scheme can be applied to both storage ring and transport beamline.

All the application packages have been simulated for the PEP-II lattice under various conditions and have worked well. They will be applied to the HER soon.

5 INTERFACES

In order to use the applications effectively, we wrote many interfaces to the control system of PEP-II and other existing programs. First, in the parser module, we have implemented two builders for decoding MAD input decks and skeleton decks used in the control system. We are defining our own standard input format to accommodate the new types of element allowed in LEGO.

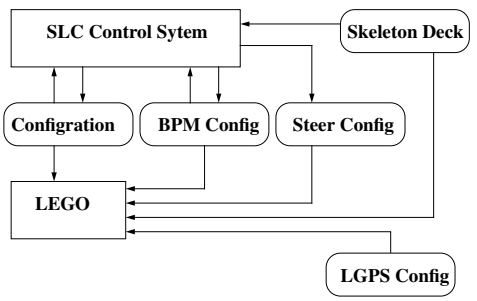

Figure 4: Interface to the control system

Furthermore, we have added the feature of loading configurations and beam position monitor files from the control system to the LEGO beamline so that we can build an offline model easily in the control room and then apply the application programs to the accelerators. This feature also complys with the requirement that an application be used in the same way for both simulated and real machines.

Since the RESOLVE program has proved very useful for commissioning of accelerators, we have developed interfaces[5] so that we can use RESOLVE as a graphical tool to identify machine errors. We plan also to add a graphical user interface for the applications mentioned earlier.

We have linked Zlib[6] to LEGO for extracting non-linear parameterized maps and for non-linear map analysis.

6 SUMMARY

We have created an object-oriented environment for simulating accelerators. It becomes a very efficient tool box to develop new applications both for simulation and operation of accelerators. In this approach, we have achieved five important design specifications. Four of them are related to the modularity of design. They are:

• separation between input language and physical description of element,

• separation between description of element and computational usage of element,

• separation between beamline and its operations,

• separation between analysis of physics and underlining method of transport.

These separations make LEGO very flexible to use and adaptable to challenging design and simulation conditions, like the interaction region of the PEP-II.

The last achievement is the common interface both for simulated machines and real accelerator for all applications.

7 ACKNOWLEDGMENT

We would like to thank Scott Berg, Jim Holt, Chris Iselin, Leo Michelotti, Nick Walker and Johannes van Zeijts for many stimulating and passionate discussions about class design during the CLASSIC meetings. We benefited very much from those discussions. We would also like to thank E. Forest for explaining the physics and procedures implemented in TRACY and DESPOT.

8 REFERENCES

[1] "PEP-II: An Asymmetric B Factory," Conceptual Design Report, SLAC-418, June 1993.

[2] Y. Cai, J. Irwin, Y. Nosochkov and Y.T. Yan, "Computational Tools and Lattice Design for the PEP-II B-Factory," AIP Conference Proceedings 391, CAP 1996,

[3] E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Patterns, Elements of Reusable Object-Oriented Software," Addison-Wesley Professional Computing Series, 1994

[4] M. Donald, Y. Cai, H. Shoaee and G. White, "An Orbit and Dispersion Correction Scheme for PEP-II," these proceedings.

[5] M. Lee *et at.*, "Lattice Commissioning Strategy for the B-Factory", these proceedings.

[6] Y.T. Yan, Y. Cai and J. Irwin, "A Flexible-Variable Truncated Power Series Algebra in Zlib," these proceedings.