Interfacing Geant4/Garfield++ and Geant4/Degrad

A step by step introduction

RD51 collaboration meeting Munich
19.06.2018

Carlos Azevedo, I3N - Physics Department, University of Aveiro
Lennert de Keukeleere, IKS KU Leuven
Dorothea Pfeiffer, European Spallation Source ERIC

Interface paper

A Geant4/Garfield++ and Geant4/Degrad Interface for the Simulation of Gaseous Detectors

Dorothea Pfeiffer[a,b,*], Lennert De Keukeleere[c], Carlos Azevedo[d], Francesca Belloni[e], Stephen Biagi[f], Vladimir Grichine[g], Leendert Hayen[c], Andrei R. Hanu[h], Ivana Hřivnáčová[i], Vladimir Ivanchenko[b,j], Vladyslav Krylov[k,l], Heinrich Schindler[b], Rob Veenhof[b,m]

[a] *European Spallation Source (ESS AB),P.O. Box 176, SE-22100 Lund, Sweden*
[b] *CERN, CH-1211 Geneva 23, Switzerland*
[c] *Instituut voor Kern- en Stralingsfysica, KU Leuven, Belgium*
[d] *I3N - Physics Department, University of Aveiro, 3810-193 Aveiro, Portugal*
[e] *CEA Saclay, 91191 Gif-sur-Yvette, France*
[f] *Department of Physics, University of Liverpool, UK*
[g] *Lebedev Physical Institute of RAS, Moscow, Russia*
[h] *NASA Goddard Space Flight Center, Greenbelt, Maryland 20771, USA*
[i] *Institut de Physique Nucléaire, Université Paris-Sud, CNRS-IN2P3, Orsay, France*
[j] *Tomsk State University, Tomsk, Russia*
[k] *Laboratoire de l'Accélérateur Linéaire (LAL), Université Paris-Sud XI, CNRS/IN2P3, 91898 Orsay, France*
[l] *Taras Shevchenko National University of Kyiv (TSNUK), Kyiv, Ukraine*
[m] *Uludag University, 16059 Nülufer-Bursa, Turkey*

Contains model verification and source examples, submitted to NIM A: https://arxiv.org/abs/1806.05880

Task division between Geant4 and Garfield++/Degrad

EUROPEAN SPALLATION SOURCE

Conduction e- produced by		all	charged particles		photons, gammas	
		A) Geant4	B) Interface	C) Heed	D) Heed	E) Degrad
Geant4	Primary particle	All particles	All charged particles	(Relativistic) charged particles	Gammas, photons	X-rays
	Physics model	PAI/PAIphoton as additional EM model in gas region		Standard and low energy EM physics	Standard and low energy EM physics	Standard and low energy EM physics
	Last action	fStopAndKill status: store e- position	Kill produced secondary e- in gas	Kill primary when entering gas	Kill primary when entering gas	Store photon interaction position, kill
Garfield++ Degrad	First Heed or Degrad method		TrackHeed TransportDel-taElectron()	TrackHeed/SRIM SetParticle() NewTrack()	TrackHeed Transport-Photon()	Run Degrad to simulate interaction
	Garfield++	Use conduction electrons from Geant4, Geant4/Heed interface or Heed for Garfield++ simulation according to needs (drift, amplification, signal creation)				

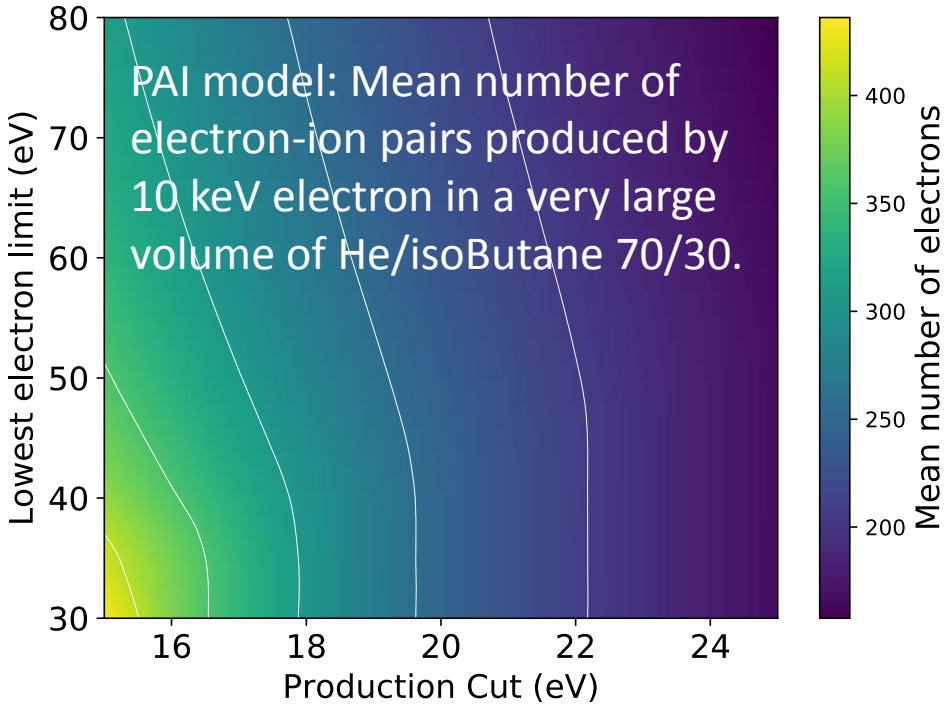
Geant4/Garfield++ interface

```
include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/include)
include_directories($ENV{GARFIELD_HOME}/Include)

link_directories($ENV{GARFIELD_HOME}/Library)
target_link_libraries(IonisationChamber ${
   Geant4_LIBRARIES} -lGarfield -lgfortran )
```

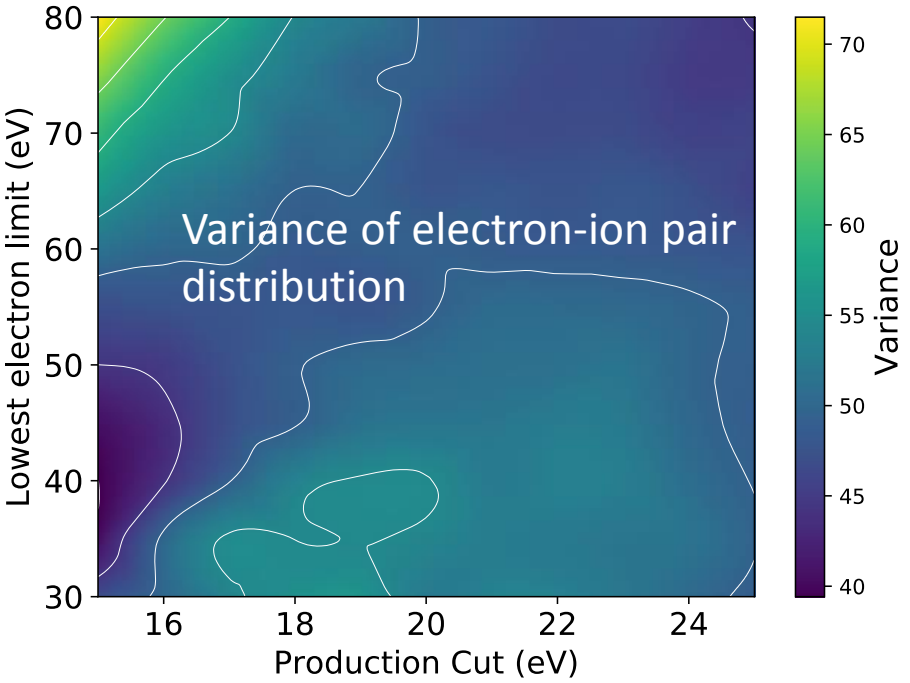
- Geant4 and Garfield++ are used in the same user C++ program
- Garfield++ includes are added to the CMakeList.txt file
- User program is linked against Geant4 and Garfield++ libraries

Geant4 parameters: Lower production cut and lowest electron energy



Lowest electron energy limit:
Energy threshold below which an electron is not tracked anymore. If the kinetic energy of an electron falls below the lowest electron energy limit value during a step, the full energy deposition independently of material is enforced.

Lower production cut:
Minimum energy transfer required to produce a new particle. If the transferred energy is lower than the lower production cut, the energy is merely recorded as being deposited in the step.



Physics list: SetCuts()

```
void UserPhysicsList::SetCuts() {
  /* Geant4 in sequential mode*/
  G4ProductionCutsTable::GetProductionCutsTable()->
   SetEnergyRange(cut*eV, 1.*TeV);
  G4EmParameters* emParams = G4EmParameters::
   Instance();
  emParams->SetLowestElectronEnergy(energy*eV);
}
```

- Lower production cut and lowest electron energy have to be set in SetCut() method of physics list
- If PAI model is used, parameters are of crucial importance to create enough electron-ion pairs
- Deposited energy is not influenced!
- Correct values depend on the gas mixture
- Paper gives guidelines how to determine the correct values

Case A: Geant4 only for creation of electron-ion pairs

EUROPEAN SPALLATION SOURCE

```
void GarfieldSeppingAction::UserSteppingAction(const
    G4Step* theStep) {
  G4Track* track = theStep->GetTrack();
  G4double time = track->GetGlobalTime();
  G4ThreeVector wp = theStep->GetPostStepPoint()->
   GetPosition();
  G4ThreeVector p = theTouchable->GetHistory()->
   GetTopTransform().TransformPoint(wp);
  G4String name = track->GetDefinition()->
   GetParticleName();

  if(name == "e-" && track->GetStatus() ==
   fStopAndKill) {
    /*Send position and time to Garfied++ to
   continue simulation*/
    SendElectronsToGarfield(p.getX(), p.getY(), p.
   getZ(), time);
  }
}
```

Performance of different options for creation of electron-ion pairs

Physics model in gas region	Simulation time per 1000 events
Geant4 PAI model (case A)	4.14 s
Geant4/Heed PAI model interface (case B)	2.24 s
Heed PAI model (case C)	64.20 s

- Comparison of simulation times for 10^5 1 GeV electrons in 1 cm of Ar/CO2 70/30
- Most efficient way of interfacing is to use Geant4 PAI model in conjunction with the Heed PAI model

Case B – D: Heed responsible for creation of electron-ion pairs

```
G4VPhysicalVolume* UserDetectorConstruction::
    Construct() {
  /* Geant4 in sequential mode*/
  G4Region* region = new G4Region("gasRegion");
  region->AddRootLogicalVolume(fLogicalVolumeGas);

  fModel = new GarfieldG4FastSimulationModel("
   UserModel", region);
}
```

- Geant4 has parametrization feature
- G4VFastSimulationModels can be used as alternative to the Geant4 physics
- Definition of a gas region in which the Garfield FastSimulationModel is used

Case B – D: Heed responsible for creation of electron-ion pairs

- Parameterisation has to be added to physics list
- G4VFastSimulation-ManagerProcess has to be defined to trigger the FastSimulationModel
- For charged particles, PAI model has to be activated as extra EM model

```
void UserPhysicsList::AddParameterisation() {
   /* Geant4 in sequential mode:
   Create G4FastSimulationManagerProcess*/
  G4FastSimulationManagerProcess* process =
      new G4FastSimulationManagerProcess("G4FSMP");
  auto it=GetParticleIterator();
  it->reset();
  while ((*it)()) {
    G4ParticleDefinition* p = it->value();
    G4ProcessManager* manager = p->GetProcessManager
   ();
    G4EmConfigurator* config = G4LossTableManager::
   Instance()-> EmConfigurator();
    G4String name = "e-";
    if (p->GetParticleName() == name)   {
      /*Add G4FastSimulationManagerProcess*/
      manager->AddDiscreteProcess(fastSimProcess);
      /*Add PAI or PAI photon model*/
      G4PAIModel* pai = new G4PAIModel(p, "
   G4PAIModel");
      config->SetExtraEmModel(name,   "eIoni", pai,
   "gasRegion", 0.*eV, 1.*TeV, pai);
    }
  }
}
```

Definition of FastSimulationModel

```
class UserFastSimulationModel : public
   G4VFastSimulationModel {

  public:
      /*Constructor and Destructor*/
      UserFastSimulationModel();
      ~UserFastSimulationModel();

      /*virtual methods*/

      /*return true if the G4FastSimulationModel has
    to be applied for a particle type*/
      virtual G4bool IsApplicable(const
   G4ParticleDefinition&);

      /*return true if conditions like kinetic
   energy of G4FastTrack are fulfilled*/
      virtual G4bool ModelTrigger(const G4FastTrack
   &);

      /*The parametrization, i.e. Garfield or Degrad
    related code, should be implemented here*/
      virtual void DoIt(const G4FastTrack&,
   G4FastStep&);
};
```

Definition of FastSimulationModel: IsApplicable()

```
#include "GarfieldG4FastSimulationModel.hh"

G4bool GarfieldG4FastSimulationModel::IsApplicable(
const G4ParticleDefinition& particleType) {
  /*Case B*/
  return &def == G4Electron::ElectronDefinition();

  /*Case C*/
  return &def == G4Proton::ProtonDefinition();

  /*Case D*/
  return &def == G4Gamma::GammaDefinition();

  /*Electroluminescence example*/
  return &def == G4Electron::ElectronDefinition();
}
```

- The FastSimulationModel is applicable for certain particle types

Definition of FastSimulationModel: ModelTrigger()

```
G4bool GarfieldG4FastSimulationModel::ModelTrigger(
    const G4FastTrack& fastTrack) {
  double ekin = fastTrack.GetPrimaryTrack()->
   GetKineticEnergy() / keV;

  /*Case B*/
  if(ekin < 2. * keV)   {return true;}

  /*Case C and D*/
  return true;

  /*Electroluminescence example*/
  if(ekin < 10. * eV) {return true;}

  return false;
}
```

- The FastSimulationModel is triggered at each step for certain particle energies

Definition of FastSimulationModel: DoIt()

```
void GarfieldG4FastSimulationModel::DoIt(const
   G4FastTrack& fastTrack, G4FastStep& fastStep) {
  /*Get world and local position from fast track,
   get momentum vector from fast track, get global
   time in ns, and x,y,z position in cm, get kinetic
    energy of fast track in eV*/
  double ekin_eV = fastTrack.GetPrimaryTrack()->
   GetKineticEnergy() / eV;

  /*Kill track in Geant4 - Garfield takes over*/
  fastStep.KillPrimaryTrack();

  /*Number of electrons produced in a collision*/
  int nc = 0;

  Garfield::TrackHeed* trackHeed = new Garfield::
   TrackHeed();
  trackHeed->EnableDeltaElectronTransport();
```

- The primary particle in Geant4 is killed
- Heed track is created

Definition of FastSimulationModel: DoIt()

- For electrons, Heed TransportDeltaElectron is used
- The produced conduction electrons are retrieved
- For charged particles, Heed NewTrack is used
- First the clusters, and then the conduction electrons are retrieved

```
/*Case B*/
trackHeed->TransportDeltaElectron(x_cm, y_cm, z_cm
 , globaltime, eKin_eV, dx, dy, dz, nc);
for (int cl = 0; cl < nc; cl++) {
  double xe, ye, ze, te;
  double ee, dxe, dye, dze;
  trackHeed->GetElectron(cl, xe, ye, ze, te, ee,
 dxe, dye, dze);
  /*Garfield++ simulation according to needs*/
}
```

```
/*Case C*/
trackHeed->SetParticle("proton");
trackHeed->SetKineticEnergy(ekin_eV);
trackHeed->NewTrack(x_cm, y_cm, z_cm, globaltime,
 dx, dy, dz);
double xcl, ycl, zcl, tcl, ecl, extra;
while (trackHeed->GetCluster(xcl, ycl, zcl, tcl,
 nc, ecl, extra)) {
  for (int cl = 0; cl < nc; cl++) {
    double xe, ye, ze, te;
    double ee, dxe, dye, dze;
    trackHeed->GetElectron(cl, xe, ye, ze, te, ee,
  dxe, dye, dze);
    /*Garfield++ simulation according to needs*/
  }
}
```

Definition of FastSimulationModel: DoIt()

- For photons, Heed TransportPhoton() is used
- The produced conduction electrons are retrieved
- With the conduction electrons, the Garfield++ simulation (drift, avalanches, signal creation) is continued as needed

```
/*Case D*/
trackHeed->TransportPhoton(x_cm, y_cm, z_cm,
 globaltime, eKin_eV, dx, dy, dz, nc);
for (int cl = 0; cl < nc; cl++) {
  double xe, ye, ze, te;
  double ee, dxe, dye, dze;
  trackHeed->GetElectron(cl, xe, ye, ze, te, ee,
 dxe, dye, dze);
  /*Garfield++ simulation according to needs*/
}
```

Geant4/Degrad interface: Electroluminescence example

- As before, a FastSimulationModel is defined and attached to a gas region
- The model is applicable for electrons with Ekin > 7 eV, produced via the photo-electric effect in Geant4
- Degrad is a Fortran program, the easiest way to execute it is to just run it using the system function
- The Degrad input cards are fed to Degrad using a pipe

```
void DegradG4FastSimulationModel::DoIt(const
   G4FastTrack& fastTrack, G4FastStep& fastStep) {
  /*Start by killing the primary track*/
  fastStep.KillPrimaryTrack();

  /*Store the interaction position and time*/
  G4ThreeVector p = fastTrack.GetPrimaryTrack()->
   GetVertexPosition();
  G4double t = fastTrack.GetPrimaryTrack()->
   GetGlobalTime();

  /*Generate a random seed to Degrad*/
  G4int n = 54217137*G4UniformRand();
  G4String seed = G4UIcommand::ConvertToString(n);

  /* Write the Degrad input cards. Example for
   photon of 5.9 keV in pure Xenon (thermalization
   energy of 7.0 eV), in a 3000.0 V/cm electric
   field anti-parallel to the photon direction. The
   gas was considered to be at 20 degrees C and 760
   Torr. For details see Degrad source file */
  G4String degradCards="printf \"1,1,3,-1,"+seed+",
    5900.0,7.0,0.0\n
    7,0,0,0,0,0\n
    100.0,0.0,0.0,0.0,0.0,0.0,20.0,760.0\n
    3000.0,0.0,0.0,1,0\n
    100.0,0.5,1,1,1,1,1,1,1\n
    0,0,0,0,0,0\" > conditions.txt";
  G4int stdout=system(degradCards.data());

  /*Call Degrad with the input conditions file*/
  stdout=system("./degrad < conditions.txt");
```

Geant4/Degrad interface: Electroluminescence example

- Degrad write the position and time of the thermalized electrons to a text file
- Text file is looped through in Geant4
- If position of electron is inside the gas volume (in Degrad the volume is indefinite, hence the check), new secondary electrons are created in Geant4

```
G4ThreeVector ep;
for (int electron=0;electron<
 NumberOfElectronsProduced; electron++) {
  ep.setX(posXDegrad*0.001+ip.getX());
  ep.setY(posZDegrad*0.001+ip.getY());
  ep.setZ(posYDegrad*0.001+ip.getZ());
  time=timeDegrad*0.001+t;
  /*Check if the electron position is inside the
 gas volume. Degrad does not use geometrical
 constraints*/
  G4Navigator* theNavigator =
 G4TransportationManager::GetTransportationManager
 ()->GetNavigatorForTracking();
  G4VPhysicalVolume* myVolume = theNavigator->
 LocateGlobalPointAndSetup(myPoint);
  G4String name=myVolume->GetName();
  /*gasName should be the name of the gas volume*/
  if (name.contains("gasName")) {
    G4DynamicParticle electron(G4Electron::
 ElectronDefinition(),G4RandomDirection(),
 thermalizationEnergy*eV);
    /*Create secondary electron*/
    G4Track *newTrack=fastStep.
    CreateSecondaryTrack(electron, ep,time,false);
    newTrack->SetTrackID(fastStep.
 GetNumberOfSecondaryTracks());
  }
}
```

Garfield++ FastSimulationModel: Electroluminescence example

- Garfield++ FastSimulationModel is valid for electrons with Ekin < 7 eV (these electrons were just created in the Degrad FastSimulationModel)
- AvalancheMicroscopic is used to create an electron avalanche
- Callback function is written to access the excitation levels
- Time and position of excitations are stored

```
  /*Electroluminescence example: Create the
   microscopic avalanche class*/
  Garfield::AvalancheMicroscopic * aval = new
   Garfield::AvalancheMicroscopic();
  /*Connect avalanche microscopic to sensor*/
  aval->SetSensor(sensor);
  /*Set the callback function*/
  aval->SetUserHandleInelastic(
   accessExcitationLevels);
  aval->AvalancheElectron(x0,y0,z0,t0,e0,0.,0.,0.);
  /*Loop over data structure filled in callback
   function, produce optical photons in Geant4
}

/*Define a global callback function*/
void accessExcitationLevels(double x, double y,
   double z, double t, int type, int level,Garfield
   ::Medium * m) {
  /*Access the excitation levels, fill data
   structure with time and position of excitations*/
}
```

Garfield++ FastSimulationModel: Electroluminescence example

- At the stored excitation positions, optical photons are produced in Geant4 as secondary particles in the DoIt() method of the FastSimulationModel

```
 G4DynamicParticle particle(G4OpticalPhoton::
OpticalPhotonDefinition(),G4RandomDirection(),
eKin);

 /*Set a number that is larger than the maximum
number of expected secondaries*/
 fastStep.SetNumberOfSecondaryTracks(1000000);
 /*Create secondary optical photon with random
polarization*/
 fastStep.CreateSecondaryTrack(particle,
G4RandomDirection(), position, time, true);
```

Conclusions

- Geant4/Garfield++ and Geant4/Degrad interface works
- Depending on the use case, different ways exist to interface the Monte Carlo programs
- Paper under review at NIM A
- During work on paper, bugs in Heed C++ implementation detected and corrected
- Possibility to create a Heed like physics process in Geant4 should be investigated Geant4 collaboration to further facilitate the simulations