



GEANT4
A SIMULATION TOOLKIT



Geometry: Magnetic Field

I. Hrivnacova, IJCLab Orsay

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

Geant4 IN2P3 and ED PHENIICS Tutorial,
22 – 26 May 2023, IJCLab

Outline

- Defining magnetic field
- Integration of trajectories in field
 - Tunable parameters of propagation in magnetic field
- 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

How to define a Magnetic field

- To create a (magnetic) field you must instantiate a `G4MagneticField` object in the `ConstructSDandField()` method of your DetectorConstruction class
- To define a **uniform magnetic field** use an object of `G4UniformMagField`:

```
auto magField
    = new G4UniformMagField(G4ThreeVector(0, 0, 1.*tesla));
```

- Non-uniform field : deriving your 'concrete' class:
 - Define your own concrete class `MyField` derived from `G4MagneticField` and implement `GetFieldValue` method:

```
void MyField::GetFieldValue(
    const double point[4], double* field) const;
```

- where `point[0..2]` represents the position in the global coordinate system and `point[3]` time
- `field[0..2]` return the field value in the given position

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:

```
G4FieldManager* 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**

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

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

Global Magnetic Field

```
void MyDetectorConstruction::CreateSDandField()
{
    // Magnetic field
    MyMagneticField* myField = new MyMagneticField();

    // Field manager
    G4FieldManager* fieldManager
        = G4TransportationManager::GetTransportationManager()
        ->GetFieldManager();
    fieldManager->SetDetectorField(myField);
    fieldManager->CreateChordFinder(myField);
}
```

Local Magnetic Field

```
void MyDetectorConstruction::CreateSDandField()
{
    // Magnetic field
    MyMagneticField* myField = new MyMagneticField();

    // Field manager
    G4Fieldmanager* 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

Global Field Messenger

- A helper class, `G4GlobalMagFieldMessenger`, is available since Geant4 10.00
 - It creates **the global uniform magnetic field**
 - **The field is activated** (set to the `G4TransportationManager` object) 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)

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

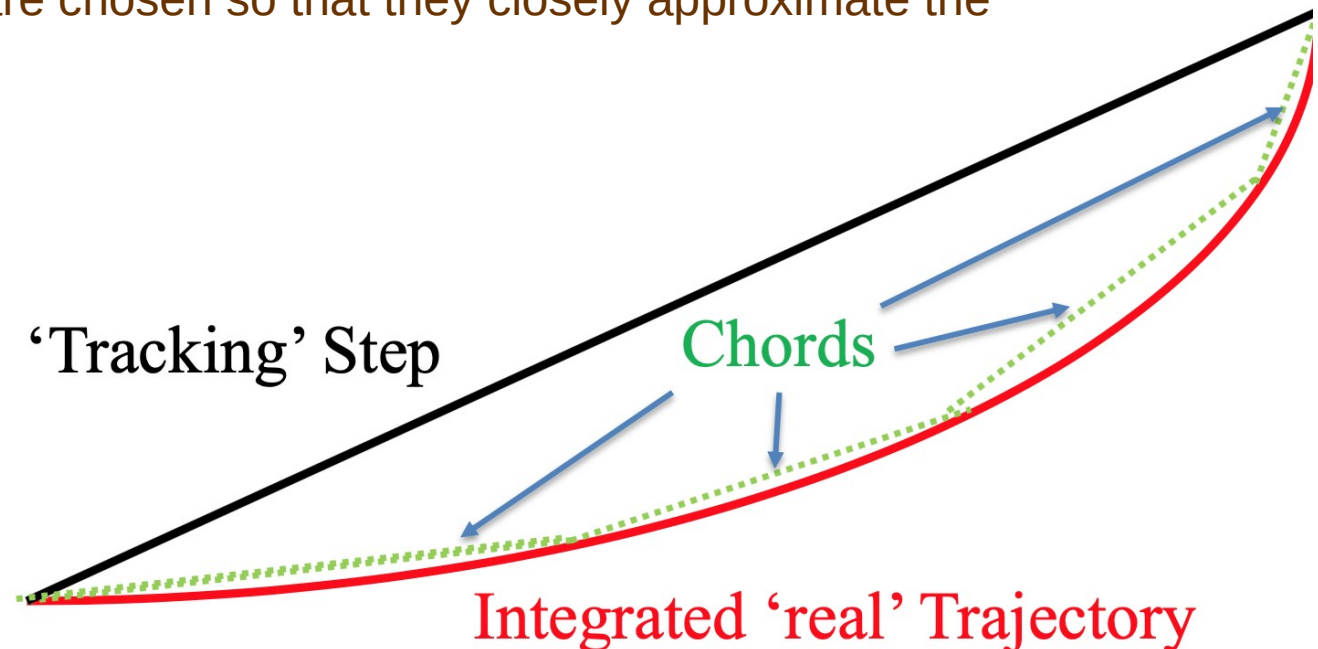
    // Register the messenger for deleting
    G4AutoDelete::Register(myFieldMessenger);
}
```

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
- By default Geant4 uses a **Runge-Kutta method** to integrate the ordinary differential equations of motion
- 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

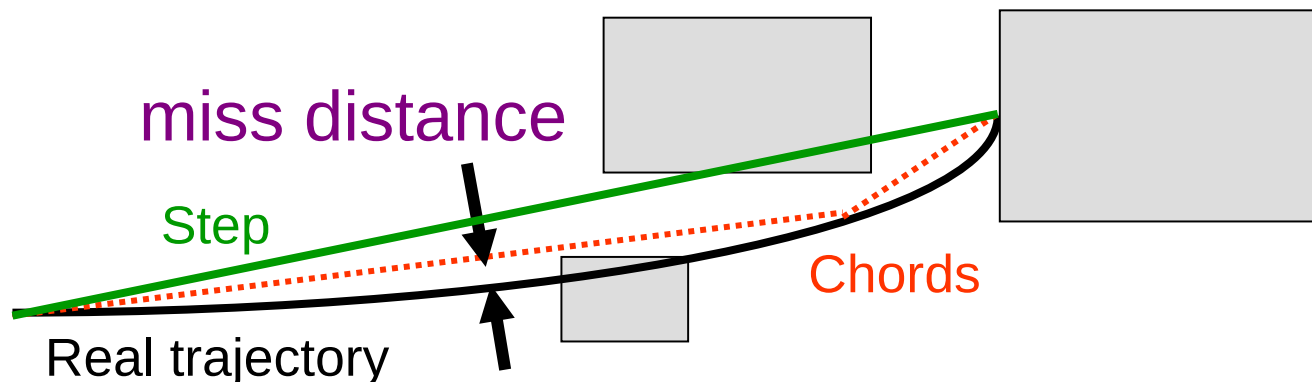


Methods of Integration

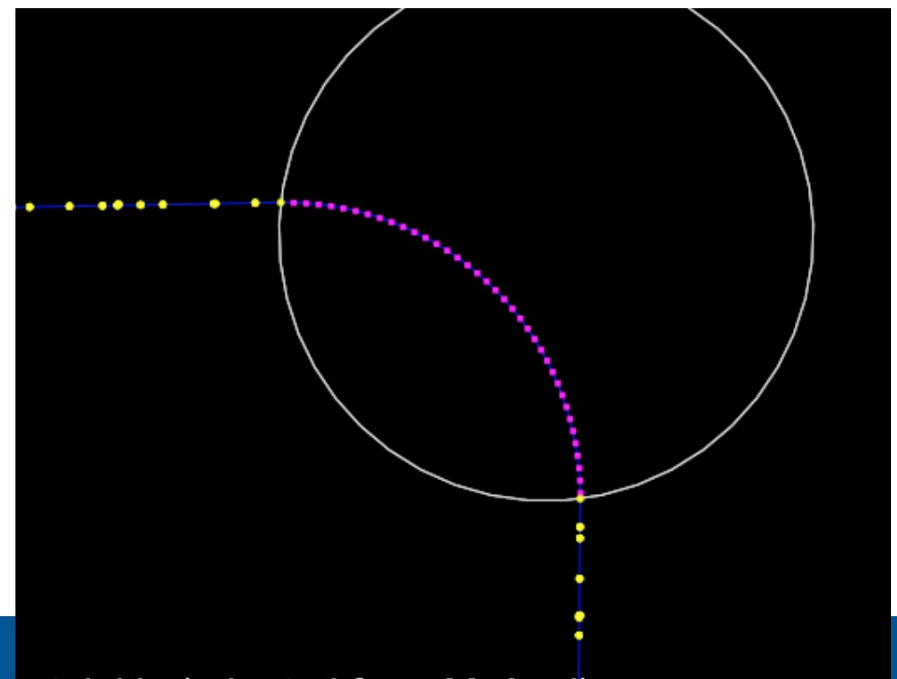
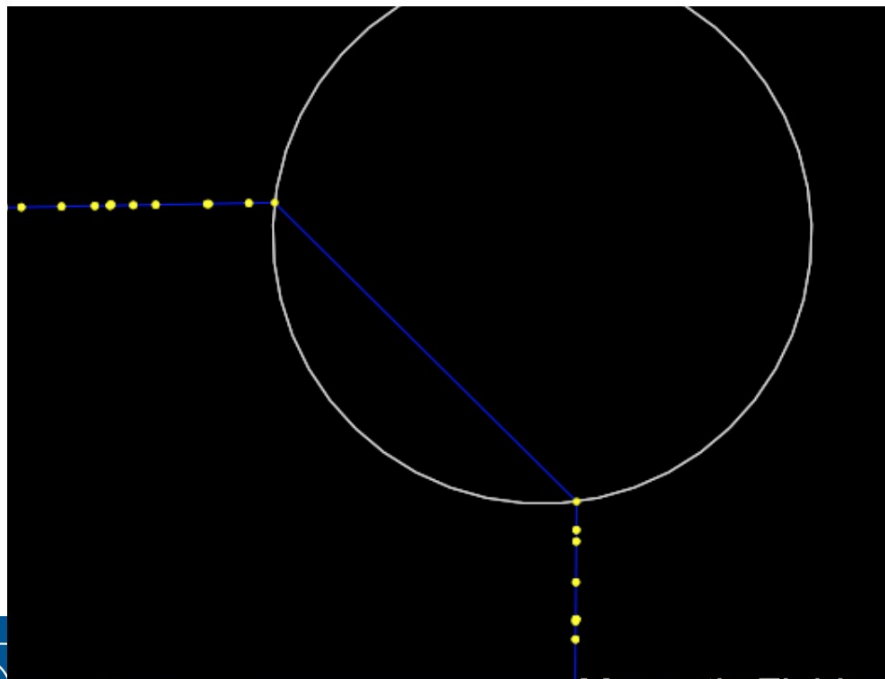
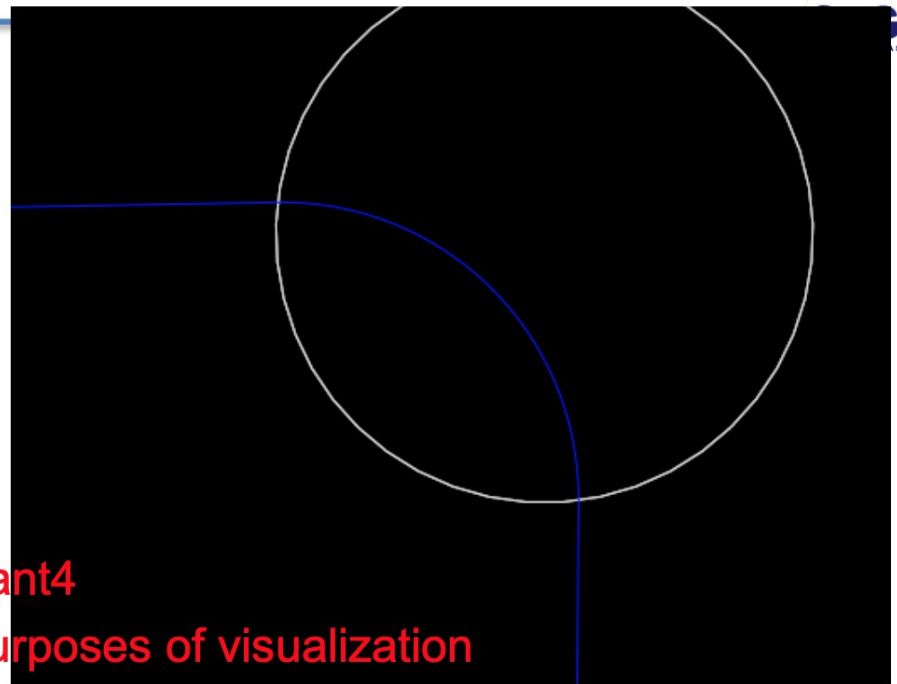
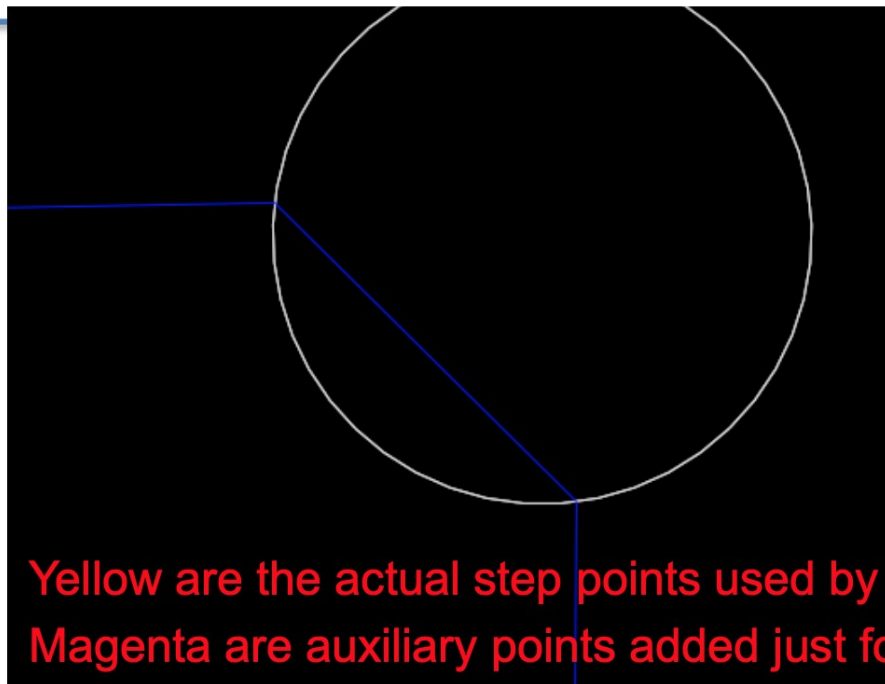
- Several other Runge-Kutta ‘steppers’ and other integration methods are available.
 - The established 4th/5th order RK ‘**Dormand Prince**’ is default
- In specific cases other solvers can also be used:
 - In a uniform field, using a ‘**helix**’ – the analytical solution.
 - In a slowly varying, smooth field, methods that **combine helix & RK**
 - High efficiency RK solvers provided in recent releases (‘**FSAL**’, RK steppers with Interpolation)

Tracking in Field

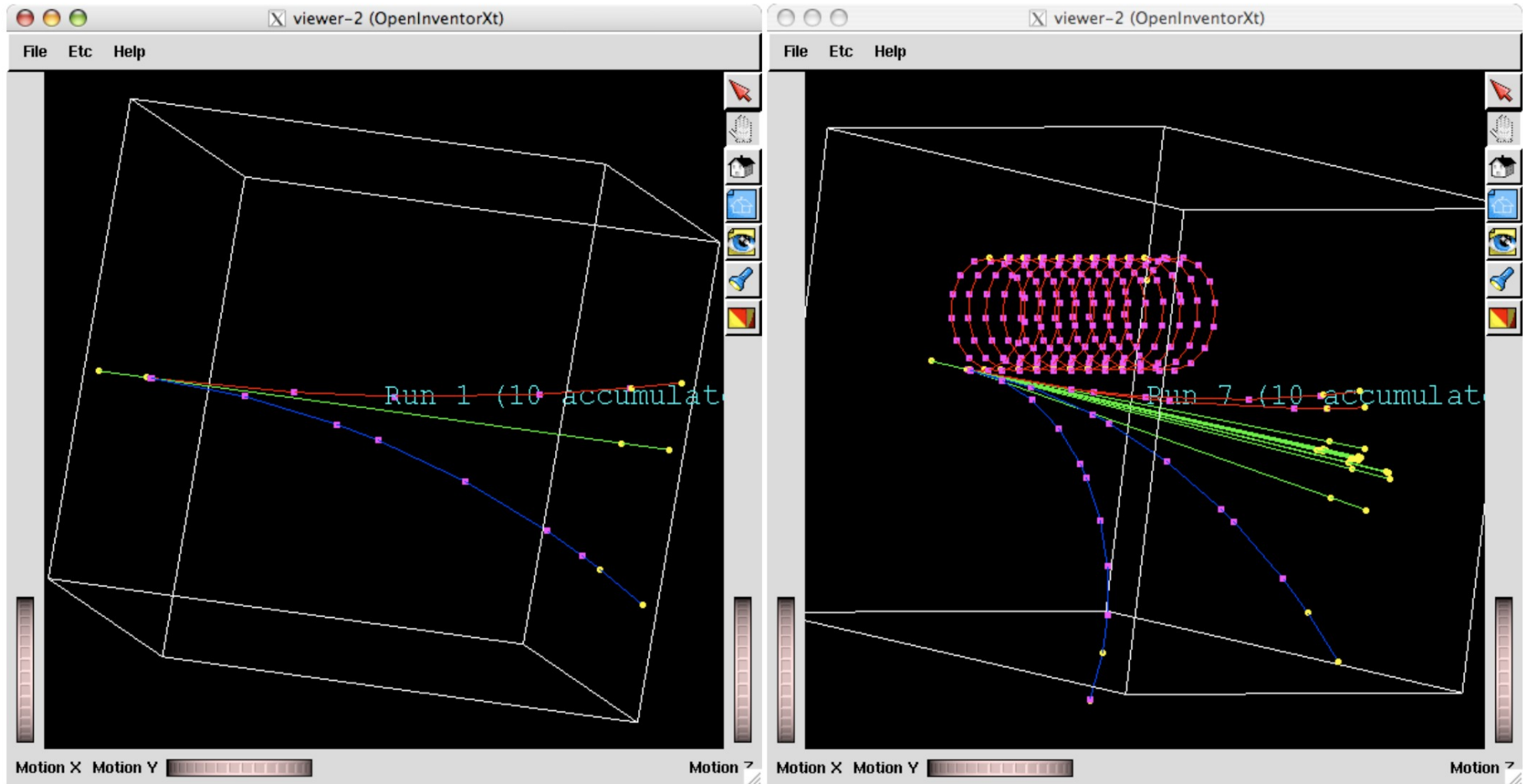
- We use the chords to interrogate the G4Navigator, to see whether the track has crossed a volume boundary.
- One physics/tracking step can create several chords.
 - In some cases, one step consists of several helix turns.
- User can set the accuracy of the volume intersection,
 - By setting a parameter called the “miss distance”
 - The curved trajectory will be approximated by chords, so that the maximum estimated distance between curve and chord (sagitta) is less than the miss distance.
 - It is a measure of the error in whether the approximate track intersects a volume
 - It is quite expensive in CPU performance to set too small “miss distance”.



Regular versus Smooth Trajectory



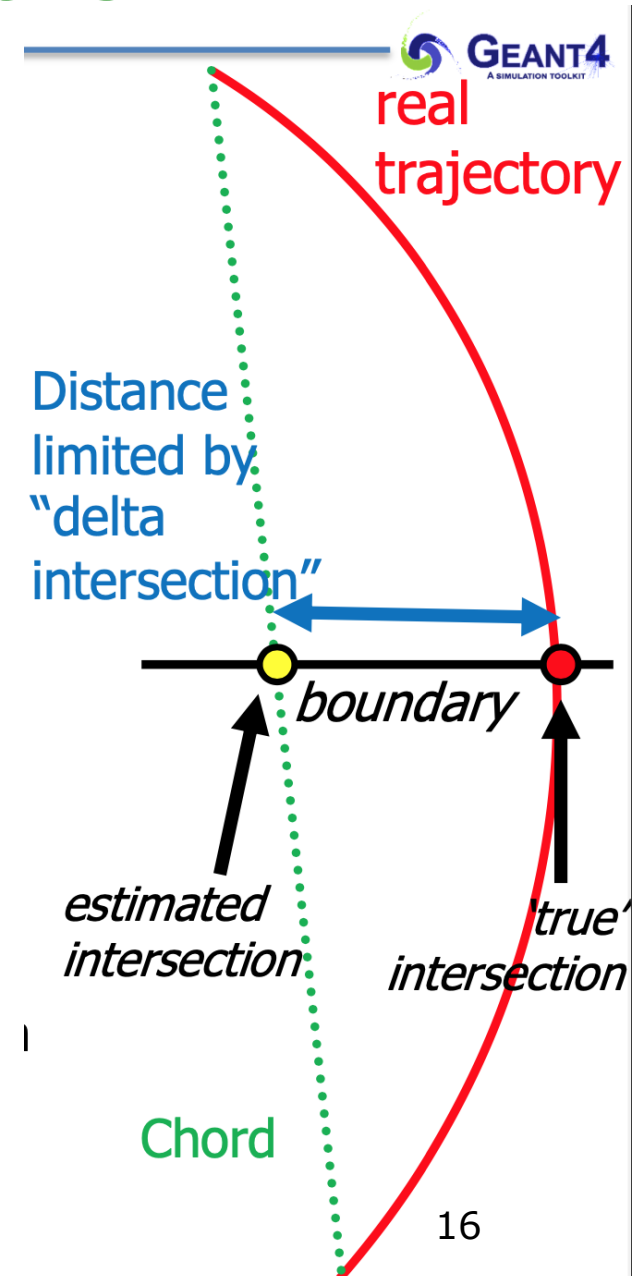
Smooth Trajectory Makes Big Difference for Trajectories that Loop in a Magnetic Field



- Yellow dots are the actual step points used by Geant4
- Magenta dots are auxiliary points added just for purposes of visualization

Tunable Parameters

- In addition to the “miss distance” there are two more parameters which the user can set in order to adjust the accuracy (and performance) of tracking in a field.
 - These parameters govern the accuracy of the intersection with a volume boundary (“delta intersection”)
 - and the accuracy of the integration of other steps (“delta one step”)
- The “delta intersection” parameter is the accuracy to which an intersection with a volume boundary is calculated.
 - If a candidate boundary intersection is estimated to have a precision better than this, it is accepted.
 - This parameter is especially important because it is used to limit a bias that our algorithm (for boundary crossing in a field) exhibits.
 - By setting a value for this parameter that is much smaller than some acceptable error, the user can limit the effect of this bias.



Tunable Parameters - 2

- The “**delta one step**” parameter is the accuracy for the endpoint of 'ordinary' integration steps, those which do not intersect a volume boundary.
 - This parameter limits the estimated relative error of the endpoint of each physics step.
- “**delta intersection**” and “**delta one step**” are strongly coupled. These 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

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. `field01` uses alternative integration methods (see file `src/F01FieldSetup.cc`)
 - `field02` demonstrates electric field