Geometry: Magnetic Field

I. Hrivnacova, IJCLab Orsay

Credits: J.Apostolakis, T. Nikitina, G.Cosmo, A. Lechner (CERN), M. Asai (SLAC) and others

Geant4 ED PHENIICS and IN2P3 Tutorial,
12 - 16 May 2025, IJCLab

Outline

- Defining magnetic field
  - Global, local magnetic fields
  - G4FieldBuilder
- Integration of trajectories in field
  - Tunable parameters of propagation in magnetic field
  - G4FieldParameters
  - UI commands
- Other types of field

Describe Your Detector

- To describe your detector you have to derive your own concrete class from G4VUserDetectorConstruction abstract base class.
- Implement the virtual method Construct(), where you
  - Instantiate all necessary materials
  - Instantiate volumes of your detector geometry
- Optionally, implement the virtual method ConstructSDandField(), where you
  - Instantiate your sensitive detector classes and set them to the corresponding logical volumes
  - Instantiate magnetic (or other) field
- Optionally you can define
  - Regions for any part of your detector
  - Visualization attributes (color, visibility, etc.) of your detector elements

Define a Magnetic field

- To define a (magnetic) field you must create a G4MagneticField object in the ConstructSDandField() method of your detector construction class
- A **uniform magnetic field:**
  - Use an object of G4UniformMagField, already available in Geant4
- **Non-uniform field** :
  - You have to define your ‘concrete’ class derived from G4MagneticField
  - And implement its GetFieldValue method
- Then let G4FieldBuilder create all Geant4 objects for field propagation

Example of Magnetic Field Class

MagneticField.hh

```
#include "G4MagneticField.hh"
#include "G4ThreeVector.hh"
...
class MagneticField : public G4MagneticField {
public:
  MagneticField();
  ~MagneticField() override;

  // Method from the base class
  void GetFieldValue(
    const G4double point[4], G4double* bfield) const override;

  // Set the field value
  void SetFieldValue(G4ThreeVector fieldValue);

private:
  // the field B value vector
  G4ThreeVector fFieldValue;
};
```

`point[0..2]` - position
(in the global coordinate system)
`point[3]` - time
`bfield[0..2]` - returned field value
in the given position

G4FieldBuilder

- Helper class for magnetic or other field construction – since Geant4 11.3
- It is created just once (on master) in your detector construction constructor
  - The instance of the field builder creates a G4FieldParameters object and the UI commands for customisation of the field default configuration

DetectorConstruction.cc

```
#include "G4FieldBuilder.hh"

DetectorConstruction::DetectorConstruction()
{
  // Create field builder
  G4FieldBuilder::Instance();
  // G4FieldBuilder::Instance()->SetVerboseLevel(2);
}
```

Field Construction

- Create your field in DetectorConstruction::ConstructSDandField()
- Set it to G4FieldBuilder and let it create all Geant4 objects for field propagation

DetectorConstruction.cc

```
void DetectorConstruction::ConstructSDandField()
{
  // Create user detector field
  auto magField = new G4UniformMagField(fFieldVector);

  // Set field to the field builder
  auto fieldBuilder = G4FieldBuilder::Instance();
  fieldBuilder->SetGlobalField(magField);

  // Construct all Geant4 field objects
  fieldBuilder->ConstructFieldSetup();
}
```

Local Field Construction

- It is possible to create a field for a part of the detector, local field
- A local field is set to G4FieldBuilder with the associated logical volume (`fRadiatorLV` in the example below)

DetectorConstruction.cc

```
void DetectorConstruction::ConstructSDandField()
{
  // Create user detector field
  auto localMagField = new MagneticField();

  // Set field to the field builder
  auto fieldBuilder = G4FieldBuilder::Instance();
  fieldBuilder->SetLocalField(localMagField, fRadiatorLV);

  // Construct all Geant4 field objects
  fieldBuilder->ConstructFieldSetup();
}
```

logical volume where `localMagField` will be applied

Global and Local Fields

- The global and local magnetic field can coexist, the global field is overriden in the volumes associated with a local field

DetectorConstruction.cc

```
void DetectorConstruction::ConstructSDandField()
{
  // Create user detector fields
  auto globalMagField = new G4UniformMagField(fFieldVector);
  auto localMagField = new MagneticField();

  // Set fields to the field builder
  auto fieldBuilder = G4FieldBuilder::Instance();
  fieldBuilder->SetGlobalField(globalMagField);
  fieldBuilder->SetLocalField(localMagField, fRadiatorLV);

  // Construct all Geant4 field objects
  fieldBuilder->ConstructFieldSetup();
}
```

logical volume where `localMagField` will be applied

Global Field Messenger

- Another helper class, G4GlobalMagFieldMessenger, is available for **uniform global magnetic field**
  - **The field** is **activated** only when its fieldValue is non zero vector.
  - It can be also used to change the field value (and activate or inactivate the field again

DetectorConstruction.cc

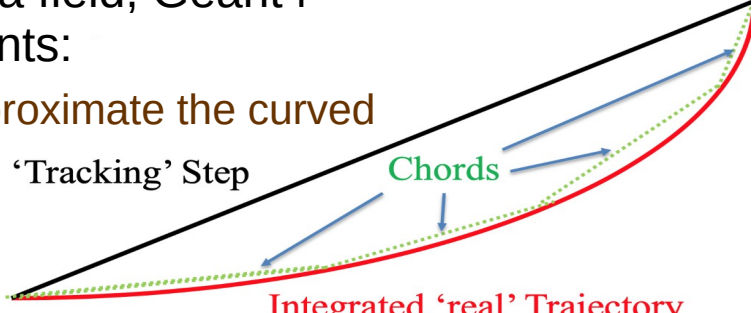
```
void DetectorConstruction::CreateSDandField()
{
   // Global magnetic field & its messenger
   G4ThreeVector fieldValue = G4ThreeVector();
   G4GlobalMagFieldMessenger* magFieldMessenger
      = new G4GlobalMagFieldMessenger(fieldValue);

   // Register the messenger for deleting
   G4AutoDelete::Register(myFieldMessenger);
}
```

See basic examples B2 and B4

Propagation in Field
Tunable Parameters

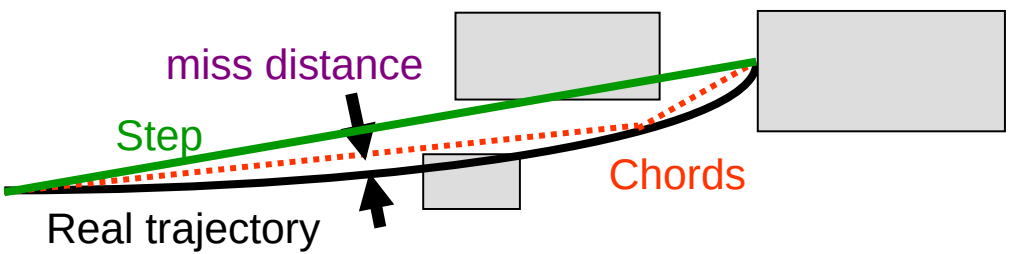
Propagation in Field

- To propagate a particle inside a field (e.g. magnetic, electric or both), we solve *the equation of motion* of the particle in the field
  - In general, this is done using a Runge-Kutta method for the integration of ordinary differential equations
  - For specific cases where an analytical solution is known, it is possible to utilize this instead
- Several other **Runge-Kutta 'steppers'** and other integration methods are available, suitable for different conditions.
  - The established 4th/5th order RK 'Dormand Prince' is default
- Using the method to calculate the track's motion in a field, Geant4 breaks up this curved path into linear chord segments:
  - Chord segments are chosen so that they closely approximate the curved path



Tracking in Field

- We use the chords to interrogate the G4Navigator, to see whether the track has crossed a volume boundary. One physics/tracking step can create several chords.
  - In some cases, one step consists of several helix turns.
- User can set the accuracy of the volume intersection by setting the following parameters:
  - miss distance: the maximum allowed distance between curve and the approximation chord (sagitta)
  - delta intersection: the accuracy to which an intersection with a volume boundary is calculated
  - delta one step: the accuracy for the endpoint of 'ordinary' integration steps, those which do not intersect a volume boundary.
  - The last two parameters are strongly coupled, their values must be reasonably close to each other (at most within one order of magnitude)
- For more look in the Electromagnetic Field section of the Application Developers Guide



G4FieldParameters

- The G4FieldParameters class defines the field configuration:
  - The field type, the type of equation of motion of a particle in a field and the integration method, as well as other accuracy parameters.
  - The default values correspond to the defaults set in Geant4 field classes
- It provides the UI commands for customisation of the field default configuration
- Includes new enumerations G4FieldType, G4EquationType, G4StepperType and static functions for conversion between enum value and a meaningful name
  - These names are then used in the messenger for building UI commands and the list of commands candidates
- It is created with the G4FieldBuilder instance

Local Field(s)

- To create a new set of parameters (configuration) for a local field, users should call the G4FieldBuilder function CreateFieldParameters with the local field root logical volume name as parameter
  - New commands will be located in the `/field/volumeName` directory

DetectorConstruction.cc

```
#include "G4FieldBuilder.hh"

DetectorConstruction::DetectorConstruction()
{
  // Create field builder
  // this will create commands for global field configuration
  G4FieldBuilder::Instance();

  // Create new configuration for a local field in
  // logical volume  "Radiator"
  G4FieldBuilder::Instance()->CreateFieldParameters("Radiator");
}
```

UI Commands

- After the G4FieldBuilder instatiation, the following commands are available to users:

```
/field/fieldType fieldType
        fieldType = Magnetic | ElectroMagnetic | Gravity
```

```
/field/equationType eqType
        eqType = EqMagnetic | EqMagneticWithSpin | EqElectroMagnetic  |
                 EqEMfieldWithSpin  | EqEMfieldWithEDM
```

UI Commands -2

- The commands for stepper type selection:

```
/field/stepperType stepperType
         stepperType = CashKarpRKF45 | ClassicalRK4 | ExplicitEuler |
               ImplicitEuler | SimpleHeum | SimpleRunge | ConstRK4 |
               ExactHelixStepper | HelixExplicitEuler | HelixHeum |
               HelixImplicitEuler | HelixMixedStepper | HelixSimpleRunge |
               NystromRK4 | RKG3Stepper
```

UI Commands - 3

- The UI commands to set parameters used in field propagation and general commands

```
/field/setMinimumStep value
/field/setDeltaChord  value
/field/setDeltaOneStep value
/field/setDeltaIntersection value
/field/setMinimumEpsilonStep value
/field/setMaximumEpsilonStep value
/field/setConstDistance value
# general commands
/field/update
/field/printParameters
/field/verboseLevel value
```

Other types of field

- The user can create their own type of field
  - inheriting from G4VField,
  - using an associated **Equation of Motion** class (inheriting from G4EqRhs) to simulate other types of fields.
  - fields be time-dependent.
- For a few cases Geant4 has an existing class:
  - pure electric field, Geant4 has G4ElectricField (and G4UniformElectricField)
  - combined electromagnetic field, the G4ElectroMagneticField class
- A different Equation of Motion class is used for electromagnetic
- For the full exercise of the options for fields you can browse `examples/extended/field/`
  - E.g. field02 demonstrates electric field

BACKUP

Old slides with explicit field configuration (without G4FieldBuilder)

How to assign a field to the whole detector

- The magnetic field is applied to geometry with means of G4FieldManager
- A **global field manager** is associated with the 'world' volume
  - It is created by G4TransportationManager already before user detector construction is called
- To associate your field with the world, you must obtain that global field manager:

DetectorConstruction.cc

```
auto globalFieldManager
       = G4TransportationManager::GetTransportationManager()
         ->GetFieldManager();
```

- And then set it in that field manager:

```
globalFieldManager->SetDetectorField(magField);
```

Global and local fields

- Other volumes can override this global field
- An alternative field manager can be associated with any logical volume, it handles then the **local field**
  - By default this is propagated to all its daughter volumes
  - The field must accept **position in global coordinates** and return **field in global coordinates**

DetectorConstruction.cc

```
auto fieldManager = new G4FieldManager(magField);
logVolume->SetFieldManager(fieldManager, true);
```

  - Where ‘true’ means to propagate field to all the volumes it contains

Global Magnetic Field

DetectorConstruction.cc

```
void DetectorConstruction::CreateSDandField()
{
   // Magnetic field
   auto myField = new MyMagneticField();

   // Field manager
   auto fieldManager
      = G4TransportationManager::GetTransportationManager()
        ->GetFieldManager();
   fieldManager->SetDetectorField(myField);
   fieldManager->CreateChordFinder(myField);

}
```

Local Magnetic Field

DetectorConstruction.cc

```
void DetectorConstruction::CreateSDandField()
{
   // Magnetic field
   auto myField = new MyMagneticField();

   // Field manager
   auto fieldManager = new G4FieldManager();
   fieldManager->SetDetectorField(myField);
   fieldManager->CreateChordFinder(myField);

   // Set field to a logical volume
   G4bool forceToAllDaughters = true;
   magneticLogical
     ->SetFieldManager(fieldManager, forceToAllDaughters);
}
```

See also basic example B5