Garfield++

Journées thématiques du Réseau Semi-conducteurs IN2P3-IRFU, 11 June 2021

- GARFIELD++ is a toolkit for the detailed simulation of signals in particle detectors that are based on ionisation measurement in gases or semiconductors.
- It inherits many concepts and techniques from the Fortran program GARFIELD, which has been widely used for simulating gas-based detectors.
- The development of the C++ version of started $\sim$ 2011.

Microscopic simulation of electron avalanches in a GEM (left) and around a wire (right).

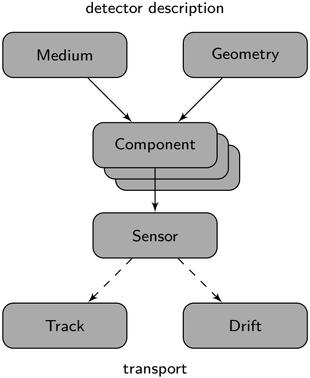
- One of the differences with respect to the Fortran version is that GARFIELD++ also includes the possibility to simulate silicon detectors.
- The source code is hosted on gitlab. Installation instructions can be found on the website and in the user guide. Pre-compiled libraries are available on cvmfs.

material properties

- gases ($\rightarrow$ Magboltz)
- semiconductors

field calculation

- analytic
- field maps
- neBEM



primary ionisation

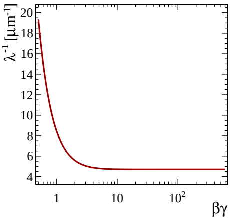
- Heed
- SRIM/TRIM
- Geant4

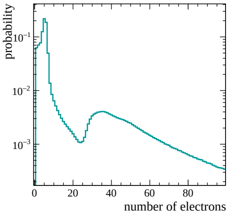
charge transport

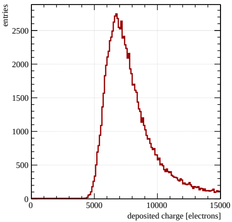
- microscopic
- MC integration
- RKF integration

Primary ionisation

- Energy loss by relativistic charged particles is well described by the PAI model.
- GARFIELD++ includes an interface with HEED (I. Smirnov), which implements an extended version of this model, simulating also atomic relaxation and delta electron transport, such that one obtains the coordinates of all low-energy electrons and holes produced along a track.
- One can also use HEED for simulating X-ray photoabsorption.
- For simulating ion tracks, one can import results calculated using SRIM or TRIM.
- For other projectiles, a possible solution is to interface GEANT4 and GARFIELD++.
  - D. Pfeiffer *et al*, NIM A 935 (2019), 121







Inverse mean free path and cluster size distribution (for $\beta\gamma = 4$) in silicon, calculated using HEED, and simulated charge deposition spectrum (for a 100 μm thick sensor)

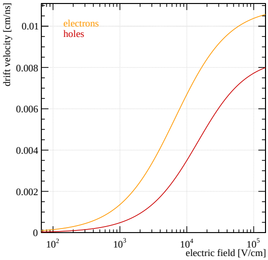
.

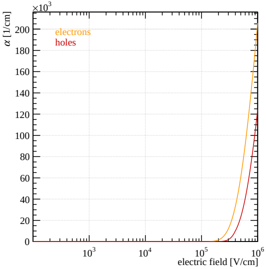
Electric fields

- For simple structures, it is possible to use parameterisations provided by the user.
- For more complex/realistic devices, one typically imports field maps calculated using (Synopsys Sentaurus) TCAD.
- This can be done by probing the electric field/potential in SVisual on a regular grid and exporting the values to a text file, which can then be read by GARFIELD.
- Or (after converting the `.tdr` output file) one can import directly the mesh (`.grd` file) and solution (`.dat` file).
- Can import maps of mobility, lifetimes and other parameters at the same time.

Charge transport

- Typical approach for silicon is to simulate drift lines of individual electrons/holes using a Monte Carlo technique based on macroscopic transport parameters.
- Parameterisations of macroscopic transport properties are based on models found in literature and device simulation programs.
  - Default for drift velocity (Si): Canali high-field mobility model.
  - Default for impact ionisation coefficient (Si): van Overstraeten - de Man model.
  - Other models (and materials other than silicon) are also available or can be implemented upon request.
- Alternatively, one can import a map of transport data from TCAD (*e. g.* attachment coefficient in irradiated devices).





Drift velocity and impact ionisation coefficient in silicon, as function of electric field.

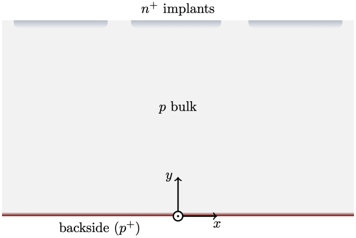
Induced signals

- Given the coordinates of each point along a simulated drift line, the induced current is calculated based on the Shockley-Ramo formalism, using the static weighting potential.
- For calculating the weighting field/potential, the same techniques as for the (drift) electric field can be followed.
  - Analytic expressions for strip and pixel weighting fields are pre-implemented.
- For geometries containing elements with non-zero conductivity, an extension of the Shockley-Ramo theorem is needed (more later).
- The front-end response can be modelled by convoluting the induced current with a transfer function (delta response function).
- The transfer function can be provided as a user-specified function or as a table, or one can use a pre-implemented analytic model, *e. g.* for a unipolar n-stage $CR - RC$ shaper,

$$f(t) = g \exp(n) \left(\frac{t}{t_p}\right)^n \exp(-t/\tau), \qquad t_p = n\tau.$$

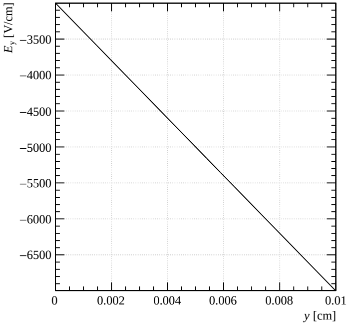
- One can also add noise to the induced current pulse, reproducing a given equivalent-noise charge at the amplifier output.

A simple example (100 µm thick overdepleted *n*-on-*p* sensor)



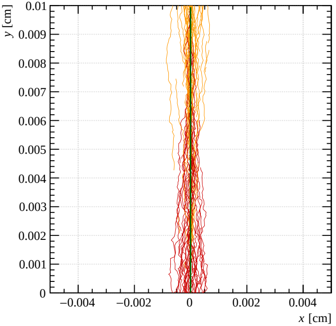
- One-dimensional approximation for the electric field.

```
int main(int argc, char *argv[]) {

  // Define the active medium.
  Garfield::MediumSilicon si;
  si.SetTemperature(293.);

  constexpr double d = 100.e-4;    // Sensor thickness [cm]
  constexpr double vbias = -50.;   // Bias voltage [V]

  // Use a parameterised (linear) drift field.
  auto eLin = [](const double x, const double y, const double z,
                 double& ex, double& ey, double& ez) {
    constexpr double vdep = -20.; // Depletion voltage [V]
    ex = ez = 0.;
    ey = (vbias - vdep) / d + 2 * y * vdep / (d * d);
  };

  Garfield::ComponentUser efield;
  efield.SetElectricField(eLin);
  efield.SetArea(-d, 0., -d, d, d, d);
  efield.SetMedium(&si);

  Garfield::Sensor sensor;
  sensor.AddComponent(&efield);
  // ...
```



- Perpendicularly incident charged particle.

```
//...
Garfield::TrackHeed track;
track.SetSensor(&sensor);
// Set the particle type and momentum [eV/c].
track.SetParticle("pion");
track.SetMomentum(180.e9);

// Simulate electron/hole drift lines using MC integration.
Garfield::AvalancheMC drift;
drift.SetSensor(&sensor);
// Use steps of 1 micron.
drift.SetDistanceSteps(1.e-4);

// Simulate a charged-particle track.
const double xt = 0.;
track.NewTrack(xt, 0, 0, 0, 0, 1, 0);
double xc = 0., yc = 0., zc = 0., tc = 0., ec = 0., extra = 0.;
int ne = 0;
// Retrieve the "clusters" along the track.
while (track.GetCluster(xc, yc, zc, tc, ne, ec, extra)) {
  // Loop over the electrons in the cluster.
  for (int j = 0; j < ne; ++j) {
    double xe = 0., ye = 0., ze = 0., te = 0., ee = 0.;
    double dxe = 0., dye = 0., dze = 0.;
    track.GetElectron(j, xe, ye, ze, te, ee, dxe, dye, dze);
    // Simulate the electron and hole drift lines.
    drift.DriftElectron(xe, ye, ze, te);
    drift.DriftHole(xe, ye, ze, te);
  }
}
```

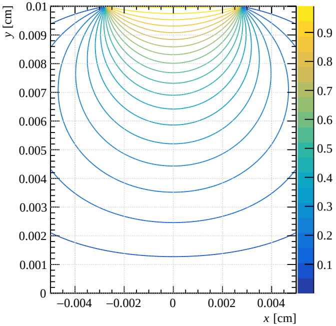


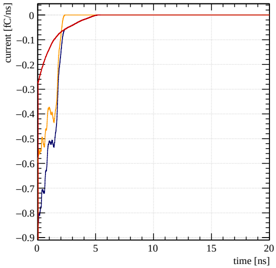
Electron and hole drift lines from a pion track.

Only 1% of the drift lines are shown.

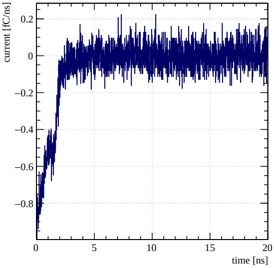
- Pre-implemented analytic expression for the weighting potential of a strip.

```
// ...
constexpr double pitch = 55.e-4; // Strip width [cm]
Garfield::ComponentAnalyticField wfield;
wfield.SetMedium(&si);
wfield.AddPlaneY(0, vbias, "back");
wfield.AddPlaneY(d, 0, "front");
wfield.AddStripOnPlaneY('z', d, -0.5 * pitch, 0.5 * pitch,
                        "strip");
wfield.AddReadout("strip");

Garfield::Sensor sensor;
sensor.AddElectrode(&wfield, "strip");
Garfield::Shaper shaper(3, 2., 1., "unipolar");
sensor.SetTransferFunction(shaper);
// Set the time bins.
const unsigned int nTimeBins = 2000;
const double tstep = 0.01; // ns
sensor.SetTimeWindow(0., tstep, nTimeBins);

// ...
while (track.GetCluster(xc, yc, zc, tc, ne, ec, extra)) {
  // ...
}
// Add noise and convolute with the delta response function.
constexpr double enc = 100.;
sensor.AddWhiteNoise("strip", enc);
sensor.ConvoluteSignals();
```

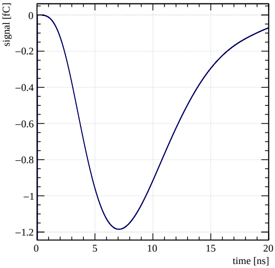




Total induced current and contributions from electrons and holes.



Induced current after adding noise (ENC = 100 e^-).



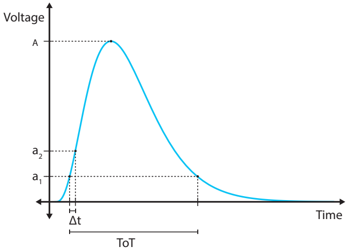
Output signal, after convolution with the front-end transfer function.

- Full source code of the above example is available on gitlab (also as a Python script).
- Step-by-step explanations can be found on the website.

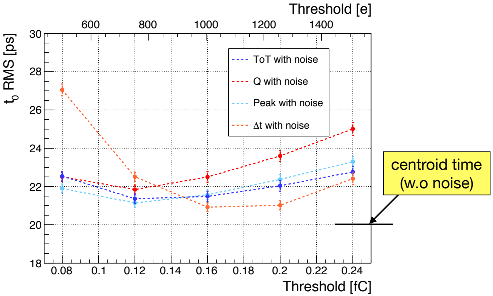
A few applications

Timing studies: planar sensors without gain

- Use a simple sensor model (uniform or linear field) to investigate the impact of charge fluctuations, pixel size, shaper, noise, slewing corrections, *etc.* on the achievable time resolution (large parameter phase space!).
- The example plot below is for a 50 µm thick sensor with $50 \times 50 \mu m^2$ pixels, at 200 V bias, 0.5 ns peaking time, ENC = 100 e^-.
- For more details: presentations by Ann Wang and Marius Mæhlum Halvorsen

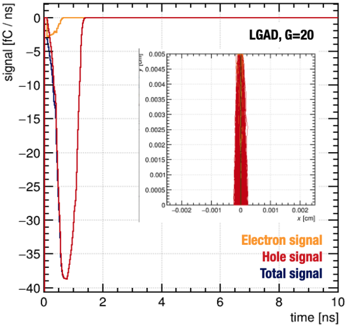


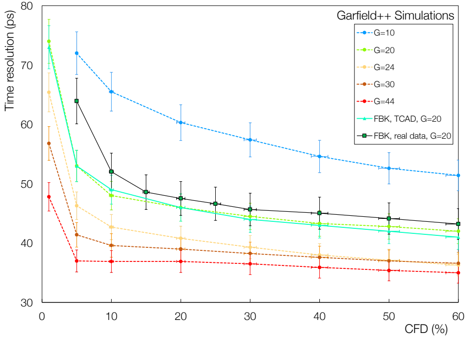
Time slewing corrections.



Timing studies: planar sensors with gain (LGAD)

- Simplified description of the electric field in the bulk and the gain layer.
- The example plot below is for a 50 µm thick pad sensor, ENC = 2000 e^-.



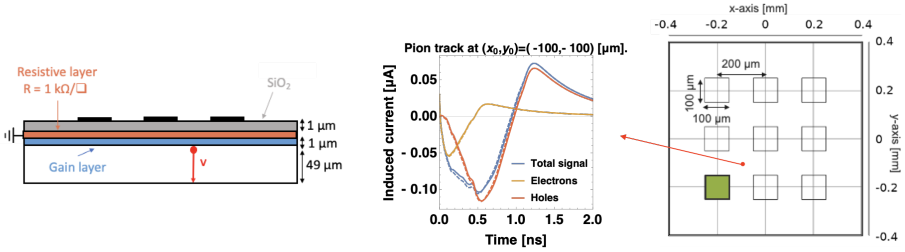


Simulated time resolution for different average gain.

Plots by Francesca Carnesecchi.

Signals in devices with resistive elements

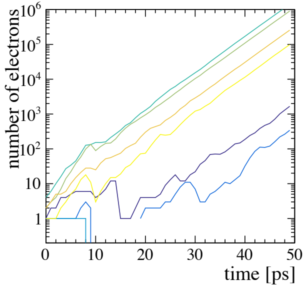
- For calculating the induced signal in geometries including elements with finite conductivity, we need an extension of the Ramo-Shockley theorem.
  - W. Riegler, NIM A 535 (2004), 287
  - W. Riegler, CERN Academic Training, December 2019
- The time-dependent weighting potential required in this formalism can be calculated analytically (for simple geometries), or using a finite-element solver (*e. g.* COMSOL). It can also be calculated in TCAD, using the following recipe.
  - Calculate the stationary solution at nominal bias conditions.
  - Apply a small voltage step ΔV on the electrode to be read out.
  - Run a transient simulation, save a map of the potential at different points in time, and subtract the static potential.
  - Split into a "prompt" component and a "delayed" component.
- Application: AC-LGADs.



Plots by Djunes Janssens. Preliminary/work in progress!

What about SPADs/SiPMs?

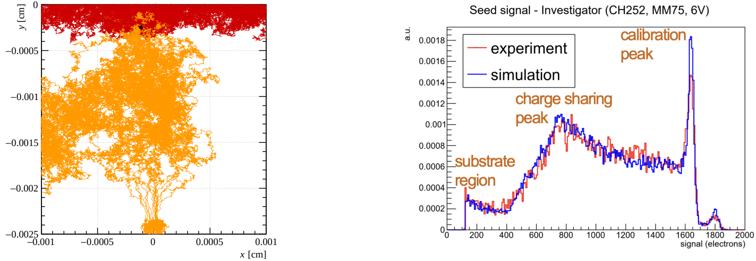
- In principle, the simulation methods discussed so far can also be used for devices operated in breakdown.
- Limitations to this approach arise from (1) space charge and (2) reduction of the electric field due to the drop in bias voltage.
- Some work ahead. . .



Growth of single-electron avalanches in a 1 µm thick multiplication layer with a uniform field of 400 kV / cm.

Monolithic sensors

- For MAPS, electric and weighting fields typically need to be calculated using TCAD.
- The example plot below is from simulations of the ALICE ALPIDE.
- More details: presentation by J. Hasenbichler



Drift lines (left) and charge collection spectrum from a ^{55}Fe source, for an ALPIDE prototype.