HepMC Proposal

Presentation of the proposal for the new version of HepMC

Lynn Garren

July 25, 2006

background

- CMS requested MathCore GenVector
- Others want no change to Lorentz vector
- Concern about dependencies
- Compromise proposed last week
- A couple more unrelated requests

The Proposal

- Remove all dependencies
- Replace HepLorentzVector with minimal simple vector
  - Prefer struct of four doubles
  - Need a bit more
- Continue to use existing code?
  - Template constructor, but not class
- http://lcgapp.cern.ch/project/simu/HepMC/download/

SimpleVector.h

- class FourVector
  - 4-momentum or position
- class ThreeVector
  - Position
- Independent
  - FourVector does not have a ThreeVector

FourVector constructors

- FourVector(double x, double y, double z, double t=0)
- FourVector(double t)
- FourVector()
- template <class T >

  inline FourVector(const T& v)
  - Uses x(), y(), z(), t()
- Side effect: problem with FourVector(0)
  - Seen, for instance, with GenParticle(0,…)

FourVector methods

- px() py() pz() e() m() m2()
- perp() perp2() mag()
- x() y() z() t()
- theta() phi() rho()
- = == !=
- pseudoRapidity() eta()
- set(,,,) setX() setY() setZ() setT()
- setPx() setPy() setPz() setE()

ThreeVector methods

- x() y() z()
- phi() theta() r()
- mag() perp2() perp()
- = == !=
- set(,,) setX() setY() setZ()
- setPhi() setTheta()

Rationale

- Methods used by HepMC
- Similar methods that seem likely to be used
- Don’t want a full vector class

PdfInfo

- CMS requests PdfInfo class
  - HepMC 1.28?
- GenEvent has pointer (undefined by default)
- double x1; // fraction of beam momentum carried by first parton ("beam side")
- double x2; // fraction of beam momentum carried by second parton ("target side")
- int kf1; // flavour code of first parton
- int kf2; // flavour code of second parton
- double QscalePDF; // Q-scale used in evaluation of PDF's (GeV)
- double pdf1; // PDF (kf1, x1, Q)
- double pdf2; // PDF (kf2, x2,Q)
- Needs to be discussed by LCG simulation

Generated Mass

- Expecting request from Monte Carlo authors
- Store generated mass
  - Additional double
  - Certainly there in HEPEVT common block
  - Precision problem
- Might perhaps store px, py, pz, m
  - Code breaks…
  - Doesn't make sense to calculate mass to store it
    - Must get generated mass directly from generator

Conclusion

- Without templated constructor, not backwards compatible
- With templated constructor, partially backwards compatible
  - Templated constructor overrides default conversion (int to double)
  - Possible obscure errors
  - FourVector is not a full physics vector implementation
- Pere promised something to help when reading
- Is this proposal acceptable?