The Next Generation of Linear Collider Simulations

ALCPG Meeting
January 2004

GLC/NLC – X-Band Linear Collider
The Past, or, DIMAD to the Bone

Thru the early 1990’s, accelerator simulations were dominated by design codes (TRANSPORT, MAD, DIMAD…)

- Written in FORTRAN
- Text parsing to read deck and commands
  - limited flexibility
- Excellent design tools
  - matching
  - linear optics calculators
  - tracking with/without SR
- Limited capabilities for simulating operation
  - apply errors and misalignments
  - do steering/tuning
  - dynamic errors – things go wrong slowly over time
  - etc etc etc
Simulation program written by K. Oide et al

“Does everything but dig the tunnels”

Similar in many ways to existing programs (self contained with parser etc) but explicitly designed to simulate a tuning process
Software tool designed to simulate high-performance linacs

NOT a design code – no matching module

Physics included was biased towards linacs (single- and multi-bunch transverse wakes in structures, quadrupole physics, misalignments, strength errors, errors in energy profile, etc)

Code engineered for simulating operations – track a beam, generate observables (BPM readings with errors, etc), apply beam-based corrections

Self-contained FORTRAN code – adding new algorithms still cumbersome
For studies of main linac steering feedbacks, coupled feedback design/simulation tools in Matlab with LIAR (All of LIAR ran as subroutine under Matlab control – “mexfiled”)

LIAR generated BPM readings, Matlab scripts calculated corrector settings, applied them to LIAR

MAT-LIAR was born!

Enormously more flexible than LIAR – tuning algorithms in Matlab can get beam-based data from LIAR, set positions/strengths needed for correction

“LIAR is the machine and Matlab is the control room”
LIAR-DIMAD

LIAR still biased towards linacs:

- No momentum compaction
- Can’t handle sextupoles or higher multipoles
- Beam representation allows fast linac tracking, no good for more complex optics

Solution: include DIMAD tracking engine in LIAR, routines to translate beam representation back and forth between the two!
LIAR and its Discontents

In principle, DIMAD/LIAR/Matlab fulfills our requirements – allows flexible simulation of tuning/operation DR→IP←DR. In practice, some limitations

- Hard to maintain – two lattice descriptions, two beam descriptions, matching back and forth
- Transformations in beam representation crude and throw away a lot of information
- A lot of “baggage” from earlier incarnations of LIAR and DIMAD
- Some functionality still only available thru LIAR’s command-line interpreter
AT is a Matlab-based accelerator code

All data (lattice, beam, etc) “lives” in Matlab

Tiny C kernel – performs tracking of rays thru various element types (bends, quads, etc).

Makes intensive use of Matlab’s tools for handling complex and heterogeneous data structures

Hyper-flexible – write a Matlab script to perform any tuning you like!
Where I think the Future Lies

- The LC world needs a fast simulation tool aimed at operations and tuning sims – like LIAR but better
- Should have 1 beam representation (pointlike rays?)
- 1 lattice representation
- Should be able to simulate from DR exit to IP more smoothly than LIAR-DIMAD hybrid
- Amount of compiled code should be minimized
- Any compiled code should probably be in C
  - third-party support for FORTRAN pretty bad these days
- Use of Matlab “wrapper” maximized
  - Why waste our time coding parsers, graphics engines, SVD, etc, when The MathWorks will do it for us?
Why do we use the Standard Input Format (or the Extended Standard Input Format, XSIF) at all?

• It’s what MAD uses, and MAD is a standard design tool
• Big improvement over TRANSPORT format (more human-readable)
• Important to have a standard language we all understand
• Important to have standards to not comply with
In the modern era, we can “write” in a lot of formats without any real access to the “guts” of the format

Example: this talk was prepared by a user typing into PPT; user has no idea what constitutes PPT “format”; doesn’t matter as long as PPT knows!

Can we imagine developing an analogous accelerator data format which can be used by codes which design, layout, simulate accelerators – which users need never see or understand?