Geant4e Track Extrapolation in the Belle II Experiment

Leo Piilonen, Virginia Tech on behalf of the Belle II Collaboration

DPF 2017       Fermilab       August 2017

This work supported by U.S. DEPARTMENT OF ENERGY Office of Science
geant4e, a part of geant4, is designed for use during event reconstruction *(not simulation)*. It computes

- the average trajectory of a charged track, assuming a local helix in local magnetic field for each step
- the covariance matrix along this trajectory due to
  - multiple scattering
  - ionization
  - track curvature

using C++ port of the geane code in geant3 (developed by the European Muon Collaboration)
During event reconstruction, use \texttt{geant4e} to propagate charged tracks outward from the drift chamber (helps in particle identification).

\begin{itemize}
  \item \(e^-\) (7 GeV)
  \item \(e^+\) (4 GeV)
  \item Vertex detectors
  \item Drift chamber
  \item Particle identifiers
  \item EM Calorimeter
  \item \(K_L\) and muon detector
\end{itemize}
Use \texttt{geant4}-based model of the Belle II detector:
- detailed detector geometry
- non-uniform solenoidal magnetic field map ($\sim 1.5$ T)
- for \texttt{geant4} simulation \textit{and} \texttt{geant4e} track propagation

\texttt{geant4} simulation of beam-induced backgrounds in Belle II
1) During event reconstruction, **each track** is extrapolated outward using six hypotheses (\( e, \mu, \pi, K, p, d \))

- **Swim each track** from outer edge of drift chamber through the calorimeter (*or until it stops*)
- store time, position, momentum and covariance matrix at entrance/exit of selected **geant4 volumes** (*useful for particle identification*)
2) During event reconstruction, **each track** is extrapolated outward even farther using only $\mu$ hypothesis

- **swim each track** through $K_L$-muon detector with Kalman filter to **matching hits** and track adjustment
- store time, position, momentum and covariance matrix at entrance/exit of each KLM layer
**Geant4e and Geant4:**

Belle II has two usage modes of the \texttt{geant4e} package:

- for reconstruction of real events:
  standalone – as intended by \texttt{geant4/geant4e} authors

- for reconstruction of simulated events:
  coexists with \texttt{geant4} since we do event generation, simulation and reconstruction in a single job

Some difficulties must be overcome!
Geant4e and Geant4, cont’d:

geant4e, as distributed, cannot be used with geant4:
- incompatible particle lists
- incompatible physics processes
- conflicting usage of sensitive-detector geometry
- distinct states when calling RunManager
- distinct step-by-step Navigators
- incompatible user actions (SteppingAction etc)

geant4e, as distributed, is limited:
- propagates only electrons, positrons and photons

We have resolved these issues and limitations. All mods are done outside the geant4(e) code base.
1) Particles and Physics Processes:

- PhysicsList is user’s concrete implementation of G4VUserPhysicsList, and must define:
  - ConstructParticle()
  - ConstructProcess()
  - SetCuts()

- geant4 and geant4e use distinct and incompatible PhysicsLists.

- Significant overhead to change PhysicsList when switching between geant4 and geant4e so avoid this!

Define a combined (and extended) PhysicsList that incorporates geant4 and geant4e functionality.
1) Particles and Physics Processes, cont’d:

- Our modified `ConstructParticle()` defines
  
  - gamma, e+, e-, mu+, mu-, pi+, pi-, pi0, kaon+, kaon-, kaon0, kaon0L, kaon0S, proton, anti_proton, neutron, anti_neutron, geantino, chargedgeantino, opticalphoton, etc for use by geant4
  
  - g4e_gamma, g4e_e+, g4e_e-, g4e_mu+, g4e_mu-, g4e_pi+, g4e_pi-, g4e_kaon+, g4e_kaon-, g4e_proton, g4e_antiproton, g4e_deuteron, g4e_antideuteron (all with PIDcode = 0) for use by geant4e

- Avoids this problem ☞ PhysicsList in the distributed geant4e defines only three particles (gamma, e+, e-) and these conflict with geant4 usage during simulation
1) Particles and Physics Processes, *cont’d*:

- Our modified `PhysicsList()` disables the generation of secondaries – optical and scintillation photons – for newly defined `g4e_*` particles since these processes get attached to *every* charged particle by `geant4`.

- Our modified `SetCuts()` does
  
  `SetCutsWithDefault()` using `default = 1.0*mm` for the regular particles, as in `geant4`

  `SetCutsWithDefault()` using `default = 1.0E9*cm` for the newly defined `g4e_*` particles, as in `geant4e`
During simulation, G4SteppingManager calls user code to process steps through “sensitive” detector volumes and record the hits therein.

During reconstruction, our custom version of StepLengthLimitProcess() disables this behaviour:

```c++
G4ParticleChange aParticleChange;

G4VParticleChange*
    ExtStepLengthLimitProcess::PostStepDoIt( const G4Track& track, const G4Step& )
{
    aParticleChange.Initialize( track );
    aParticleChange.ProposeSteppingControl( AvoidHitInvocation );
    return &aParticleChange;
}
```
3) geant4e navigation and “target” geometry:

- Avoid the special G4ErrorPropagationNavigator in geant4e. Instead, use the standard G4Navigator defined in geant4.

- geant4e requires a target surface (G4ErrorCylSurfaceTarget is an infinite-length cylinder). After each geant4e step, G4ErrorPropagationNavigator would check if the track crossed this surface. Our steering code does this check.

- Our custom version of G4ErrorCylSurfaceTarget is a closed finite-length cylinder that includes the two endcap surfaces.
4) Distinct `geant4`/`geant4e` run states and user actions:

- During our custom `geant4e` initialization, detect its co-existence with `geant4` by a non-empty `G4ParticleTable`.

  - If `geant4e` is running stand-alone, there is no need to preserve the `geant4` state from one event to next.

  - If `geant4e` co-exists with `geant4`, restore the `geant4` idle state and save pointers to its UserActions for swapping out/in during the later track extrapolation:

```cpp
InitGeant4e();
G4StateManager::GetStateManager()->SetNewState(G4State_Idle);
m_savedTrackingAction = UserTrackingAction;
m_savedSteppingAction = UserSteppingAction;
```
4) Distinct run states and user actions, *cont’d*:

During reconstruction of one event:

```cpp
if ( geant4e co-exists with geant4 ) { // hide geant4 actions
    UserTrackingAction = NULL;
    UserSteppingAction = NULL;
}

// extrapolate each track in the event using g4e_* particles;

if ( geant4e co-exists with geant4 ) { // restore geant4 actions
    UserTrackingAction = m_savedTrackingAction;
    UserSteppingAction = m_savedSteppingAction;
}
```
5) Other geant4e modifications:

- The distributed `MagFieldLimitProcess` in geant4e assumes that the magnetic field is along the z axis. Our custom version removes this assumption.

- The distributed `G4EnergyLossForExtrapolator` defines energy-loss processes for electrons and positrons only. Our custom version extends these to muons, pions, kaons, protons and deuterons (and anti-particles).

  - In geant4e, this applies the mean energy loss to each particle during extrapolation. Fluctuations in energy loss and multiple scattering are incorporated in the growth of the covariance matrix.
6) Track-extrapolation use in track–cluster matching:

- Record a crossing ("ExtHit") when the extrapolated track enters/exits each selected volume in the PID detectors or when track is near(est) a reconstructed cluster
7) Track-extrapolation use in muon identification:

- Extrapolate each reconstructed track from the CDC exit point into the KLM (barrel and endcap) using `geant4e`.
  - default is muon hypothesis only
- Look for matching 2D hit upon crossing each KLM layer
- Kalman fitting: If there is a matching 2D hit in the layer, use its position and uncertainty to adjust the position and direction of the extrapolated track before continuing to the next layer.
- Accumulate $\chi^2$ between in-plane hit and track position.
- Finish extrapolation when the track exits the KLM or stops.
- Use extrapolated vs measured range and $\chi^2$/n.d.f. to compute particle-ID likelihoods via PDF-table lookup.
KLM Performance for Muon Identification

Muon identification efficiency (solid curves)

Pion fake rate x 10 (dashed curves)

\[ \ln(\mathcal{L}_\mu / \mathcal{L}_\pi) > 0 \]

\[ \ln(\mathcal{L}_\mu / \mathcal{L}_\pi) > 10 \]

\[ \ln(\mathcal{L}_\mu / \mathcal{L}_\pi) > 20 \]
8) Track-extrapolation of cosmic rays:

- Typical cosmic ray is reconstructed as two tracks
- Lower track #2 is extrapolated forward into bottom half of the detector
- Upper track #1 is extrapolated backward into top half of the detector, using the back-propagation feature of geant4e, so that:
  - energy increases
  - covariance grows
  - time flows backward
Conclusion

In the Belle II software library, we have implemented \texttt{geant4e} track propagation for particle identification (in the PID detectors) and muon identification (in the KLM) during event reconstruction, either standalone or in harmonious co-existence with \texttt{geant4} event simulation:

- merged particle list that comprises \texttt{geant4}-standard and custom \texttt{g4e_*} particles
- distinct physics processes for \texttt{geant4}-standard and custom \texttt{g4e_*} particles
- common \texttt{geant4}-based detector geometry
- no hit invocation in sensitive volumes during \texttt{geant4e}
- distinct states and user actions for \texttt{geant4} and \texttt{geant4e}
- Kalman fitting for muon extrapolation
- all customizations are outside the \texttt{geant4} code base