

Geant 4

March 2006, Geant4 v8.0p01

Bremsstrahlung Splitting with Geant4

Jane Tinslay

SLAC

Stanford
Linear
Accelerator
Center

Outline

- What is variance reduction
- What is uniform bremsstrahlung splitting
- G4WrapperProcess class
- A simple bremsstrahlung splitting implementation
- Summary

Variance Reduction

- Use variance reduction techniques to reduce computing time taken to calculate a result with a given variance
- **Want to increase efficiency of the Monte Carlo**
- Measure of efficiency is given by

$$\varepsilon = \frac{1}{s^2 T}$$

- **s = variance on calculated quantity**
- **T = computing time**

Uniform Bremsstrahlung Splitting

- Aim to increase efficiency by reducing computing time spent tracking electrons
- When a bremsstrahlung interaction occurs, instead of sampling photon energy & angular distributions once, sample distributions N times
 - Create N secondaries
 - Enhances secondary photon production
- Electron energy is reduced by energy of just one photon
 - Energy is not conserved per event

- Remove bias introduced from generating multiple secondaries by assigning a statistical weight to each secondary

$$weight = \frac{Parent\ weight}{N}$$

- N = number of secondary photons
- Preserves photon energy and angular distributions
- Currently, no default bremsstrahlung splitting in Geant4 toolkit
- User can implement bremsstrahlung splitting by
 - Directly modifying bremsstrahlung source code
 - Using G4WrapperProcess
 - May be slightly less efficient
 - Less invasive
 - Easier to implement

G4WrapperProcess

- If want to modify the behaviour of a process, see if it is possible to use G4WrapperProcess
 - G4WrapperProcess is itself a process
- Original process becomes a data member of G4WrapperProcess
- G4WrapperProcess registered with process manager in place of original process
- Function calls to G4WrapperProcess will be forwarded to original process
- Sub-class G4WrapperProcess and override methods to modify original process behaviour

- G4WrapperProcess structure

G4WrapperProcess.hh

```
class G4WrapperProcess :public G4VProcess {
...
protected:
    G4VProcess* pRegProcess;
...
inline
void G4WrapperProcess::RegisterProcess(G4VProcess* process) {
    pRegProcess=process;
...
}
inline G4VParticleChange* G4WrapperProcess::PostStepDolt(
                                const G4Track& track,
                                const G4Step& stepData) {
    return pRegProcess->PostStepDolt(track, stepData);
}
```

- Sub-class G4WrapperProcess to modify original process behaviour

MyWrapperProcess.hh

```
class MyWrapperProcess : public G4WrapperProcess {  
...  
    G4VParticleChange* PostStepDoIt(const G4Track& track, const G4Step& step) {  
        // Do something interesting  
    }  
}
```

- Create new wrapper process, register original process with wrapper, add wrapper process to process manager

MyPhysicsList.cc

```
void MyPhysicsList::ConstructProcess() {  
...  
    MyWrapperProcess* wrapper = new MyWrapperProcess();  
    wrapper->RegisterProcess(originalProcess);  
    processManager->AddProcess(wrapper, ...  
}
```

A Simple Bremsstrahlung Splitting Implementation

- Sub-class G4WrapperProcess
- Modify PostStepDoIt method to generate multiple secondaries
- Register bremsstrahlung process with wrapper
- Register wrapper with process manager

BremSplittingProcess.cc

```
G4VParticleChange* BremSplittingProcess::PostStepDolt(const G4Track& track, const G4Step& step) {
    ...
    G4double weight = track.GetWeight()/fNSplit;

    // Loop over PostStepDolt method to generate multiple secondaries.
    for (i=0; i<fNSplit; i++) {
        ...
        particleChange = pRegProcess->PostStepDolt(track, step);
        ...
        for (j=0; j<particleChange->GetNumberOfSecondaries(); j++) {
            secondaries.push_back(new G4Track(*(particleChange->GetSecondary(j))));
        }
    }
    ...
    while (iter != secondaries.end()) {
        G4Track* myTrack = *iter;
        myTrack->SetWeight(weight);
        // particleChange takes ownership
        particleChange->AddSecondary(myTrack);

        iter++;
    }
    ...
    return particleChange;
}
```

Summary

- Uniform bremsstrahlung splitting technique briefly described
- Simple method to implement bremsstrahlung splitting using G4WrapperProcess introduced
- Other topics to be explored which would improve Monte Carlo efficiency:
 - Russian roulette, selective bremsstrahlung splitting