Lucretia Status and Plans

PT

SLAC

"I hear the roar of the big machine
Two worlds and in-between…"

Remind me – what's Lucretia?

- A Matlab-based toolkit for simulations of beam dynamics in single-pass electron beamlines
  - **Most components are Matlab scripts or functions**
  - **A handful of compiled C functions which run under Matlab ("mexfiles")**
    - Compute intensive activities like tracking
      - Tracks pointlike rays with user-defined charge, not 4-D (or 6-D) macroparticles
        - » Can construct 4-D macroparticles out of small #'s of rays
        - » Lucretia distribution includes functions to do this
    - R-matrix calculation, Twiss propagation, lattice verification
      - Share a lot of code with Tracking
  - **Inspired by Accelerator Toolkit (AT), an electron ring application in Matlab**
- Designed with the ILC in mind
  - **Somewhat useful for other applications – LCLS, XFEL**
  - **Designed from the ground up to be an all-in-one tool for LET studies**
  - **As "bulletproof" as we can make it**
    - Careful memory management
    - Careful error handling
    - Segmentation faults fixed as quickly as possible
    - Runs under Matlab
      - Matlab data (ie, beamline state, etc) preserved even if a fault is detected
- Main code maintained by SLAC ILC team
  - **LIAR and DIMAD not actively supported**
  - **Support for MAD-8 Acc limited**
    - MAD-8 is mainly used as a design code
    - Functionality needed for LET types of simulations not available

What's In Lucretia?

- Accelerator representation
  - **Five cross-referenced global arrays**
    - BEAMLINE, GIRDER, KLYSTRON, PS, WF
    - Support the usual spectrum of elements and physics effects, plus relationships between elements
    - Good representation of errors and offsets
    - Implicit support for multiple beamlines
      - Several beamlines can be stored sequentially in global arrays
      - You can track, compute R-matrices, etc on any subset of interest
- Beam representation
  - **Charge-weighted pointlike rays**
  - **Inherently multi-bunch**
- Beam instrumentation
  - **Captures "observables"**
    - IE, beam position reading, beam size reading
  - **Also captures "unobservables"**
    - Sometimes you need these to understand why your simulation didn't work the way you wanted it to
  - **Built-in support for BPM resolution limits, electrical offsets**
  - **No support in the tracking engine for limitations on other instruments**
    - Harder to quantify
    - Always possible to write a script which sits on top of the tracker and applies customized errors to the profile monitors, charge monitors, etc.
- Seed control
  - **Everything which uses random numbers uses Matlab rand and randn functions**
  - **Control of those generators == control of your seeds!**

Connection to Matlab

- Lucretia functions all run from the Matlab command line
- Compiled functions make use of Matlab functions
  - **Sort, spline, rand/randn, gammaln, pascal, abs, det, plus the Matlab API functions**
- Codebase carefully organized to separate Matlab-dependent code from generic algorithm code
  - **Can in principle replace Matlab connections with, e.g., Octave's**
  - **Haven't really pursued this**
    - SLAC users all have Matlab licenses
    - True Matlab/Octave compatibility level not known
    - Some users like to use other toolboxes, e.g. Simulink
- Some licensing issues with Matlab
  - **Mainly related to running large #'s of seeds simultaneously**
    - Implying large #'s of license seats, which we don't have
  - **Mostly but not entirely addressed with Matlab compiler**
    - Compiler output does not need licenses to run
    - Some toolboxes not available in this mode
  - **Trying to convince Mathworks that SLAC should get educational license rate!**

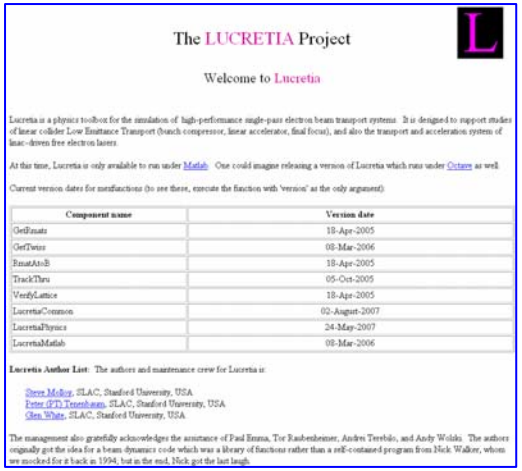
Why Matlab?

- A library / toolkit has more flexibility than a program but takes more user effort to accomplish anything
  - **"The Ikea model – make the customer do all the work!"**
- Matlab supplies a lot of nice features
  - **Command parser (Matlab command line)**
  - **Memory management**
  - **Full-featured graphics**
  - **GUI**
  - **Highly optimized numerical tools**
- Matlab offers a nice mix of batch and interactive operation features
  - **Can run a simulation and then interactively poke around at the results**
- These features are available to greater or lesser extent in a pure compiled library such as BMAD or Merlin
  - **Some people (like me) are more productive with Matlab than in a pure-compiled environment**
    - That's how Mathworks makes their money…

Other Lucretia Features

An online manual in the form of a heavily cross-linked website:

http://www.slac.stanford.edu/accel/ilc/codes/Lucretia/


The LUCRETIA Project

Welcome to Lucretia

Lucretia is a physics toolbox for the simulation of high-performance single-pass electron beam transport systems. It is designed to support studies of linear collider Low Emittance Transport (bunch compressor, linear accelerator, final focus), and also the transport and acceleration system of linac-driven free electron lasers.

At this time, Lucretia is only available to run under Matlab. One could imagine releasing a version of Lucretia which runs under Octave as well.

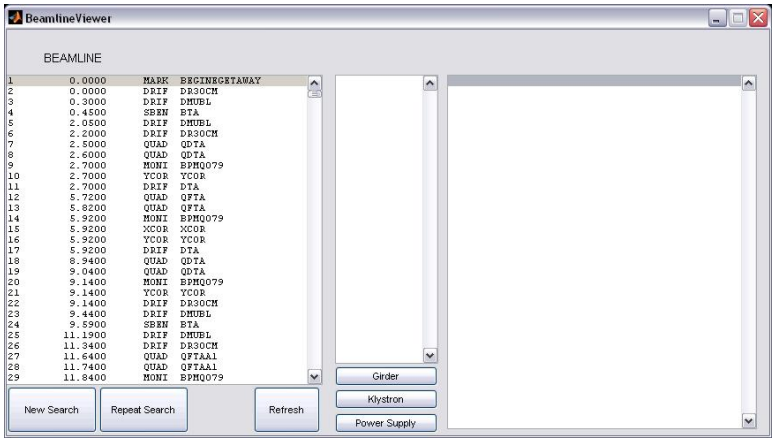
Current version dates for mexfunctions (to see these, execute the function with 'version' as the only argument):

Component name	Version date
GetEmatz	18-Apr-2005
GetTwiss	08-Mar-2006
EmatAtoB	18-Apr-2005
TrackThru	05-Oct-2005
VerifyLattice	18-Apr-2005
LucretiaCommon	02-August-2007
LucretiaPhysics	24-May-2007
LucretiaMatlab	08-Mar-2006

Lucretia Author List: The authors and maintenance crew for Lucretia is:

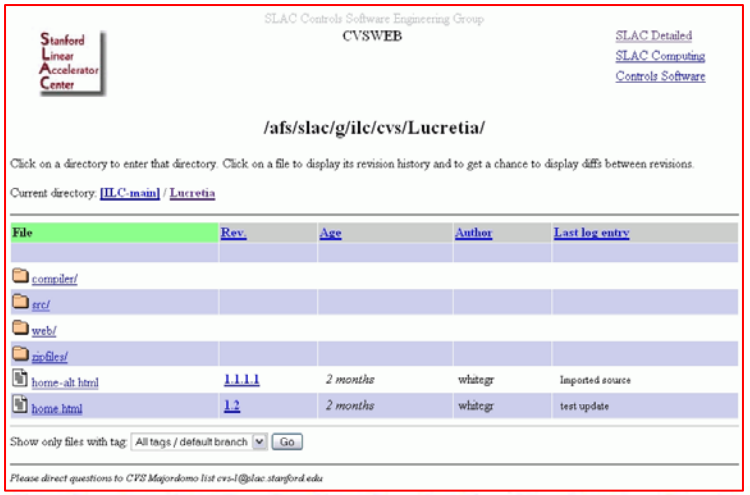
Steve Molloy, SLAC, Stanford University, USA
Peter (PT) Tenenbaum, SLAC, Stanford University, USA
Glen Whate, SLAC, Stanford University, USA

The management also gratefully acknowledges the assistance of Paul Emma, Tor Raubenheimer, Andrei Terebilo, and Andy Wolski. The authors originally got the idea for a beam dynamics code which was a library of functions rather than a self-contained program from Nick Walker, whom we mocked for it back in 1994, but in the end, Nick got the last laugh.



A GUI for studying the beamline data structures (recent)

CVS control via a CVSWEB interface (new!)


SLAC Controls Software Engineering Group

CVSWEB

Stanford Linear Accelerator Center

SLAC Detailed
SLAC Computing
Controls Software

/afs/slac/g/ilc/cvs/Lucretia/

Click on a directory to enter that directory. Click on a file to display its revision history and to get a chance to display diffs between revisions.

Current directory: [ILC-main] / Lucretia

File	Rev.	Age	Author	Last log entry
compiler/				
src/				
web/				
zipfiles/				
home-alt.html	1.1.1.1	2 months	whitegr	Imported source
home.html	1.2	2 months	whitegr	test update

Show only files with tag: All tags / default branch Go

Please direct questions to CVS Majordomo list cvs-l@slac.stanford.edu

ATF2 and Matlab Middleware

- Lucretia is being used for ATF2 simulations
  - **Developing tuning algorithms, etc.**
- Would like to be able to use new algorithms directly on ATF2 itself
- Would also like to be able to get ATF2 machine state data, study it in Lucretia
- Matlab “middleware” being developed
  - **User scripts and functions get / put data into “middleware”**
    - PS settings, BPM readings, etc.
  - **Middleware can talk to either the real accelerator or the Lucretia simulation**

At the same time, a more general purpose tool is being developed to write a machine description from the Matlab middleware to a text file which can be loaded by any application…

Lucretia -- Plans

- What features are needed for ILC LET work which are not yet supported in Lucretia?
  - **Undulator in electron linac**
    - Optical effects, if any
      - Betatron effects – focusing, coupling
      - Dispersion (esp. from errors in the undulator)
    - Energy loss, growth in energy spread from SR
  - **IR Solenoid**
    - Wrapped around IR magnets
    - With a 7 mrad angle wrt the beam line itself
  - **Cavity issues**
    - RF kicks
    - SRWF at cavity center

Undulator

- How to do this?
  - **Lumped element (ie, UNDULATOR element in lattice)**
    - Most concise way to represent it
    - May be rather difficult to implement
    - May not be very accurate
  - **Gigantic array of sector bends**
    - Easy to implement
    - Not very concise
    - Slow in simulation
    - Probably not very accurate
      - But maybe enough for a single-pass lattice?
  - **Field map**
    - Can be as accurate as we need
    - Not concise
    - Slow
    - Difficult to implement
    - Need this anyway for IR solenoid (next slide)
- How to proceed
  - **Get field map from e+ source team**
  - **Study the 3 options above and compare to the field map they give us**
    - IE, how accurate is the sector bend method, how slow is it; if we use a lumped element, would it be better; etc.
  - **Make a decision and go forward**
- Hope to have this implemented by mid-Spring of 2008 (late April?)

IR Solenoid

- Somewhat complicated field profile
- Wraps around lots of IR elements
- Includes anti-solenoid and Detector Integrated Dipole (DID)
  - **Complicated set of SC windings which steer the beam around**
- Tentative conclusion: field map is the best way to represent this
  - **User can specify arbitrary slice spacing**
    - As accurate as he/she needs, with usual execution speed tradeoff
  - **All the strange fields can be included**
    - Obeying Maxwell's equations is the user's obligation
      - Consistent with the "Ikea Model"
  - **Use Matlab's interp2 to do 2-D interpolation of the field maps**

IR Solenoid (2): Field Map

- Tentative implementation plan
  - **Add a FIELDMAP global data structure**
    - Beamline quads, sextupoles, etc. still described by unsliced elements in BEAMLINE
    - FIELDMAPs are superimposed over elements in BEAMLINE
    - FIELDMAPs can use PS, GIRDER data structures
      - IE, a bunch of FIELDMAPs can be powered by one PS, on a GIRDER with a x-ing angle wrt the beamline
- Why not just use superimposed elements, a la BMAD?
  - **Trivial reason: superimposing elements on top of other elements would require radical changes in beamline representation; adding FIELDMAP on top is less drastic**
  - **Non-Trivial reason: IR fields are so complicated we think that a lot of "slices" will be needed no matter what we do**
    - If we can't use a small number of superimposed elements, seems like we might as well use a FIELDMAP
  - **Not absolutely sold on this approach yet**
- Tentative schedule: order of Late May 2008
  - **Want an extra month compared to the undulator program to convince ourselves this is the right idea!**

Cavity Issues

- Not yet sure what to do here!
  - **Waiting for outcome of WakeFest meeting**
    - Hopefully some of these problems will go away!

AML Support

- Plans brewing to migrate ILC lattices from XSIF to AML
  - **see talk in ~40 minutes from now**
- Part of the plan includes AML support for Lucretia
  - **Via a "mex-ed" version of the Universal Accelerator Parser (UAP) and a Matlab function to extract the data from UAP and generate Lucretia data structures**
  - **Will almost certainly do this regardless of AML migration**
    - UAP includes XSIF, MAD, BMAD language support
    - Plan to add SAD support to UAP
    - We get *a lot* of functionality by doing this
    - Consolidates parser support – by keeping updated link to UAP we get all these languages, don't need to maintain a separate XSIF parser, SAD parser, etc.
- Timescale is 2nd half of CY08

Documentation Improvements

- Lucretia has online documentation
  - **Big, cross-linked website**
- Good if you want to know the syntax of some function
- Not so good if you want to learn to use Lucretia
  - **“Like using man pages to try and learn how to use something”**
- Need to add more straightforward tutorial examples
- Time frame of first half of FY08

Conclusions

- Lucretia is a way-cool Matlab based simulation tool for LET studies
  - **Fanatical user base**
    - "Cult of Lucretia"
  - **Expanding role in ATF2 studies / commissioning**
- Several new features planned for FY08
  - **Field maps for IR solenoids**
  - **Some sort of representation for undulator**
  - **UAP support**
  - **Better documentation for new users**
- Other things we might need to do
  - **Improved cavity representation**
- Things we could do but probably won't
  - **Octave version**