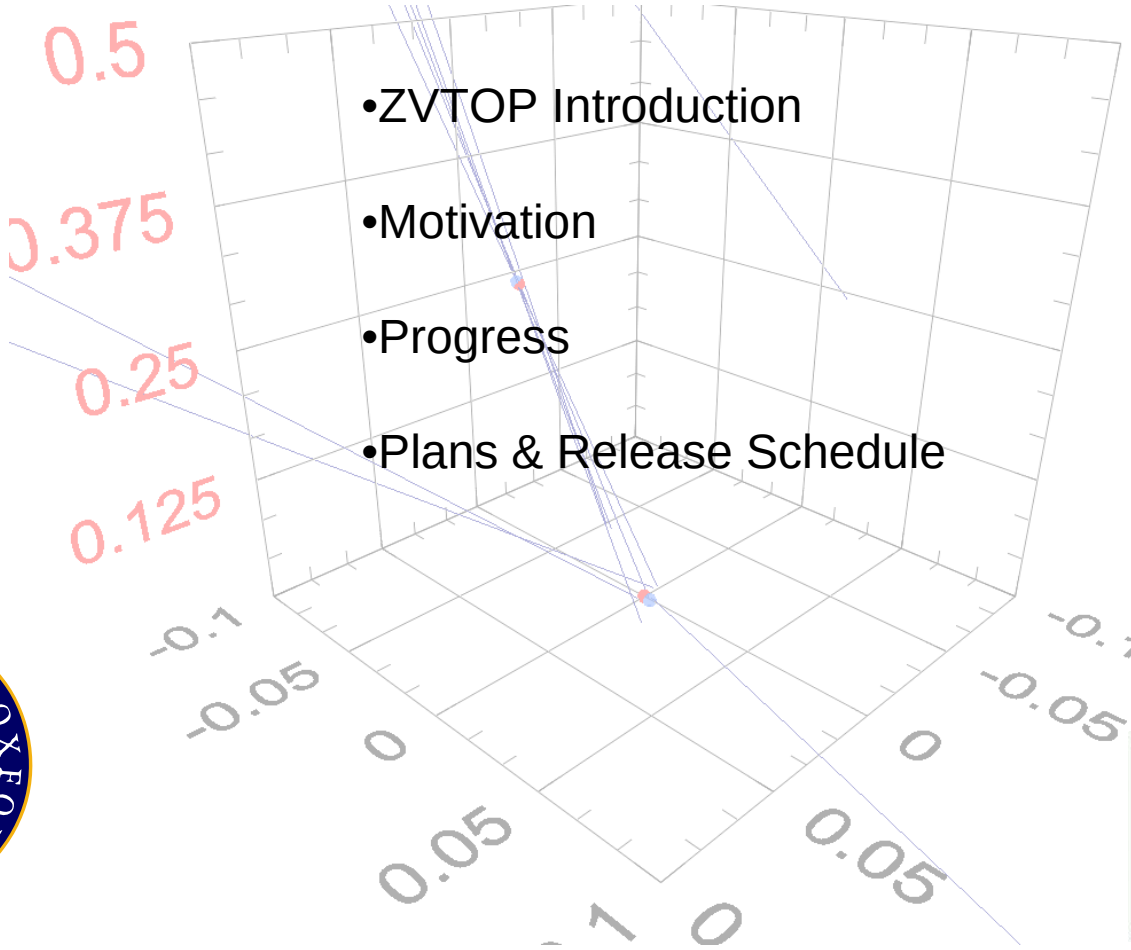
Towards a C++ based ZVTOP

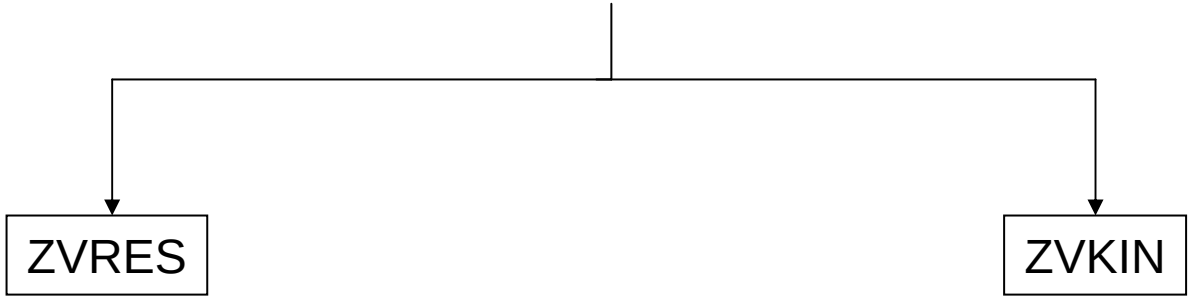
Ben Jeffery (Oxford)
LCFI Collaboration



- ZVTOP Introduction
- Motivation
- Progress
- Plans & Release Schedule

ZVTOP - Introduction

ZVTOP Vertex Finder - initially developed for SLD by Dave Jackson
contains two separate algorithms for different jet topologies:



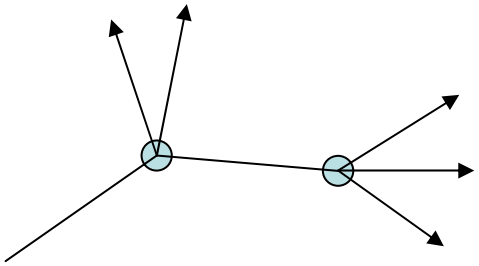
Topological - Multi track vertices

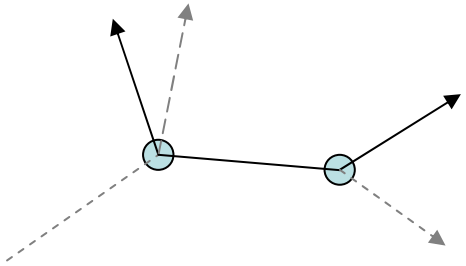
Detailed in:
D. Jackson, NIM A 388 (1997) pp247

In existing FORTRAN based simulation tools, including SGV in use by LCFI.

Kinematic *'Ghost Track'*- includes one track vertices
Used at SLD but not in SGV version

In early development - not detailed in this talk

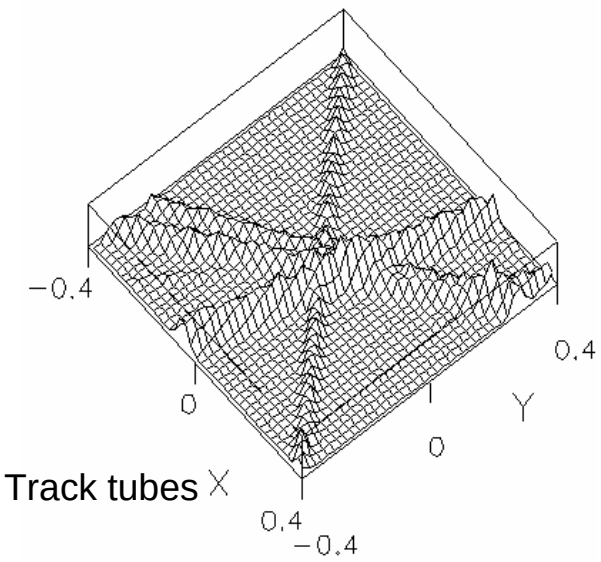




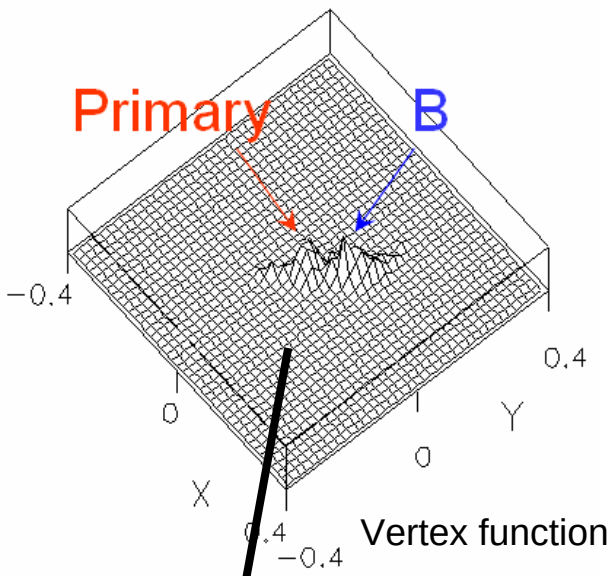
ZVRES - Introduction

Basic Idea: Tracks represented by Gaussian error 'tubes'

Tubes combined to give vertex function:

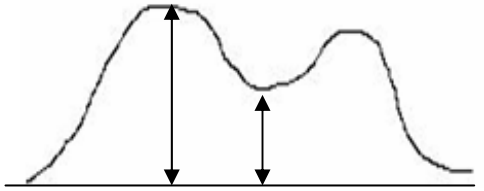


$$\sum_{i=1}^{n} Tube_i - \frac{\sum_{i=1}^{n} Tube_i^2}{\sum_{i=1}^{n} Tube_i}$$



Vertex function peaks resolved into distinct vertices by cut on peak-valley ratio.

Remaining ambiguities in track assignment resolved by magnitude of vertex function.



ZVTOP - Motivation

Verified reference implementation needed in the ILC software framework

⟶ Object oriented C++

Existing FORTRAN code used at SLD exists, and has been used, in fast simulation SGV. (Having come with some modifications via OPAL)

Was a possibility just to wrap this with C++, BUT:

- ZVKIN not included - needed for vertex charge tagging.
- Minimal documentation.
- Difficulty of additions or changes:
  - ILC boundary conditions - all scale dependant parts needed updating.

Approach

Design and code from the original ZVTOP paper.

- Complete understanding and documentation.
- Direct rewrite would not be object oriented.
- Identifies undocumented parts of the FORTRAN by comparison.

ZVTOP - Changes

Some approximations in FORTRAN removed for C++:

Tubes: FORTRAN has parabolic approximation with only diagonal error matrix terms.

Tube = $f_i(\mathbf{r}) = \exp\left\{-\frac{1}{2}\left[\left(\frac{x' - (x'_0 + \kappa y'^2)}{\sigma_T}\right)^2 + \left(\frac{z - (z_0 + \tan(\lambda)y')}{\sigma_L}\right)^2\right]\right\}$

C++ uses helix and full error:

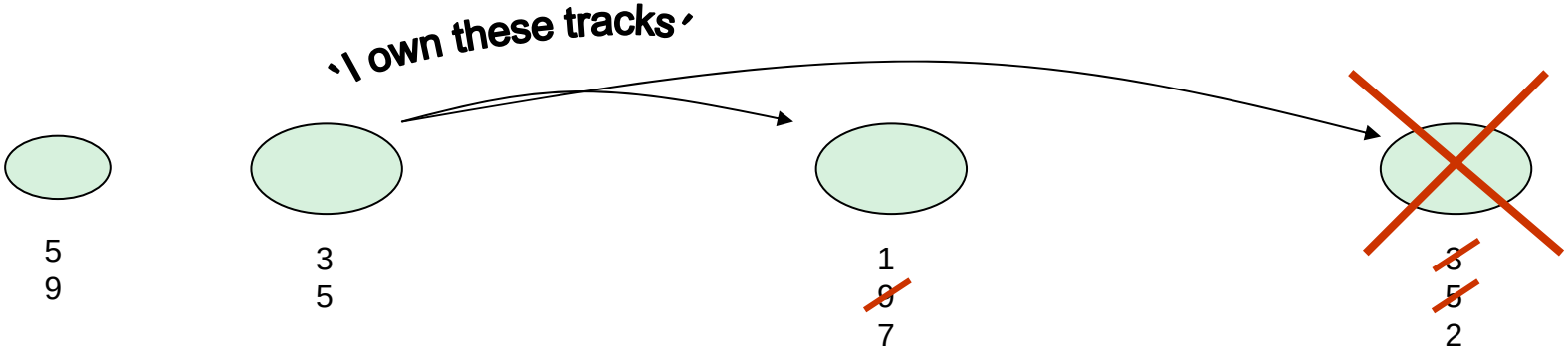
Tube = $f_i(\underline{r}) = e^{(-\frac{1}{2}\underline{p}\,\underline{\underline{V}}^{-1}\underline{p}^T)}$

p = Residual to track
V = Covariance Matrix

Track-Interaction Point and two track fitting changed from analytic approximation to full fit.

Algorithm structure changed for object orientation:

Based around idea of candidate vertices – Merging, track removal etc.
Gives flexibility and can be reused for ZVKIN.

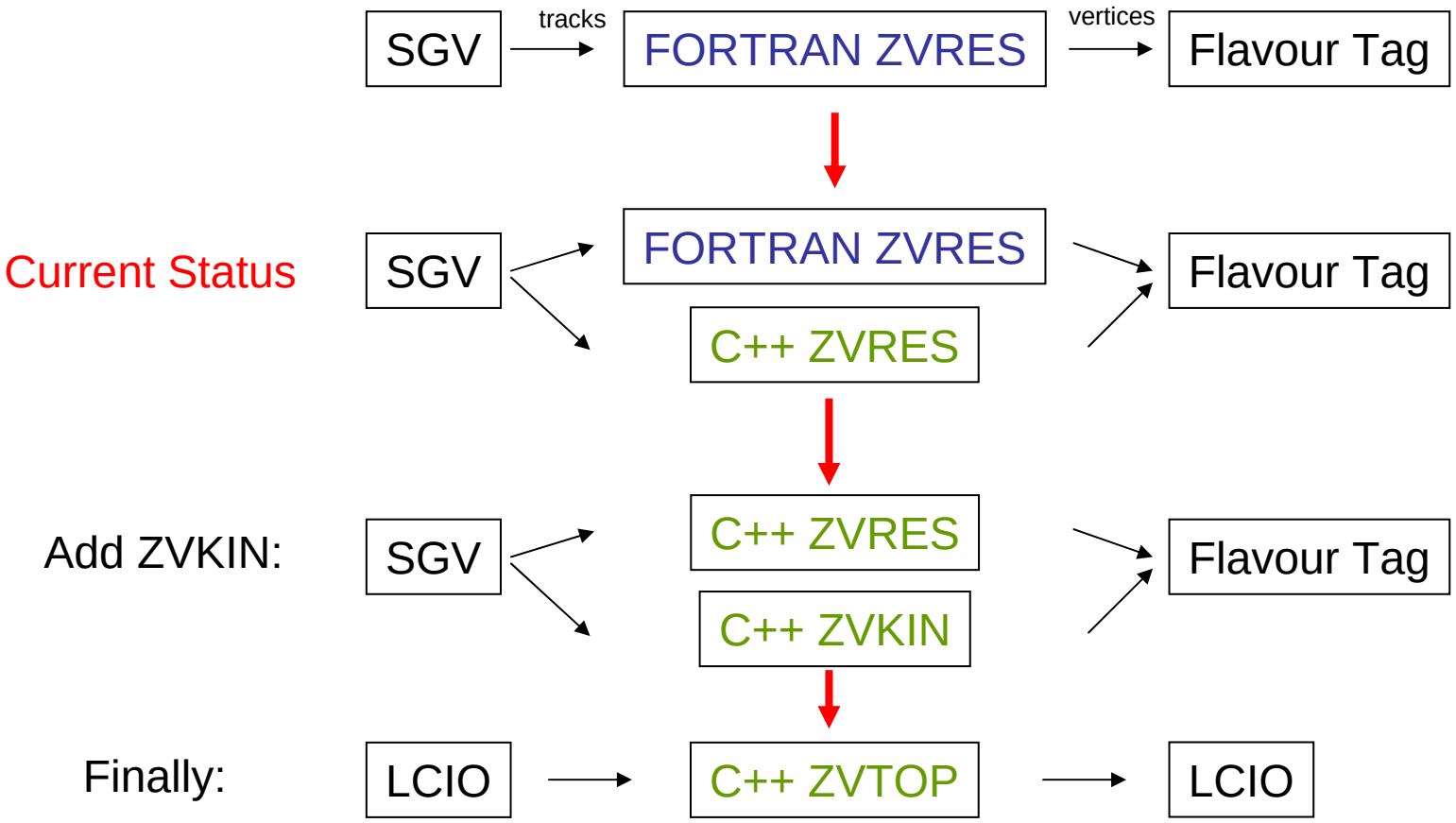


Modular – should allow for change of vertex fitters etc
Current fitter thanks to Mark Grimes at Bristol

ZVTOP - Progress

Initial aim: replace FORTRAN ZVRES in SGV for testing

- allows comparison of intermediate algorithm states when working on identical tracks
- new version can be verified to be at least as good as FORTRAN

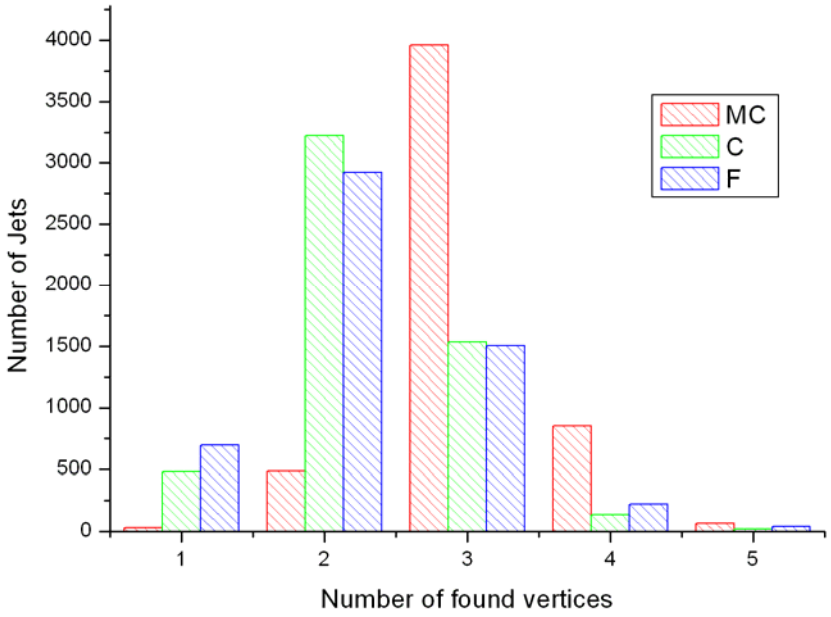


ZVTOP - Progress

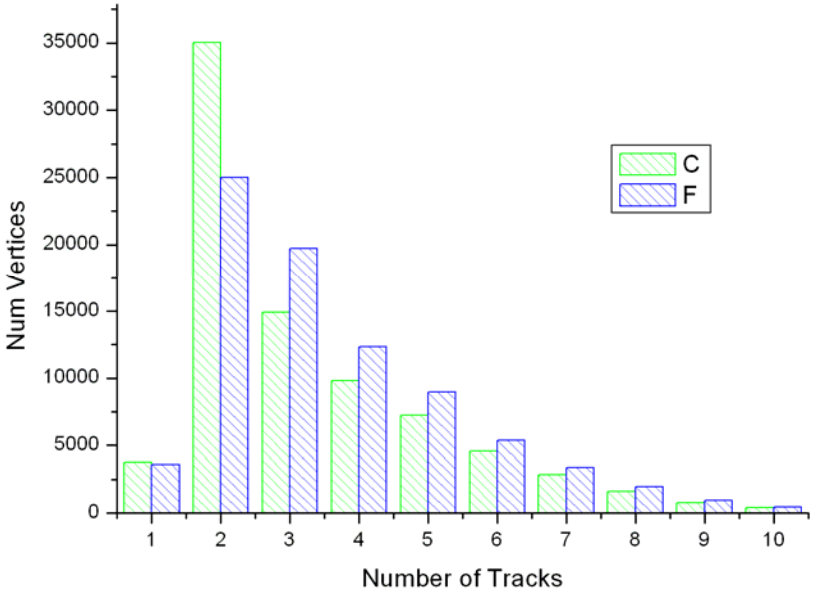
ZVRES Code complete
Detailed comparison testing in progress

Technical details of algorithm in FORTRAN but not in original paper found:

- •Interaction Point handling
- •Vertex Resolution distance cut - important for increased ILC resolution
- •Vertex function magnitude cut

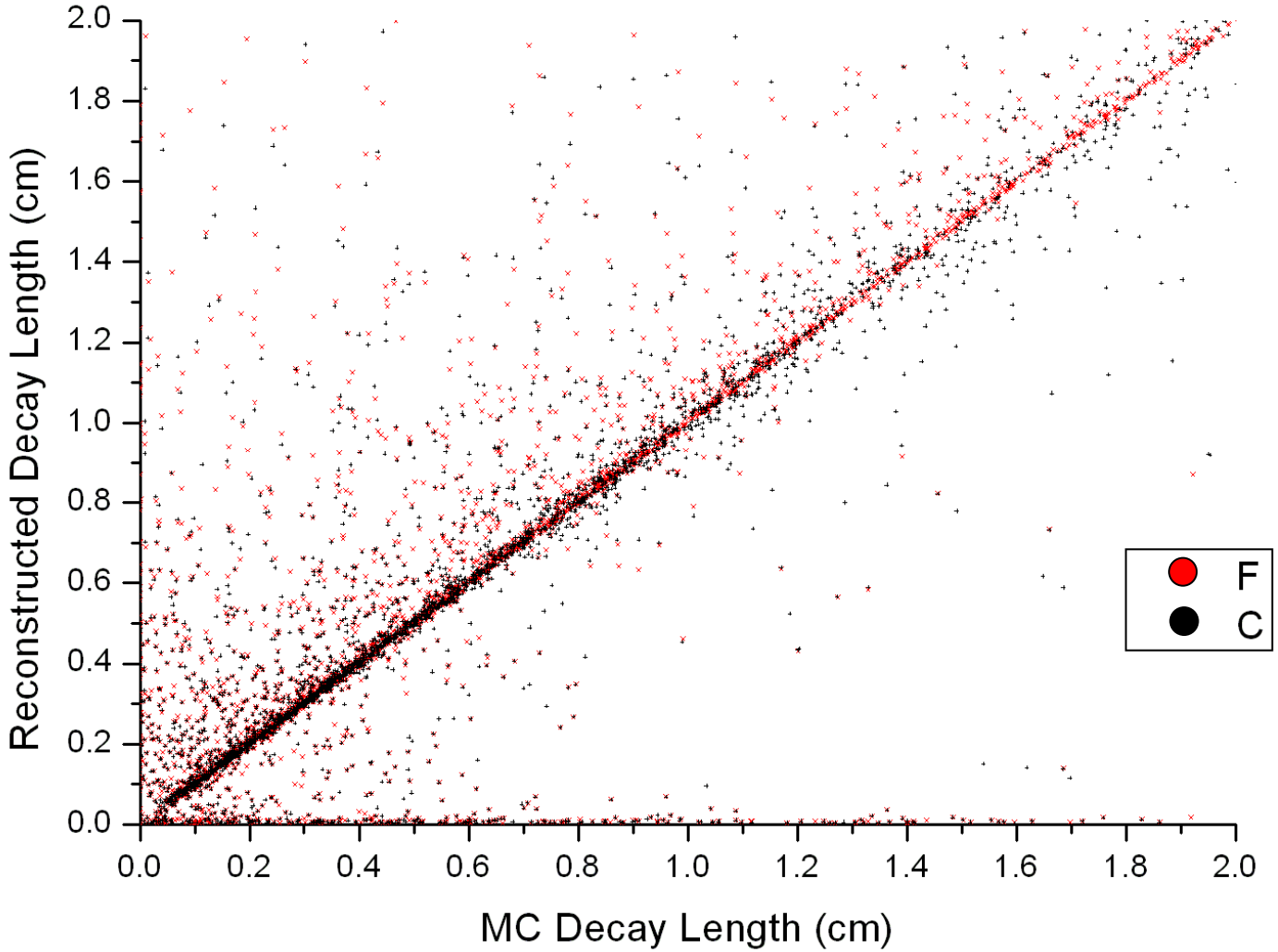


Sample run on B Jets



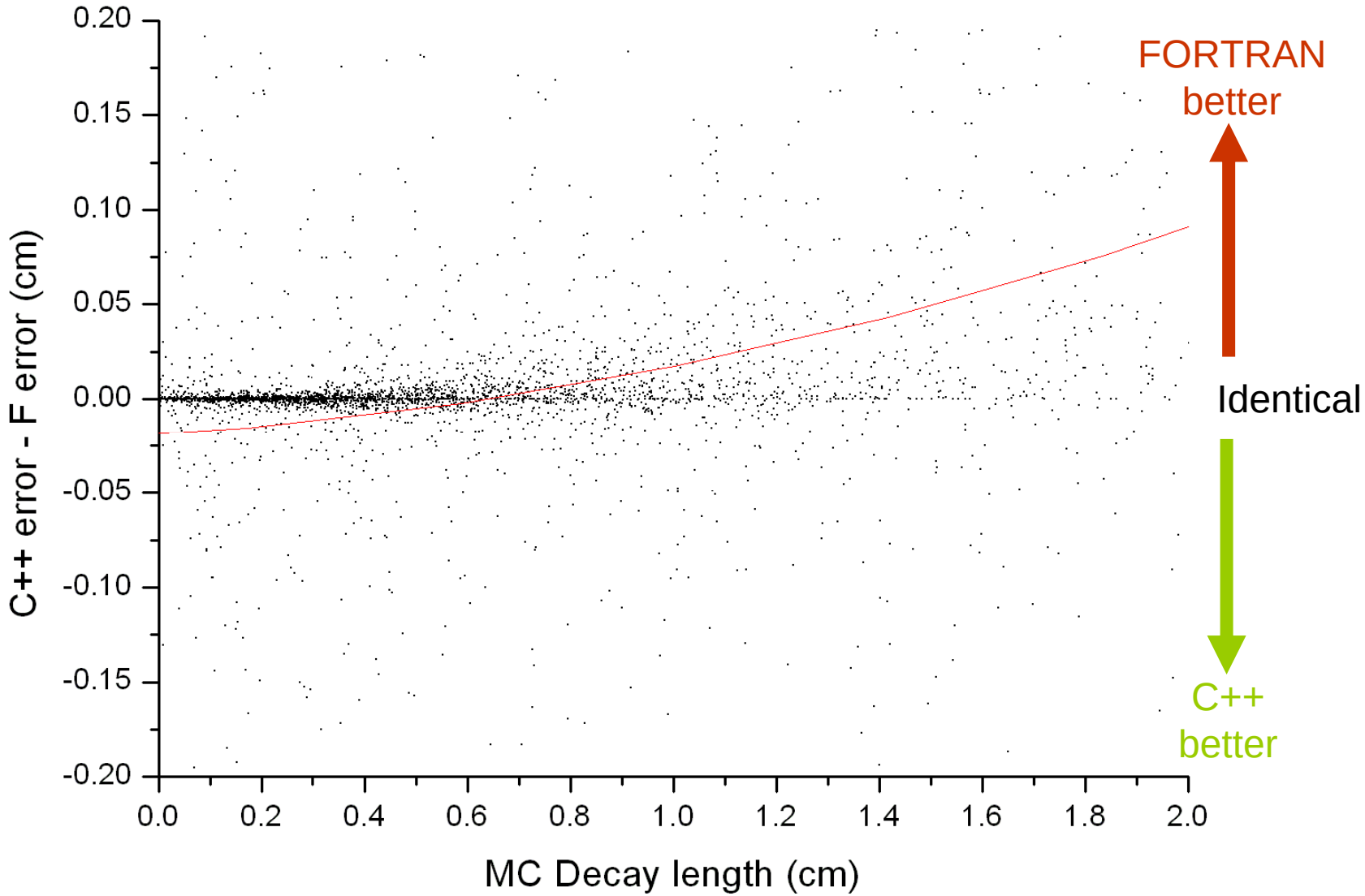
Good agreement between codes, but some differences still to be found

ZVTOP - Progress



First vertex decay length shows good overall qualitative similarity, and a good number of exact coincidences.

ZVTOP - Progress



Absolute difference in decay length error shows problem in c++ at long decay lengths – investigating.

ZVTOP - Future

- Finish ZVRES convergence

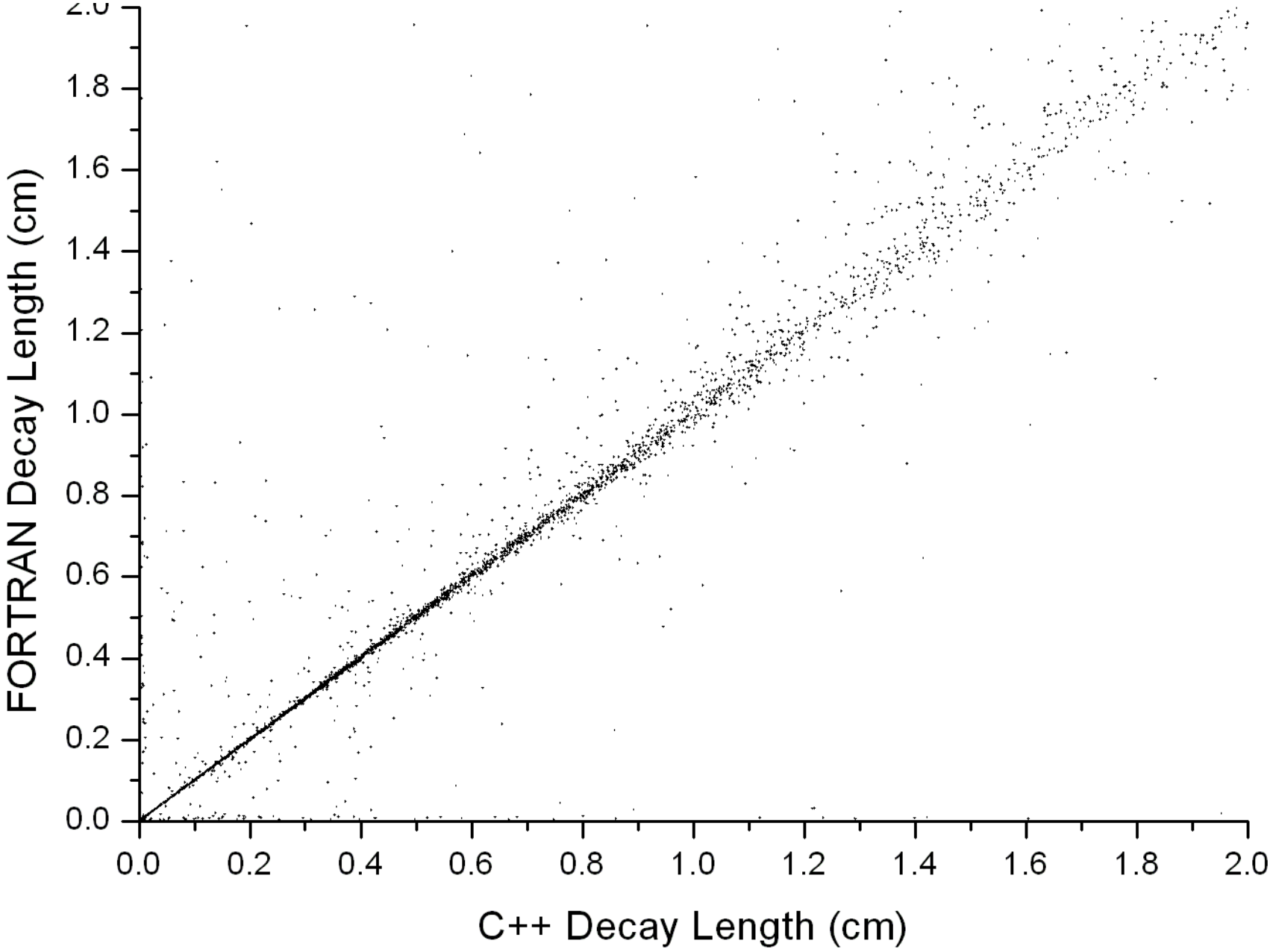
- Write ZVKIN ghost track code based on Dave Jackson's notes
- Test ZVKIN – direct comparison not avaliable, but SLD plots exist.

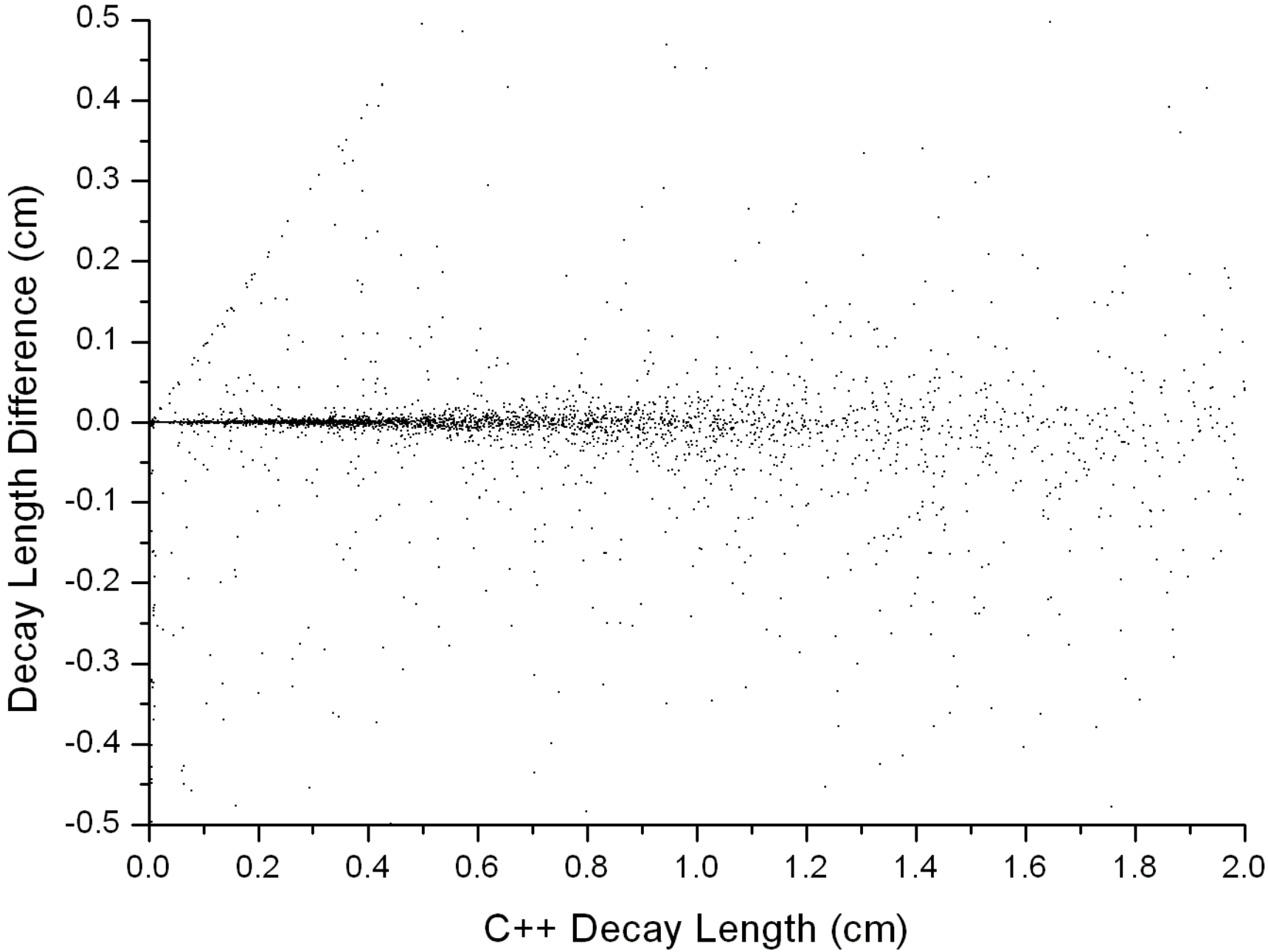
- Polish both ZVRES and ZVKIN

- LCIO Interface

- Documentation, Documentation, Documentation!

- Hoping for release as part of LCFI Vertex Package in June.


FORTRAN Decay Length (cm)
C++ Decay Length (cm)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0


Decay Length Difference (cm)
0.5
0.4
0.3
0.2
0.1
0.0
-0.1
-0.2
-0.3
-0.4
-0.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
C++ Decay Length (cm)