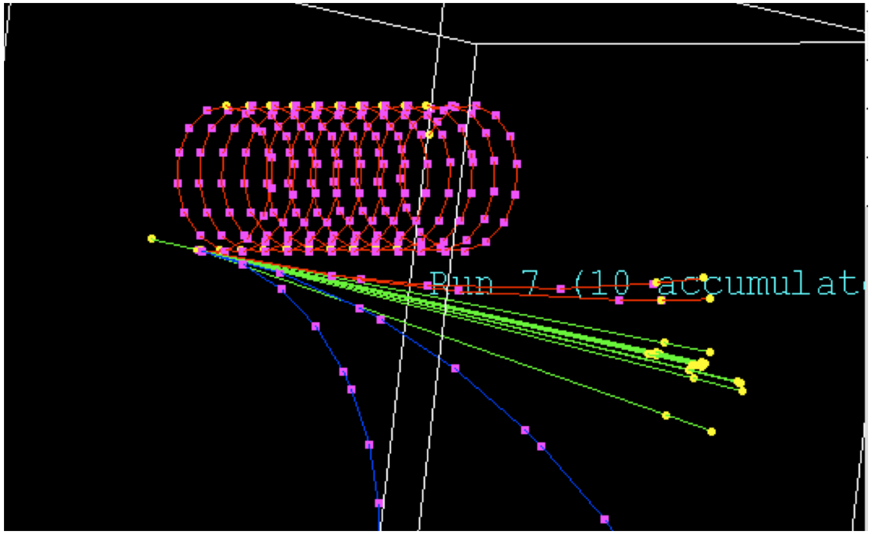
Version 10.5

Magnetic Fields

John Apostolakis (CERN)
Geant4 Beginners Course



Overview

- Magnetic field
- Integration of trajectories in field
- Other types of field

Slides adapted from M. Asai (SLAC)

Version 10.5

Defining a magnetic field

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
  - Uniform field : Use an object of the *G4UniformMagField* class

```
G4MagneticField* magField =
  new G4UniformMagField(G4ThreeVector(1.*Tesla,0.,0.);
```

  - Non-uniform field : Create your own concrete class derived from *G4MagneticField* and implement the **GetFieldValue** method.

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

    - Point[0..2] are x,y,z position in global coordinates, Point[3] is time
    - field[0..2] are output x,y,z components of magnetic field (in G4 units)

How to assign a field to the whole detector

- A global field manager is associated with the ‘world’ volume
  - it already exists, before G4VUserDetectorConstruction is called,
  - it is created / set in G4TransportationManager.
- To associate your field with the world, you must obtain that global field manager:

```
G4Fieldmanager* globalFieldMgr = G4TransportationManager::
    GetTransportationManager()-> GetFieldManager();
```

- And then set it in that field manager:

```
globalFieldMgr->SetDetectorField(field);
```

- Hands-on: look at the ConstructSDandField() method’s code in the Detector Construction method(s) of basic examples B2/B2b and B5.

Global and local fields

- Other volumes 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
  - By default this is propagated to all its daughter volumes

```
G4FieldManager* localFieldMgr
     = new G4FieldManager(magField);
logVolume->setFieldManager(localFieldMgr, true);
```

where '**true**' makes it push the field to all the volumes it contains, unless a daughter has its own field manager.

- Customizing the field propagation classes
  - Choosing an appropriate stepper for your field
  - Setting precision parameters

Magnetic field (2)

```
MyField* myMagneticField = new MyField();
G4Fieldmanager* fieldMgr = new G4FieldManager();
fieldMgr->SetDetectorField(myMagneticField);
fieldMgr->CreateChordFinder(myMagneticField); // Default
   parameters

G4bool forceToAllContained = true; // Propagate to all
fMagneticLogical->SetFieldManager(fieldMgr,
                                   forceToAllContained);
// Register the field and its manager for deletion
G4AutoDelete::Register(myMagneticField);
G4AutoDelete::Register(fieldMgr);
```

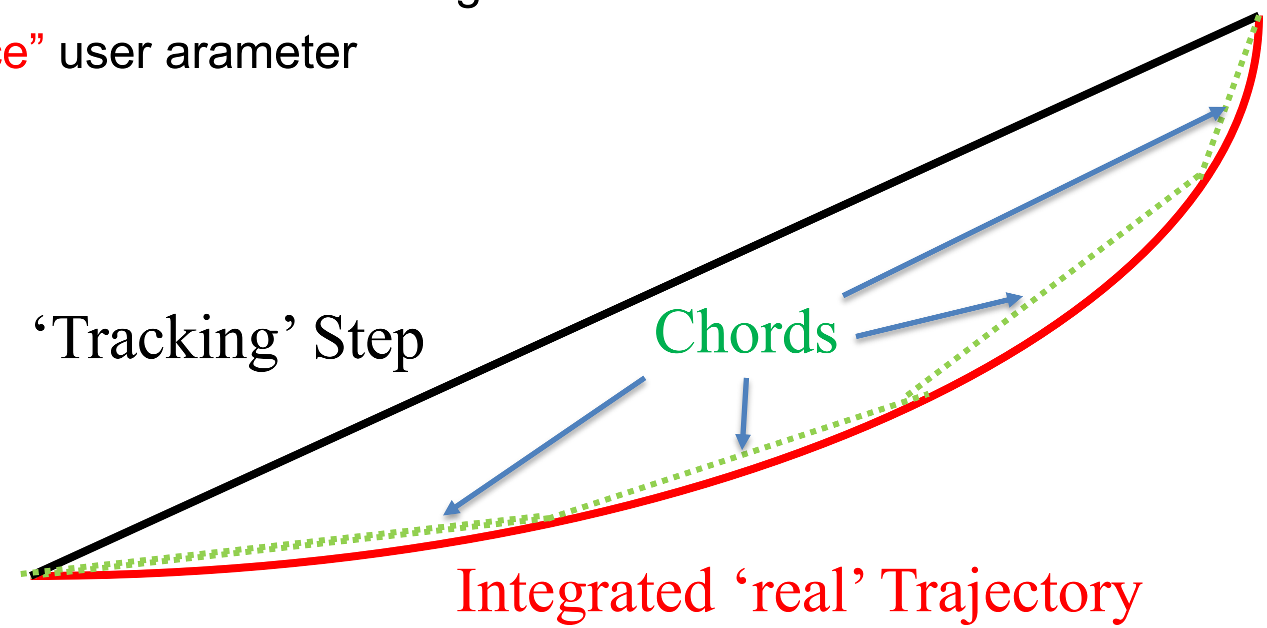
- /example/basic/B5 is a good starting point

Version 10.5

Integration of the trajectory of motion

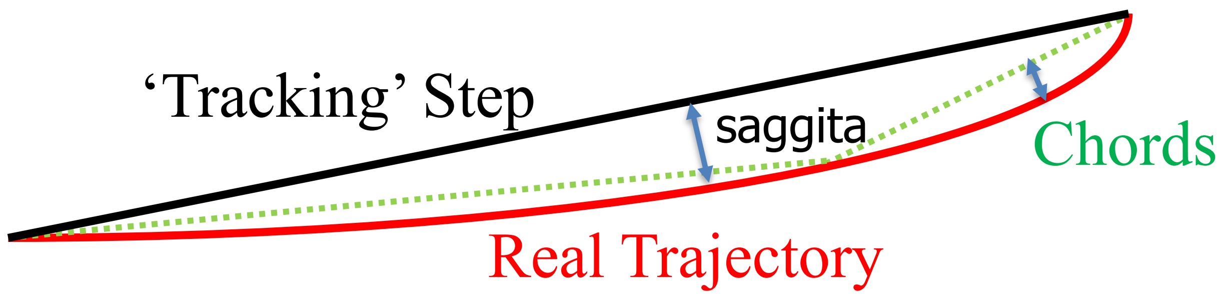
Integration of motion in field

- In order 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 G4 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 chosen so that they closely approximate the curved path.
  – Chords are chosen so that their sagitta is smaller than the value of the "miss distance" user arameter



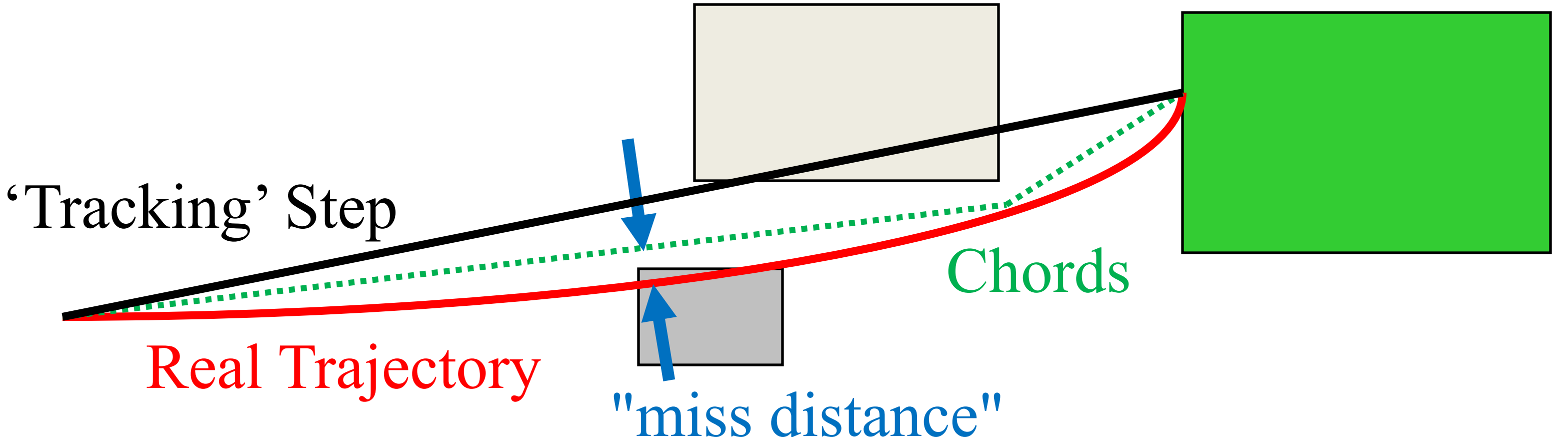
Methods of integration

- Several other Runge-Kutta ‘steppers’ and other integration methods are available.
  - The established 4th/5th order RK ‘Dormand Prince’ is now default (G4 10.4)
- 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)

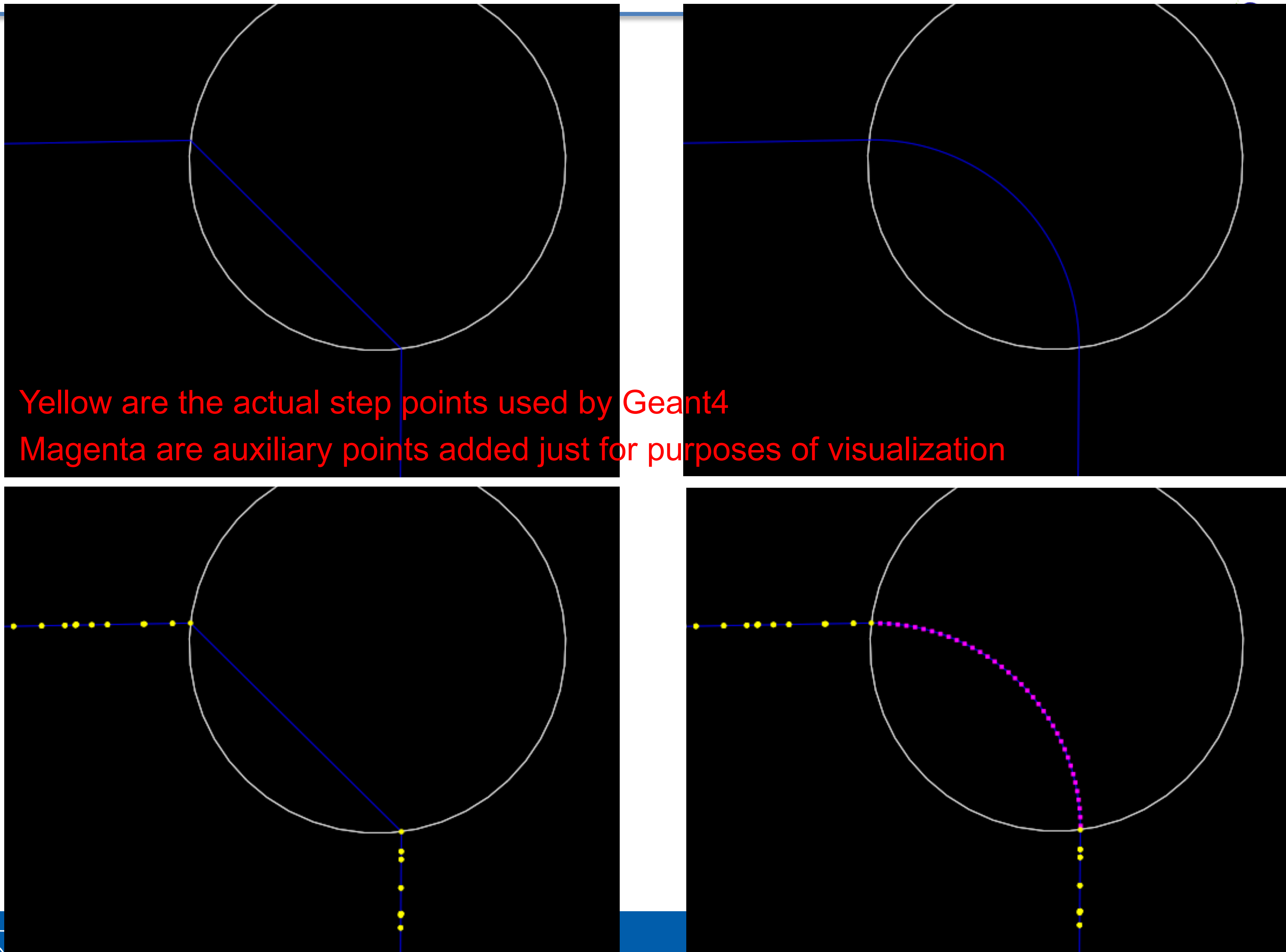


Tracking in field

- 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 may consist of several helix turns.
- User can set the accuracy of the volume intersection,
  - By setting a parameter called the "miss distance"
    - It is a measure of the error in whether the approximate track intersects a volume
    - It is compared with the estimated saggita of a chord
    - It is quite expensive in CPU performance to set too small "miss distance".



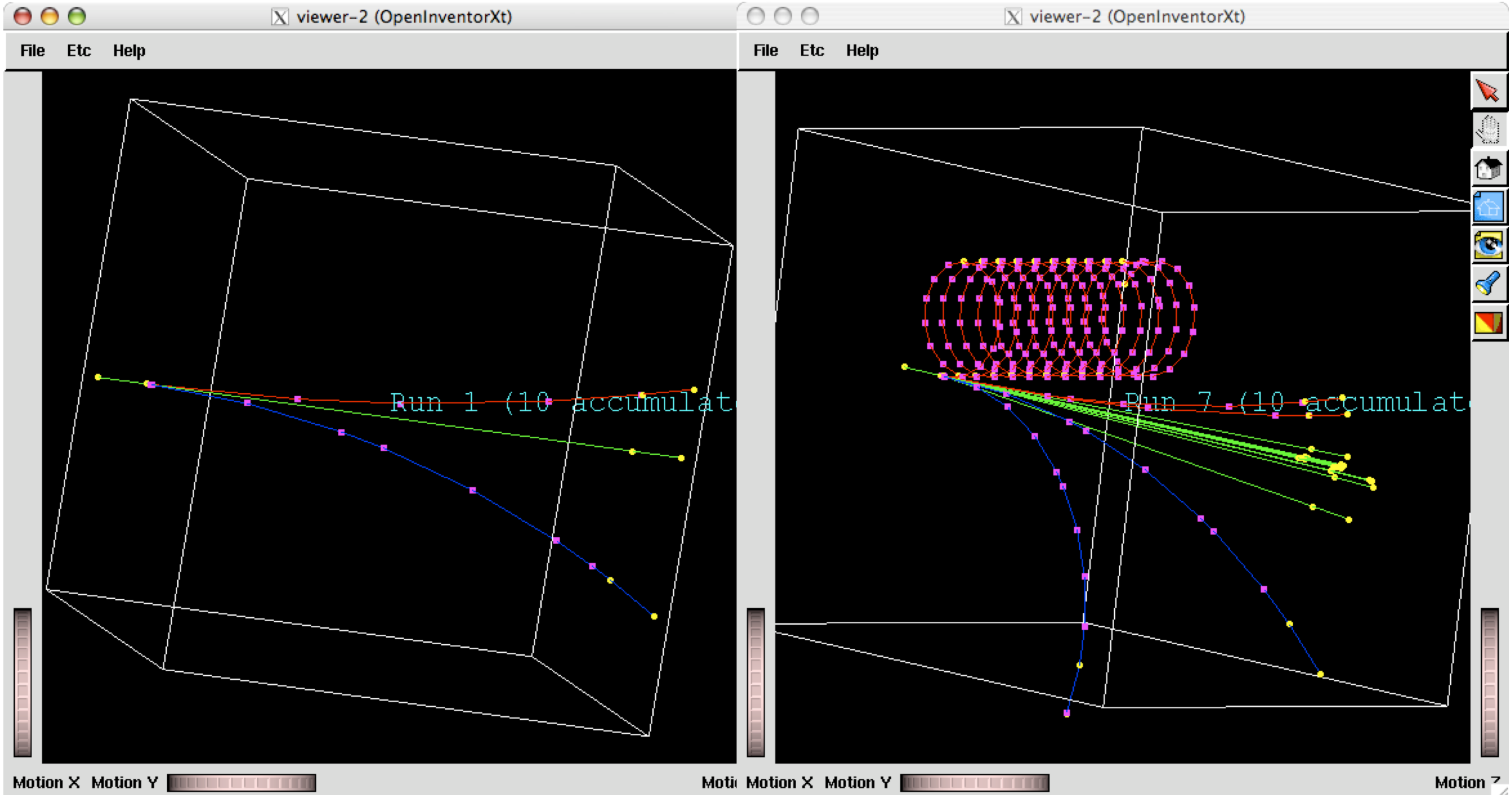
Regular versus Smooth Trajectory



Yellow are the actual step points used by Geant4

Magenta are auxiliary points added just for purposes of visualization

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



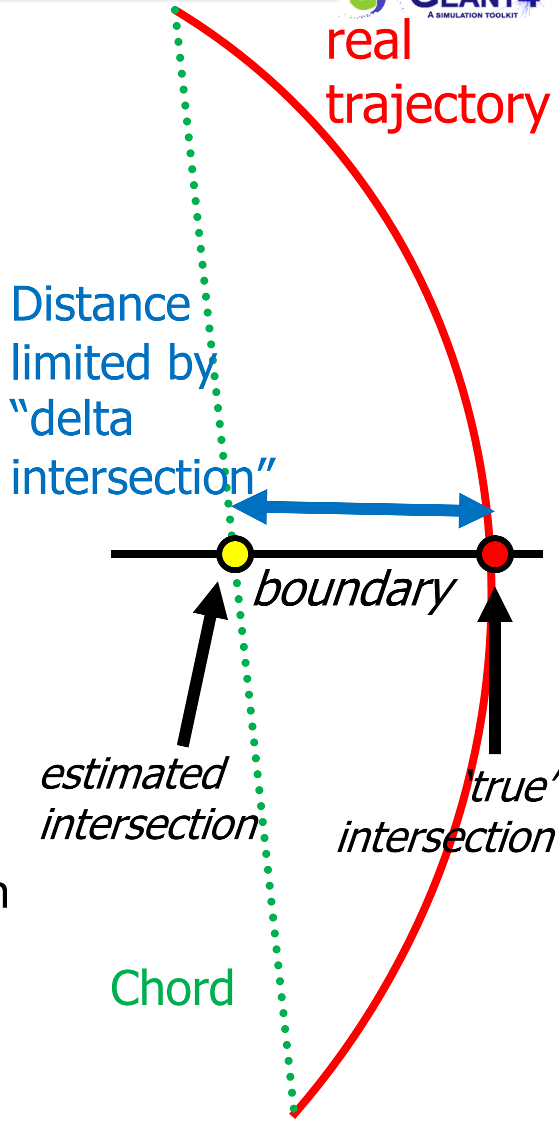
- Yellow dots are the actual step points used by Geant4
- Magenta dots are auxiliary points added just for purposes of visualization

Version 10.5

Tuning precision of tracking in field

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 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 our 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, the user can limit the effect of this bias.



Tunable parameters

- The "epsilon" parameters guide the accuracy for the endpoint of 'ordinary' integration steps, ones 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
- These tunable parameters can be set by

```
theChordFinder->SetDeltaChord( miss_distance );
theFieldManager->SetDeltaIntersection( delta_intersection );
```

The best way to obtain a specific precision for the integration is to give a maximum relative error allowed:

```
double epsilon = 1.0e-6;
theFieldManager->SetEpsilonMax( epsilon );
```

Typically the same value should also be set to the EpsilonMin parameter as well:

```
theFieldManager->SetEpsilonMin( epsilon );
```

- For more look in Section 4.3 (Electromagnetic Field) of the "Guide for Application Developers".

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