Code Fest (not Wars!)

Paul Lebrun

FNAL

Outline

- Goal/motivation for discussing codes.
- Thoughts..
- Specifics brief studies, suggested by PT (and others)
  - LIAR/Lucretia: From MatLab to Octave (increase availability..)
  - Towards XML-based decks..
- ➔ Other such topics could/should be discussed…

Goal in bringing the subject.

- A lot (well.. some) of man-power is available, out there, we should try to optimize it…
  - Newcomers (young or old) to the ILC do not want necessarily to start from scratch
  - They are pragmatic: they go and ask “next door” for some code, and go from there.
  - Next door can be thousands of miles away.. But in many case, it is actually close enough, such collaborations can and do work.

A simple question..

- **What can we simulate, with the required accuracy?**
  - Some problems are non-trivial, beyond our reach, if high accuracy is required : example: in geophysics, earth magnetic field vs time, Tevatron beam lifetime, with a relative accuracy of ~ 100%, 200%...at best
  - Not a competency issue: tough problems..
- For ILC, LET sounds easier than….
  - Collective effects in DR
  - Halo and backgrounds at I.P. ?
- Except: Full dynamical simulation of LET with beam line defects, GM, beam & mechanical jitter and controls problems, including possible halo propagation.

Scope and Schedule for difficult such problems

- Perhaps results from such simulations will never be required by review committees (it was not quite requested for LHC, perhaps I am wrong).
- Not right away at least….we have some time.
- Yet, from now on, the pressure to certify existing designs will keep rising, leaving little time to develop the necessary infrastructure for such difficult problems.
- Now is probably not a bad time to start scoping and thinking..

What do we want to "integrate"? design/simulation vs code.

- While the first is a must, the second is debatable…
  - Simple format or "marked-up language based" decks can describe the machine..
  - Beam files also relatively straightforward to move from one package to an other (unlike HEP data structure)
- Scale matters: if the amount of time spend to "integrate" becomes substantial and this has to be done slightly differently for each code/problem, by each participants, independently, it will become a substantial cost.

Multiple Codes:

- Allows for verification!
  - Done in the past, successfully, albeit on problem relatively straight-forwards, not necessarily on the problems we need to solve.
- Different computational techniques must be applied to solve different problems.
  - Macro particles vs slices, vs simple ray tracing, vs Lie Algebra based beam transport.

Compromise on the number of codes.

- “Supported and certified” : only a few…
  - Such that we move forward on more and more complicate problems..
- Private: more than a few.
  - Such that we can explore new techniques..

Enough generalities, specifics..

- Home work suggested by PT: Matlab vs Octave.
  - Use of MatLab: Motivation:
    - Cost and ease of installation (i.e., obtaining the right of use in a pragmatic sense)
    - Not sure how we can use Matlab based on farms, although it has been done at Daresbury (?)
  - Suggestion:
    - replace with Octave: free-ware.
    - Negotiate with MathWorks (in the works.. But got no feedback recently…)

MatLab -> Octave in Lucretia

- Only brief (very brief) study, in collaboration with Jim Amundsen (FNAL)
  - Matlab scripts are not the problem, they will probably port easily, with minor modifications.
  - The trouble comes with the C (or Fortran) API to MatLab: memory in some (how many?) cases is manage by Matlab, data flows to and from C/Fortran <-> Matlab. Octave has only a C++ interface…
    - Write a C++ interface.. Lucretia C/C++ <-> Octave.
      - Informal Estimate (J.A. !!!) ~ one month full time
      - Maintenance cost not included.
    - Rewrite the C part of Lucretia in C++.
      - Also significant.

XML-Based Decks ???

- Proposed at PAC05.. Or even earlier.
- Goals:
  - No longer have to maintain antiquated MAD parsers.
  - Supports a much richer and accurate description of a machine than currently allowed by MAD(8,9, x..)

But…

- Most, if not all, the XML parsers are written for OO based packages (as opposed to procedural methods).
- Parsing is not the problem, managing and interface to old precepts will be difficult and disruptive..
- Unless the code is already OO (eg., Merlin ??..) ➔ Try this first where it is easier.. Pending..

Other topics..

- ???