Geant 4

Detector Description:
Sensitive Detector & Field

http://cern.ch/geant4
PART III

Detector Sensitivity

- Sensitive detectors
- Primitive scorers
- Hits & digits
- Read-out geometry
Detector sensitivity

- A logical volume becomes sensitive if it has a pointer to a concrete class derived from `G4VSensitiveDetector`
- A sensitive detector either
  - constructs one or more hit objects or
  - accumulates values to existing hits using information given in a `G4Step` object

**NOTE:** you must get the volume information from the “`PreStepPoint`”
A G4VSensitiveDetector object should be assigned to G4LogicalVolume.
In case a step takes place in a logical volume that has a Sensitive Detector object, the Sensitive Detector is invoked with the current G4Step object.
- Either implement dedicated sensitive detector classes, or use predefined scorers.

![Diagram](image-url)
Provided Primitive Scorers

- **Track length**
  - G4PSTrackLength, G4PSPassageTrackLength

- **Deposited energy**
  - G4PSEnergyDepsit, G4PSDoseDeposit, G4PSChargeDeposit

- **Current/Flux**
  - G4PSFlatSurfaceCurrent, G4PSSphereSurfaceCurrent, G4PSPassageCurrent, G4PSFlatSurfaceFlux, G4PSCellFlux, G4PSPassageCellFlux

- **Others:** G4PSMinKinEAtGeneration, G4PSNofSecondary, G4PSNofStep, …

Detector Description: Sensitive Detector & Field - Geant4 Course
Sensitive Detector vs. Primitive Scorer

**Sensitive detector**
- User must implement his/her own detector and hit classes
- One hit class can contain many quantities. A hit can be made for each individual step, or accumulate quantities
- Basically one hits collection is made per one detector
- Hits collection is relatively compact

- Use primitive scorers
  - if **not** interested in recording each individual step **but** accumulating some physics quantities for an event or a run, and
  - if do **not** need too many of them
  - Otherwise... consider implementing your own sensitive detector

**Primitive scorer**
- Many predefined scorers are provided in Geant4. One can add his own
- Each scorer accumulates a quantity for each event
- G4MultiFunctionalDetector creates many collections (maps), i.e. one collection per one scorer
- Keys of maps are redundant for scorers of same volume
Sensitive detector and Hit

- Each “Logical Volume” can have a pointer to a sensitive detector.
- Hit is a snapshot of the physical interaction of a track or an accumulation of interactions of tracks in the sensitive region of your detector.
- A sensitive detector creates hit(s) using the information given in G4Step object. The user has to provide his/her own implementation of the detector response.
- Hit objects, which still are the user’s class objects, are collected in a G4Event object at the end of an event.
  - The UserSteppingAction class should NOT do this.
Hit class – 1

- Hit is a user-defined class derived from `G4VHit`.
- You can store various types of information by implementing your own concrete Hit class.
- For example:
  - Position and time of the step
  - Momentum and energy of the track
  - Energy deposition of the step
  - Geometrical information
  - Or any combination of above
Hit class - 2

- Hit objects of a concrete hit class must be stored in a dedicated collection which is instantiated from `G4THitsCollection` template class.
- The collection will be associated to a `G4Event` object via `G4HCofThisEvent`.
- Hits collections are accessible:
  - through `G4Event` at the end of event,
  - through `G4SDManager` during processing an event.
    - Used for Event filtering.
Readout geometry

- Readout geometry is a virtual and artificial geometry which can be defined in parallel to the real detector geometry
- A readout geometry is optional
- Each one is associated to a sensitive detector
Digitization

- Digit represents a detector output (e.g. ADC/TDC count, trigger signal)
- Digit is created with one or more hits and/or other digits by a concrete implementation derived from G4VDigitizerModule
- In contradiction to the Hit which is generated at tracking time automatically, the digitize() method of each G4VDigitizerModule must be explicitly invoked by the user’s code (e.g. EventAction)
Defining a sensitive detector

- Basic strategy

G4LogicalVolume* myLogCalor = ......;
G4VSensitiveDetector* pSensitivePart =
    new MyCalorimeterSD("/mydet/calorimeter");
G4SDManager* SDMan = G4SDManager::GetSDMpointer();
SDMan->AddNewDetector(pSensitivePart);
myLogCalor->SetSensitiveDetector(pSensitivePart);
PART III

Magnetic Field

- Field Propagation & accuracy
- Global & Local Field
- Tunable parameters
- Field Integration
Field Propagation

- In order to propagate a particle inside a field (e.g. magnetic, electric or both), we integrate the equation of motion of the particle in the field.
- In general this is best done using a Runge-Kutta (RK) method for the integration of ordinary differential equations.
  - Several RK methods are available.
- In specific cases other solvers can also be used:
  - In a uniform field, using the known analytical solution.
  - In a nearly uniform but varying field, with RK+Helix.
Chords

- Once a method is chosen that allows Geant4 to calculate the track's motion in a field, Geant4 breaks up this curved path into linear chord segments.

- The chord segments are determined so that they closely approximate the curved path; they’re chosen so that their **sagitta** is small enough.
  - The **sagitta** is the maximum distance between the curved path and the straight line.
  - Small enough: is smaller than a user-defined maximum.

- Chords are used to interrogate the Navigator.
  - to see whether the track has crossed a volume boundary.
The accuracy of the volume intersection can be tuned by setting a parameter called the "miss distance".

- The miss distance is a measure of the error resolution by which the chord may intersect a volume.
- Default miss distance is 0.25 mm.
- Setting small miss distance may be highly CPU consuming.

One step can consist of more than one chord.
- In some cases, one step consists of several turns.
How to set a Magnetic Field ...

- Magnetic field class
  - Uniform field:
    - `G4UniformMagField` class object
  - Non-uniform field:
    - Concrete class derived from `G4MagneticField`
- Set it to `G4FieldManager` and create a Chord Finder

```cpp
G4FieldManager* fieldMgr =
    G4TransportationManager::GetTransportationManager()
    ->GetFieldManager();
fieldMgr->SetDetectorField(magField);
fieldMgr->CreateChordFinder(magField);
```
Global & Local Fields

- One field manager is associated with the 'world'
- Other volumes/regions in the geometry can override this
  - An alternative field manager can be associated with any logical volume
    - The field must accept position in global coordinates and return field in global coordinates
- The assigned field is propagated to all the daughter volumes

```cpp
G4FieldManager* localFieldMgr = new G4FieldManager(magField);
logVolume->setFieldManager(localFieldMgr, true);
```
where 'true' makes it push the field to all the daughter volumes, unless a daughter has its own field manager.

- It is possible to customise the field propagation classes
  - Choosing an appropriate stepper for the field
  - Setting precision parameters
Tunable Parameters

In addition to the “miss distance” there are two more parameters which can be set in order to adjust the accuracy (and performance) of tracking in a field.

- Such parameters govern the accuracy of the intersection with a volume boundary and the accuracy of the integration of other steps.

The “delta intersection” parameter is the accuracy to which an intersection with a volume boundary is calculated.

- This parameter is especially important because it is used to limit a bias that the algorithm (for boundary crossing in a field) exhibits.
- The intersection point is always on the 'inside' of the curve. By setting a value for this parameter that is much smaller than some acceptable error, one can limit the effect of this bias.
The “delta one step” parameter is the accuracy for the endpoint of 'ordinary' integration steps, those which do not intersect a volume boundary
- It is a limit on the estimation error of the endpoint of each physics step

Parameters “delta intersection” and “delta one step” are strongly coupled
- These values must be reasonably close to each other (within one order of magnitude)

Parameters can be set by:
```cpp
theChordFinder->SetDeltaChord ( miss_distance );
theFieldManager->SetDeltaIntersection ( delta_intersection );
theFieldManager->SetDeltaOneStep ( delta_one_step );
```
Imprecisions ...

- ... are due to approximating the curved path by linear sections (chords)
  - Parameter to limit this is maximum sagitta $\delta_{\text{chord}}$
- ... are due to numerical integration, ‘error’ in final position and momentum
  - Parameters to limit are $\varepsilon_{\text{integration}}$ max, min
- ... are due to intersecting approximate path with the volume boundary
  - Parameter is $\delta_{\text{intersection}}$
Key elements

- Precision of track required by the user relates primarily to:
  - The precision (error in position) \( e_{pos} \) after a particle has undertaken track length \( s \)
  - Precision DE in final energy (momentum) \( \delta_E = \Delta E/E \)
  - Expected maximum number \( N_{\text{int}} \) of integration steps

- Recipe for parameters:
  - Set \( \varepsilon_{\text{integration}} (\text{min, max}) \) smaller than
    - The minimum ratio of \( e_{pos} / s \) along particle’s trajectory
    - \( \delta_E / N_{\text{int}} \) the relative error per integration step (in E/p)
  - Choosing how to set \( \delta_{\text{chord}} \) is less well-defined. One possible choice is driven by the typical size of the geometry (size of smallest volume)
## Where to find the parameters ...

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Name</th>
<th>Class</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_{\text{miss}}$</td>
<td>DeltaChord</td>
<td>G4ChordFinder</td>
<td>0.25 mm</td>
</tr>
<tr>
<td>$d_{\text{min}}$</td>
<td>stepMinimum</td>
<td>G4ChordFinder</td>
<td>0.01 mm</td>
</tr>
<tr>
<td>$\delta_{\text{intersection}}$</td>
<td>DeltaIntersection</td>
<td>G4FieldManager</td>
<td>1 micron</td>
</tr>
<tr>
<td>$\varepsilon_{\text{max}}$</td>
<td>epsilonMax</td>
<td>G4FieldManager</td>
<td>0.001</td>
</tr>
<tr>
<td>$\varepsilon_{\text{min}}$</td>
<td>epsilonMin</td>
<td>G4FieldManager</td>
<td>$5 \times 10^{-5}$</td>
</tr>
<tr>
<td>$\delta_{\text{one step}}$</td>
<td>DeltaOneStep</td>
<td>G4FieldManager</td>
<td>0.01 mm</td>
</tr>
</tbody>
</table>
Volume miss error

- Due to the approximation of the curved path by linear sections (chords)

  - Parameter $\delta_{\text{chord}} = \text{maximum sagitta}$
  - Effect of this parameter as $\delta_{\text{chord}} \rightarrow 0$

$$s_{\text{1step propagator}} \sim (8 \delta_{\text{chord}} R_{\text{curv}})^{1/2}$$

so long as $s_{\text{propagator}} \leq s_{\text{phys}}$ and $s_{\text{propagator}} > d_{\text{min(integr)}}$
Due to error in the numerical integration (of equations of motion)
Parameter(s): $\varepsilon_{\text{integration}}$

- The size $s$ of the step is limited so that the estimated errors of the final position $\Delta r$ and momentum $\Delta p$ are both small enough:
  $$\max\left(\frac{||\Delta r||}{s}, \frac{||\Delta p||}{||p||}\right) < \varepsilon_{\text{integration}}$$

- For Classical RK4 Stepper
  $$s_{1\text{step}}^{\text{integration}} \sim (\varepsilon_{\text{integration}})^{1/3}$$
  for small enough $\varepsilon_{\text{integration}}$

- The integration error should be influenced by the precision of the knowledge of the field (measurement or modeling).
  $$N_{\text{steps}} \sim (\varepsilon_{\text{integration}})^{-1/3}$$

Detector Description: Sensitive Detector & Field - Geant4 Course
Integration error - 2

- $\varepsilon_{\text{integration}}$ is currently represented by 3 parameters
  - $\text{epsilonMin}$, a minimum value (used for big steps)
  - $\text{epsilonMax}$, a maximum value (used for small steps)
  - $\text{DeltaOneStep}$, a distance error (for intermediate steps)

$$\varepsilon_{\text{integration}} = \frac{\delta_{\text{one step}}}{S_{\text{physics}}}$$

- Determining a reasonable value
  - Suggested to be the minimum of the ratio (accuracy/distance) between sensitive components, ...  

- Another parameter
  - $d_{\text{min}}$ is the minimum step of integration

Defaults
- $0.5 \times 10^{-7}$
- $0.05$
- $0.25 \text{ mm}$
- Default $0.01 \text{ mm}$
Intersecting approximate path with volume boundary

- In trial step AB, intersection is found with a volume at C
- Step is broken up, choosing D, so
  \[ S_{AD} = S_{AB} \times \frac{|AC|}{|AB|} \]

- If \(|CD| < \delta_{\text{intersection}}\)
  - Then C is accepted as intersection point.
- So \(\delta_{\text{int}}\) is a position error/bias
\( \delta_{\text{int}} \) must be small
- compared to tracker hit error
- its effect on reconstructed momentum estimates should be calculated
  - ... and limited to be acceptable

Cost of small \( \delta_{\text{int}} \) is less
- than making \( \delta_{\text{chord}} \) small
- it is proportional to the number of boundary crossings – not steps

Quicker convergence / lower cost
- Possible with optimization

If C is rejected, a new intersection point E is found. E is good enough
- if \(|EF| < \delta_{\text{int}}\)
Customizing field integration

- **Runge-Kutta** integration is used to compute the motion of a charged track in a general field. There are many general steppers from which to choose:
  - Low and high order, and specialized steppers for pure magnetic fields
- By default, Geant4 uses the classical fourth-order **Runge-Kutta** stepper (**G4ClassicalRK4**), which is general purpose and robust.
  - If the field is known to have specific properties, lower or higher order steppers can be used to obtain the results of same quality using fewer computing cycles
- If the field is calculated from a field map, a lower order stepper is recommended:
  - The less smooth the field is, the lower the order of the stepper that should be used
  - The choice of lower order steppers includes the third order stepper (**G4SimpleHeum**), the second order (**G4ImplicitEuler** and **G4SimpleRunge**), and the first order (**G4ExplicitEuler**)
    - A first order stepper would be useful only for very rough fields
    - For somewhat smooth fields (intermediate), the choice between second and third order steppers should be made by trial and error
Trying a few different types of steppers for a particular field or application is suggested if maximum performance is a goal.

Specialized steppers for pure magnetic fields are also available:
- They take into account the fact that a local trajectory in a slowly varying field will not vary significantly from a helix.
- Combining this in with a variation, the Runge-Kutta method can provide higher accuracy at lower computational cost when large steps are possible.

To change the stepper:

```
theChordFinder
    ->GetIntegrationDriver();
    ->RenewStepperAndAdjust( newStepper );
```
Other types of field

- It is possible to create any specialised type of field:
  - inheriting from `G4VField`
  - Associating an *Equation of Motion* class (inheriting from `G4EqRhs`) to simulate other types of fields
  - Fields can be time-dependent

- For pure electric field:
  - `G4ElectricField` and `G4UniformElectricField` classes

- For combined electromagnetic field:
  - `G4ElectroMagneticField` class

- The *Equation of Motion* class for electromagnetic field is `G4MagElectricField`.

```cpp
G4ElectricField* fEMfield = new G4UniformElectricField( G4ThreeVector(0., 100000.*kilovolt/cm, 0.) );
G4EqMagElectricField* fEquation = new G4EqMagElectricField(fEMfield);
G4MagIntegratorStepper* fStepper = new G4ClassicalRK4( fEquation, nvar );
G4FieldManager* fFieldMgr = G4TransportationManager::GetTransportationManager()->GetFieldManager();
fFieldManager->SetDetectorField( fEMfield );
G4MagInt_Driver* fIntgrDriver = new G4MagInt_Driver( fMinStep, fStepper, fStepper->GetNumberOfVariables() );
G4ChordFinder* fChordFinder = new G4ChordFinder(fIntgrDriver);
```