**CHEF**

Continually yet sporadically evolving since the early 1990’s, CHEF is a toolkit of libraries written in C++ which provide a framework for instantiating beamline models, tracking orbits of particles through beamline elements, and constructing transition maps between designated locations. Synergia accesses CHEF via its python bindings in order to perform these functions. The name itself is an acronym for

**Collaborative**

objects cooperate to perform complex tasks

**Hierarchical**

comprising a partially ordered collection of C++ class libraries

**Expansible**

designed for future extensions

**Framework**

a set of cooperating object classes that make up a reusable design for a given type of applications [[1]](http://compacc.fnal.gov/~amundson/html/chef.html#id4)

CHEF’s libraries are partially ordered in “layers.” Ideally, classes implemented within a layer may use each other and those in preceding layers but not succeeding ones. [[2]](http://compacc.fnal.gov/~amundson/html/chef.html#id5) These are:

**basic\_toolkit & integrators**

Low level mathematics and utility classes. Earlier versions of these libraries included containers and smart pointers which at one time were not available elsewhere but have since been replaced by classes in [boost](http://www.boost.org/) libraries, the [Standard Template Library](http://www.sgi.com/tech/stl/) or the most current C++ standard. Among those remaining in this layer are: templated algebraic Matrix and Vector classes, Splines, Frames and certain Distribution and integrator classes. In addition, this layer contains the header files MathConstants.h and PhysicsConstants.h, whose use guarantees consistency in the values of fundamental parameters throughout CHEF’s libraries.

**mxyzptlk**

Classes for implementing the functionality of automatic differentiation (AD) and differential algebra (DA). These include Lie operators which are used for normal form analysis and constructing exponential maps.

**beamline**

Classes which model the geometry and physics of beamline hardware and the particles which traverse them. Once instantiated, a hardware object is dynamic, which means some of its attributes and its alignment can be modified at runtime.

**parsers & factories**

Parsers interpret MAD syntax description files (.lat files) and factories instantiate beamline models based on those interpretations. Until very recently this was restricted to MAD 8 syntax, but a MAD-X parser has now been written for Synergia and will be incorporated it into this layer of CHEF.

**physics\_toolkit**

Helper classes that perform specific tasks, such as finding closed orbits, calculating lattice functions or performing normal form analysis.

CHEF also contains a high level layer of graphical user interfaces (GUIs) which Synergia does not use, as it employs its own graphical tools. Synergia does use the python bindings, which are considered part of this layer as they were originally written to serve a python command line GUI to CHEF.

CHEF uses the automatic differentiation features of its mxyzptlk layer to construct polynomial representations of transition maps to any specified order. By taking advantage of C++ operator overloading and templated programming, CHEF has become truly isomorphic in that literally the same lines of code are used for both tracking particles and constructing maps. [[3]](http://compacc.fnal.gov/~amundson/html/chef.html#id6) Synergia can use either, depending on which is more efficient, to propagate bunches between applications of space charge kicks. First order maps are also used to calculate the usual attributes of a lattice – tunes, lattice functions, dispersion, chromaticity and the like – regardless of complications such as coupling, non-traditional closed orbit or alignment errors. At higher orders, CHEF can construct normal form coordinates to study higher order effects, such as amplitude dependence of tunes. Normal form coordinates can also be used to populate bunches in what would be equilibrium distributions in the absence of space charge.

**Code sample**

As a simple example of CHEF’s usage, consider writing a C++ program that calculates and stores Hermite polynomials – i.e. *functions*, not values – in an array. We shall use the definition,



Based on this, an algorithm for generating  can be formulated:



where *g*′(*x*)=*dg*(*x*)/*dx.*

Below is a small program that implements this algorithm to compute Hermite polynomials and store them in an array.

#include <mxyzptlk/Jet.h>

int main( int argc, char\*\* argv )

{

// Preliminary setup

// -----------------

int n = atoi( argv[1] );

IntArray w(1); w[0] = 1;

Jet\_\_environment::BeginEnvironment(2\*n);

coord x( 0.0 );

Jet\_\_environment::EndEnvironment();

// Calculation and storage of Hermite polynomials

// ----------------------------------------------

Jet f = exp( - x\*x );

Jet g = f;

Jet H[n+1]; // Instantiate array to hold Hermite polynomials

for( int k = 0; k <= n; ++k ) {

H[k] = g / f;

g = - g.D( w ); // Differentiation done in this line

}

// ... < additional user code can go here > ...

}

Notice that the syntax of the calculation in the code transparently mimics the algorithm. If invoked with command line argument 4, the program would store the polynomials,

*H*0(*x*)=1

*H*1(*x*)=2*x*

*H*2(*x*)=4*x*2−2

*H*3(*x*)=8*x*3−12*x*

*H*4(*x*)=16*x*4−48*x*2+12

as elements of the array H[0...4]. A user could see the coefficients of, for example, *H*4

, by adding a line

H[4].printCoeffs();

or store the value of *H*2(1.5)

for later use with a line like

double x = H[2](1.5);

|  |  |
| --- | --- |
| [[1]](http://compacc.fnal.gov/~amundson/html/chef.html#id1) | Definition taken from [www.objs.com/survey/ComponentwareGlossary.htm](http://www.objs.com/survey/ComponentwareGlossary.htm) |

|  |  |
| --- | --- |
| [[2]](http://compacc.fnal.gov/~amundson/html/chef.html#id2) | This ideal is broken in a few places but not many. |

|  |  |
| --- | --- |
| [[3]](http://compacc.fnal.gov/~amundson/html/chef.html#id3) | Refactoring CHEF into its current templated form was designed and accomplished by Jean-Francois Ostiguy of Fermilab’s Accelerator Division. While templates are used internally, they are invisible at the application level, so a user need know nothing about templated programming. |

|  |  |
| --- | --- |
| [4] | Mathematically, a jet is an equivalence class of functions with matching derivatives to a specified order. The obviously simplest representative of a jet is a polynomial. |

**Additional information**

There is little documentation on CHEF, but some information is available online:

[CHEF: An Interactive Program for Accelerator Optics Calculations](http://accelconf.web.cern.ch/AccelConf/p05/PAPERS/FPAT006.PDF)

[CHEF: a Framework for Accelerator Optics and Simulation](http://accelconf.web.cern.ch/accelconf/icap06/PAPERS/TUAPMP02.PDF)

[Recent Improvements to CHEF, a Framework for Accelerator Computations](http://accelconf.web.cern.ch/AccelConf/PAC2009/papers/tu2pbc02.pdf)

[Theory and Praxis of Map Analysis in CHEF; Part 1: Linear Normal Form](http://lss.fnal.gov/archive/test-fn/0000/fermilab-fn-0826-cd.pdf)

*This describes and proves the algorithm for correctly normalizing eigenvectors of the linear part of the one-turn map regardless of dimension.*

[Theory and Praxis of Map Analysis in CHEF; Part 2: Nonlinear Normal Form](http://lss.fnal.gov/archive/test-fn/0000/fermilab-fn-0837-apc-cd.pdf)

[MXYZPTLK version 3.1 user’s guide: A C++ library for automatic differentiation and differential algebra](http://lss.fnal.gov/archive/test-fn/0000/fermilab-fn-0535.pdf)

*This needs to be updated to reflect changes made since 1995, but basic concepts are the same and almost all of the algebraic syntax is intact.*

Two of the earliest papers are not online.

**MXYZPTLK and BEAMLINE: C++ Objects for Beam Physics**

in Advanced Beam Dynamics Workshop on Effects of Errors in Accelerators, their Diagnosis and Correction.

(Corpus Christi, Texas. October 3-8, 1991)

American Institute of Physics. Conference Proceedings No.255 (1992)

**MXYZPTLK: A C++ Hacker’s Implementation of Automatic Differentiation**

in Automatic Differentiation of Algorithms: Theory, Implementation, and Application.

(ed. G. Corliss and A. Griewank) SIAM. 1991.