Fermilab | U.S. DEPARTMENT OF ENERGY | Office of Science

Geometry 3
Field Integration

Soon Yung Jun (Fermilab)

9th International Geant4 Tutorial in Korea

Dec 5-9, 2022

KiSTi www.kisti.re.kr | 국립암센터 NATIONAL CANCER CENTER | PAL POHANG ACCELERATOR LABORATORY | 연세대학교 방사선융합공학과 미래의료방사선 융합교육연구팀 | Jefferson Lab | Fermilab | CERN | cnrs IN2P3

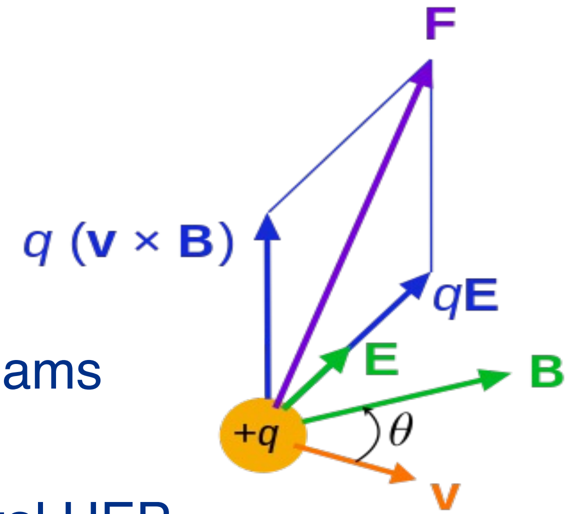
Contents

This talk is based on Geant4 tutorial materials from past events, especially slides by John Apostolakis (CERN) and Makoto Asai (Jefferson Lab).

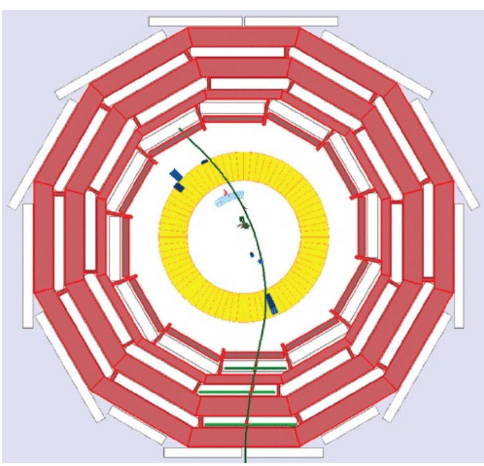
Introduction

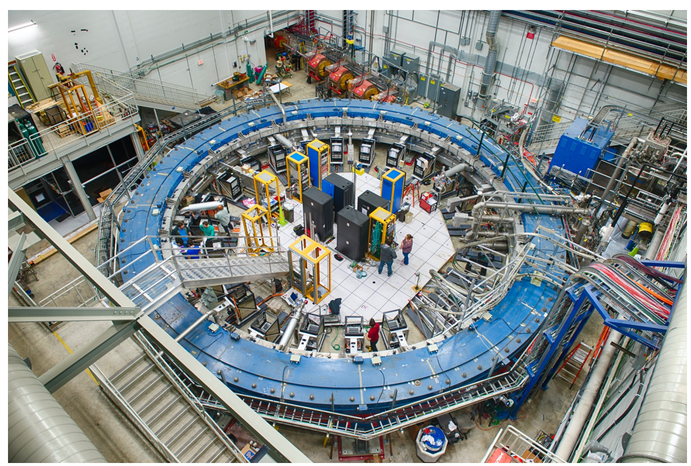
- Charged particle transport in an EM field and uses in experiments

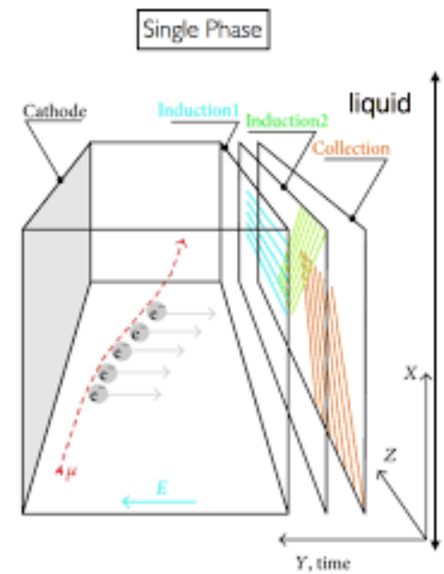
$$\mathbf{F} = q\,(\mathbf{E} + \mathbf{v} \times \mathbf{B})$$



  - Measure momenta of charged tracks
  - Bend or accelerate charged particles or beams
  - Drift charged particles
  - And many other applications in experimental HEP







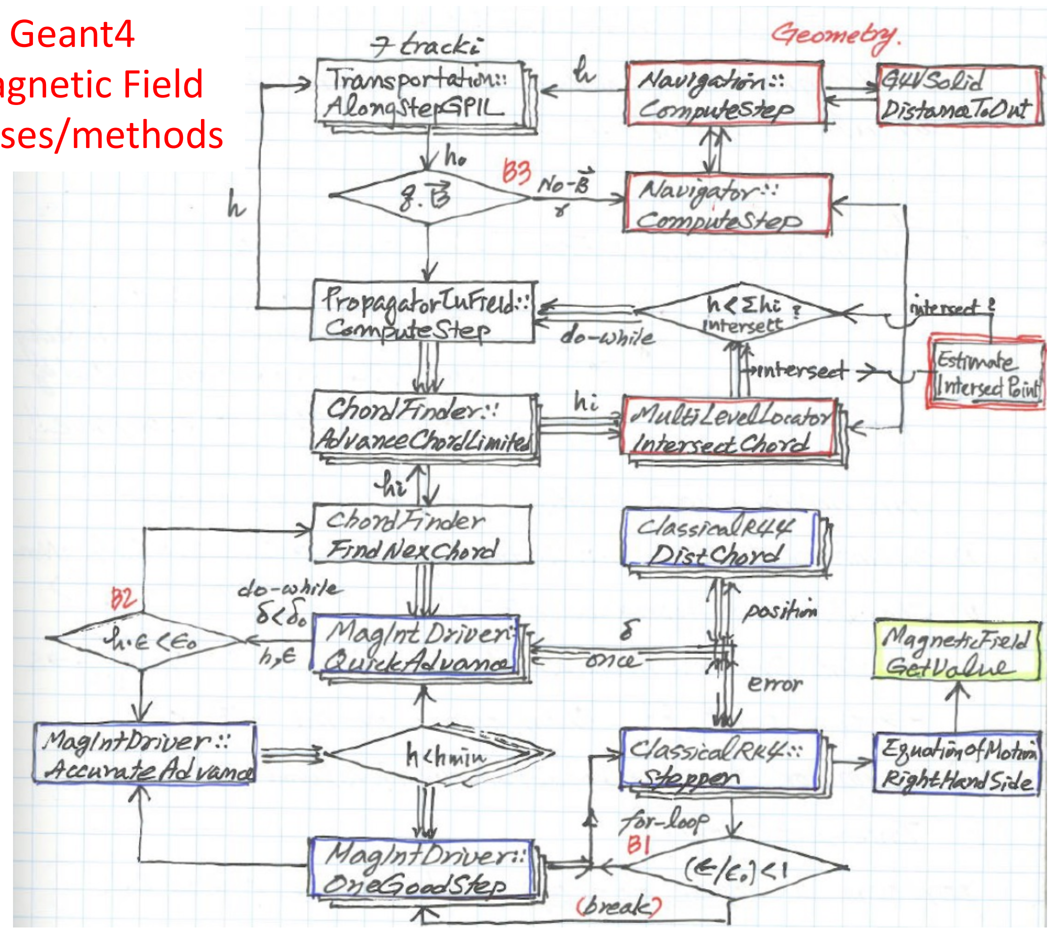
Magnetic Field Support in Geant4

- Geant4 provides general user interfaces for field implementations
  - *G4MagneticField* (the base for a user magnetic field)
  - *G4VUserDetectorConstruction::ConstructSDandField*
  - Also supports *G4UniformMagField* (and *G4UniformElecticField*)
- Supports various field integration methods (steppers)
  - The current default stepper is *G4DormandPrince745*
  - Many other *Runge-Kutta* family integrators including *G4ClassicalRK4*
  - QSS (quantized state system) and symplectic algorithms are in integration
- Provides tunable parameters to control accuracy and performance
- Source codes and examples
  - source/geometry/magneticfield
  - source/geometry/navigation (*G4PropagationInField*)
  - examples/basic/B2 and B5
  - examples/extended/field

Magnetic Field Integration and Driver

- Goal: find the final position and direction of a charged particle in a magnetic (electric or both) field for a given step within a tolerance
- Guiding principle: stride in a plain (slow varying field), crawl in a valley (stiff field)
  - Accuracy
  - Performance
- Field components
  - Field
  - Equation of motion
  - Stepper
  - Driver
  - Chord Finder
  - (Multi-level Locator)
  - Propagator in Field
  - Transportation

Geant4
Magnetic Field
Classes/methods



Magnetic Field Implementation

Magnetic Field Implementation

1) Create a user magnetic field class derived from *G4MagneticField*
   – Implement the mandatory *MyField::GetFieldValue(…)* method
2) Instantiate the user field in the user detector construction
   – Set the field to *G4FieldManager* in *MyDetector::ConstructSDandField()*

```
//--------------------------------------------------------------------//
/*!
 *  A user magnetic field class
 */
class MyField : public G4MagneticField
{
  public:
    MyField();
    ~MyField() override;

   // Return field values at a given point
   void GetFieldValue(const G4double point[4], G4double* field) const;
};
```

```
//--------------------------------------------------------------------//
/*!
 * Evaluate the field at a given position [and time]
 *
 * \param point[4] input position and time, point[4] = {x, y, z, t}
 * \param *field returning magnetic values, field[3] = {Bx, By, Bz}
 */
void MyField::GetFieldValue(const G4double point[4], G4double* field) const
{
  // Implementation detail
  ;
}
```

```
//! \file MyDetector.cc

// Static thread_local stoage
G4ThreadLocal MyField* MyDetector::fMyField = 0;
G4ThreadLocal G4FieldManager* MyDetector::fFieldManager = 0;

//----------------------------------------------------//
/*!
 *  Construct sensitive detectors and a user magnetic field
 */
void MyDetector::ConstructSDandField()
{
  // Create a user field
  fMyField = new MyField();

  // Set the user field to the field manager
  G4FieldManager* fFieldManager = new G4FieldManager();
  fFieldManager->SetDetectorField(fMyField);
}
```

- For a uniform magnetic field, use the *G4UniformMagField* class

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

Global and Local Fields

- One field manager (*G4FieldManager*) is associated with the 'world' and it is set in *G4TransportationManager*

```
// Set the user field to the field manager of G4TransportationManager
G4FieldManager* fFieldManager
  = G4TransportationManager::GetTransportationManager()->GetFieldManager();
fieldManager->SetDetectorField(magneticField);
```

- Other volumes can override this
  - An alternative field manager can be associated with any logical volume
  - By default, this is propagated to all its daughter volumes

```
// Override the field manager of a logical volume by a local field manager
G4FieldManager* localFieldMgr = new G4FieldManager(magneticField);
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.

- A field can be nullified in a volume with a *nullptr* of G4MagneticField

```
G4MagneticField* bField = nullptr;
logVolume->SetFieldManager(new G4FieldManager(bField));
```

Field Integration

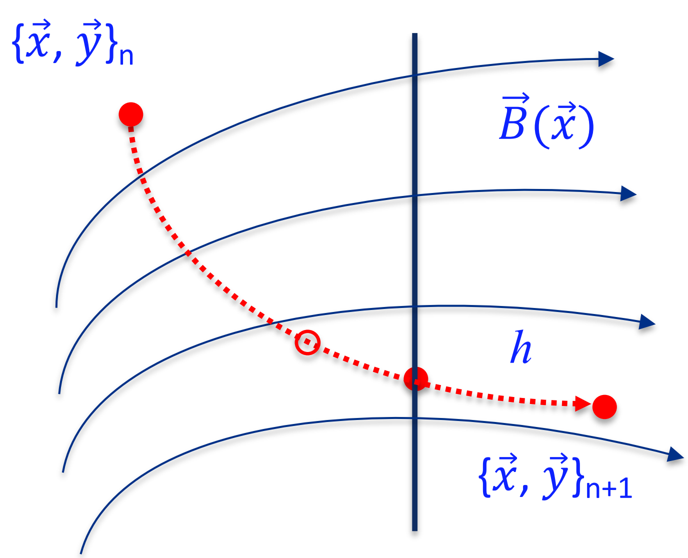
Field Integration

- In order to propagate a charged particle inside an electric or magnetic field (or both), we solve the equation of motion

$$m\frac{d^2\vec{x}}{dt^2} = q[\vec{E} + \vec{v} \times \vec{B}] = f(\vec{x}, t)$$

- Decompose the equation along the particle trajectory, $s = vt$

$$\frac{d\vec{x}}{ds} := \vec{y}$$
$$\frac{d\vec{y}}{ds} = f(\vec{x}, \vec{y})$$



- Use an integration method to find $\{x_{n+1}, y_{n+1}\}$ for any given step size h

$$x_{n+1} = x_n + h$$
$$y_{n+1} = y_n + hf(x_n, y_n)$$

Explicit Runge-Kutta Integration

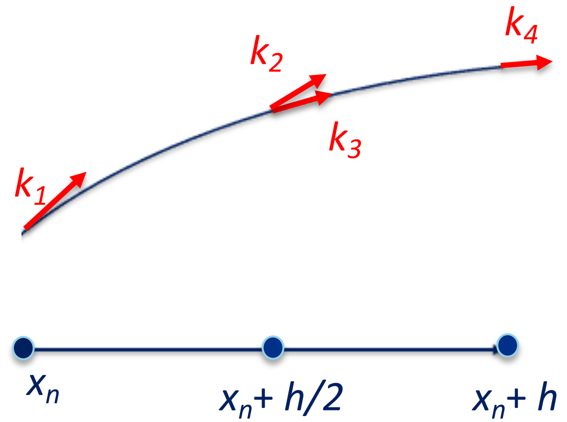
- The Taylor expansion of rhs at the set of intermediate points, $h_i = c_i h$ ($i=1,2,\ldots,s$, the number of stages) subject to $\Sigma b_i = 1$ and $\Sigma a_{ij} = c_i$

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i + \mathcal{O}(h^{s+1}), \quad k_i = f(x_n + c_i h,\ y_n + h\sum_{j=1}^{i-1} a_{ij}k_j)$$

- The classical 4[th] order Runge-Kutta solution (order = # of stages)

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(h^5)$$

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 &= f(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_4 &= f(x_n + h, y_n + k_3) \end{aligned}$$



  - Adaptive step control by a truncation error: difference between two small steps and one big step: total 11 evaluations of rhs of the equation
  - If the error is bigger than a tolerance, propose a new h_t (sub-step) and repeat this until $h = \Sigma h_t$.

Dormand-Prince RK5(4)7M

- Use the higher order solutions (the *5th* order RK) and a *4th* order embedded solution

$$y_{n+1} = y_n + h\sum_{i=1}^{6} b_i k_i + \mathcal{O}(h^6)$$

$$y^*_{n+1} = y_n + \sum_{n=1}^{7} b^*_i k_i + O(h^5)$$

$$y_{err} = y_{n+1} - y^*_{n+1} = \sum_{n=1}^{7}(b^*_i - b_i)k_i$$

Butcher Table

c_i	a_ij						
0							
1/5	1/5						
3/10	3/40	9/40					
4/5	44/45	-56/15	32/9				
8/9	19372/6561	-25360/2187	64448/6561	-212/729			
1	9017/3168	-355/33	46732/5247	49/176	-5103/18656		
1	35/384	0	500/1113	125/192	-2187/6784	11/84	
b*_i	35/384	0	500/1113	125/192	-2187/6784	11/84	0
b_i	5179/57600	0	7571/16695	393/640	-92097/339200	187/2100	1/40

  - It uses 6 field evaluations per integration, as it provides the derivative at the endpoint (avoid to calculate it at the start of the next step, FSAL)
  - RK5(4)7M is the most efficient and stable one among algorithms in J. R. Dormand and P. J. Prince, J. Comp. and App. Math., v6, no 1 (1980)
- Other available steppers or embed solutions provided by Geant4
  - BogackiSampine45 (and 23)
  - Runge-Kutta-Felberg (4th order embedded solution)
  - CashKarper (4th order) and etc.

Field Parameters

Tunable Parameters

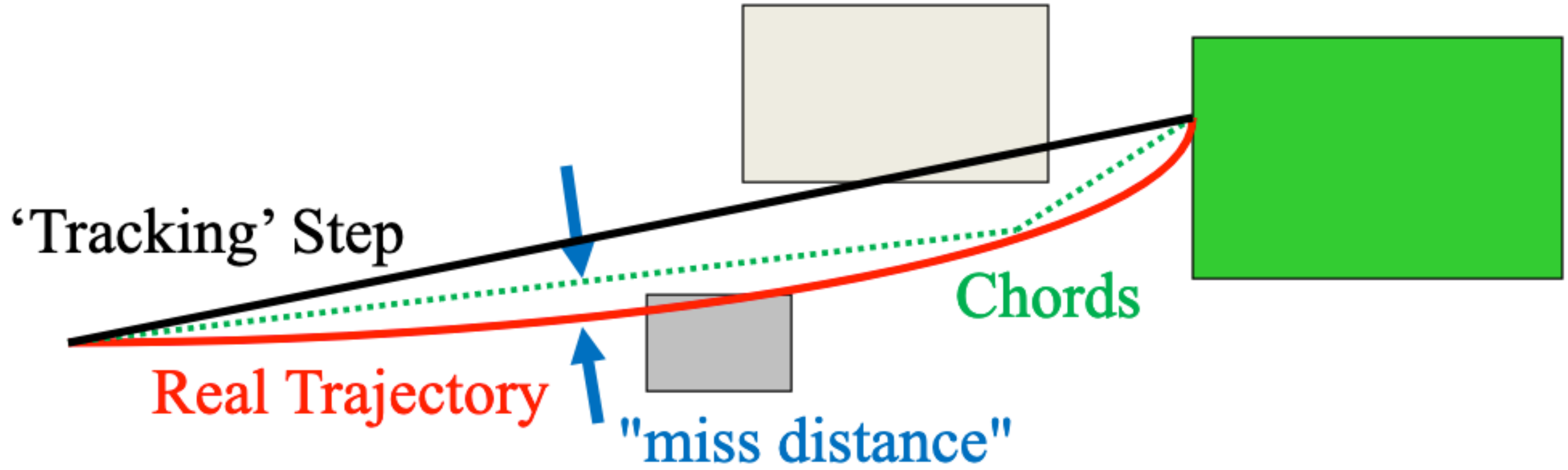
- The most important accuracy parameter is the maximum relative tolerance ε_{max} for the integration error for a given step *s* and particle momentum *p*
  - ε_{max} limits the estimated error for large steps: $|\Delta x| < \varepsilon_{max}\, s$ and $|\Delta p| < \varepsilon_{max}\, |p|$
- The "delta one step" parameter is accuracy for endpoint of integration steps that do not intersect a volume boundary.
  - It also limits on the estimated error of the endpoint of each physics step (essentially it is < 1000 δ_1-step.)
  - Values of δ-intersection and δ_1-step should be within one order of magnitude.

```
// Tunable parameters can be set by, for an example
fChordFinder->SetDeltaChord(miss_distance);

fFieldManager->SetDeltaIntersection(delta_intersection);
fFieldManager->SetDeltaOneStep(delta_one_step);
fFieldManager->SetEpsilonMax(epsilon_max);
fFieldManager->SetEpsilonMin(0.1 * epsilon_min);
```

- Further details are described in Section 4.3 (Electromagnetic Field) of the Geant4 Application Developers Guide.

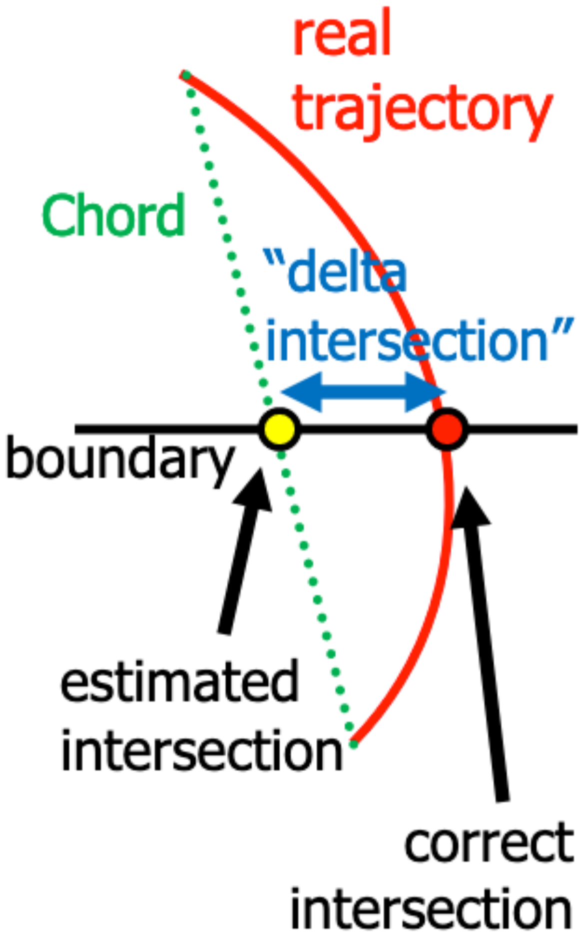
Miss Distance

- Depending on the error from the integration, Geant4 breaks up the curved path into linear chord segments which approximate the path
- 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
- User can set the accuracy of the volume intersection by a "miss-distance" which is a measure of the error whether the approximate track intersects a volume (CPU performance is sensitive to this)



Delta Intersection

- The "delta intersection" (δ_{int}) 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 (boundary crossing) algorithm 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.
- The user can set this parameter to adjust the accuracy and performance of charged particle tracking in a field.



Customizing Field Integration

Customizing Field Integration

- Runge-Kutta integrations are used to trace a charged particle in a general field. There are many general (low or high order) steppers to choose, and specialized steppers for pure magnetic fields.
- The default is the general purpose *G4DormandPrinceRKF45* an embedded 4th-5th order RK stepper (Embedded = compares 4th & 5th order to estimate error)
  - If the field is very smooth, you may consider higher order steppers
  - of most potential interest in large volumes filed with gas or vacuum
- If the field is calculated from a field map, a lower order stepper is recommended. The less smooth field, the lower order of a stepper that should be used. The choice of lower order steppers includes
  - The 3rd order: G4SimpleHeum
  - The 2nd order: G4ImplicitEuler and G4SimpleRunge
  - The 1st order: G4ExplicitEuler (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

Creating a Different Stepper

- Example of how to choose a different field stepper and driver

```
//--------------------------------------------------------------------------//
/*!
 *  An example of how to choose a different field stepper and driver
 */
void MyDetectorConstuction::ConstructSDandField()
{
  // Create a user field or use G4UniformMagneticField(G4ThreeVector(0, 0, Bz))
  G4MagneticField* bField = new MyMagneticField();

  // Create the equation of motion (include G4MagIntegratorDriver.hh)
  auto equation = new G4Mag_UsualEqRhs(bField);

  // Create the integration stepper
  auto stepper = new G4DormandPrince745(equation, fNvariables);

  // Create the integration driver, which manages the error control
  auto driver = new G4IntegrationDriver(fMinStep, stepper, fNvariables);

  // Create the chord finder, and set the field manager
  G4FieldManager* fieldManager
    = G4TransportationManager::GetTransportationManager()-> GetFieldManager();
  fieldManager->SetDetectorField(bField);
  fieldManager->SetChordFinder(new G4ChordFinder(driver));
}
```

  – Note: the default fNvariables = 6, {x, y , z, pz, py, pz}, but it can be extended for time and polarization(or spin) components

Basic and Extended Examples of Field

- examples/basic
  - B2: Use G4GlobalMagFieldMessenger to create a global, uniform magnetic field
  - B5: Creating a custom magnetic field and assigning it to a field
- examples/extended/field
  - field01: exploring integration methods
  - field02: a combined E+B (electric + magnetic) field
  - field03: defining a local field in a logical volume
  - field04: overlapping field elements (magnetic, electric or both)
  - field05: tracking of polarization and spin-frozen condition
  - field06: tracking ultra cold neutrons in a gravity field
  - BlineTracer: trace and visualize magnetic field lines

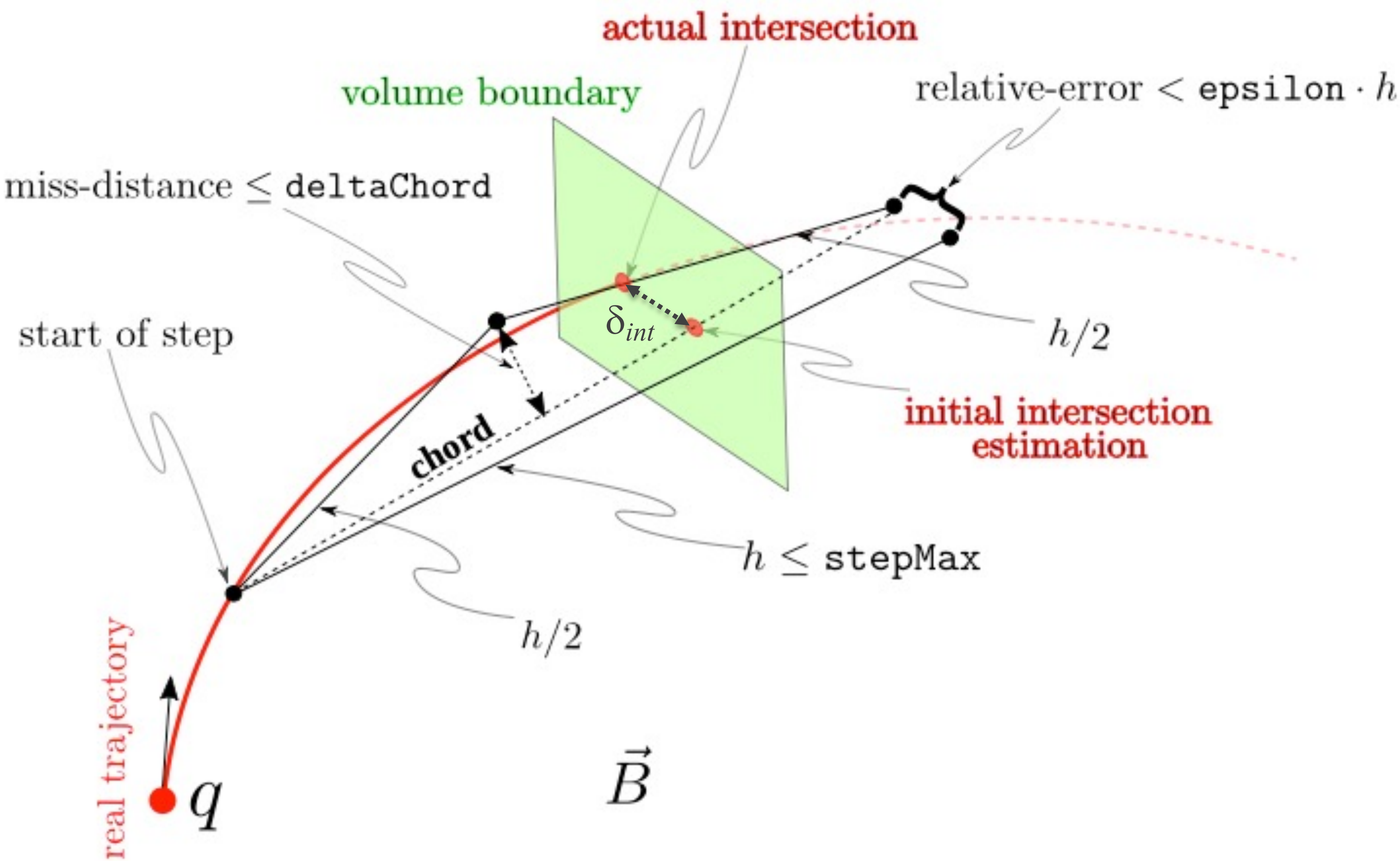
Summary: Propagation in a Magnetic Field

- Geant4 supports general user interfaces for field implementation
  - *G4MagneticField::GetFieldValue*
  - *G4VDetectorConstructopn::ConstructSDandField*
- Runge-Kutta (RK) integration is used to track a charged particle in any magnetic, electric, combined EM, gravitational or a mix field
  - Many general steppers are available/applicable to any equation/field
- Default in Geant4 is the general purpose *G4DormandPrince745* with a 5^{th} order of the RK stepper with the 4^{th} order embedded solution
- Different types of integration methods are available and Geant4 provides interfaces to control field parameters for accuracy and performance tunings and to customize the user field integration

Backup

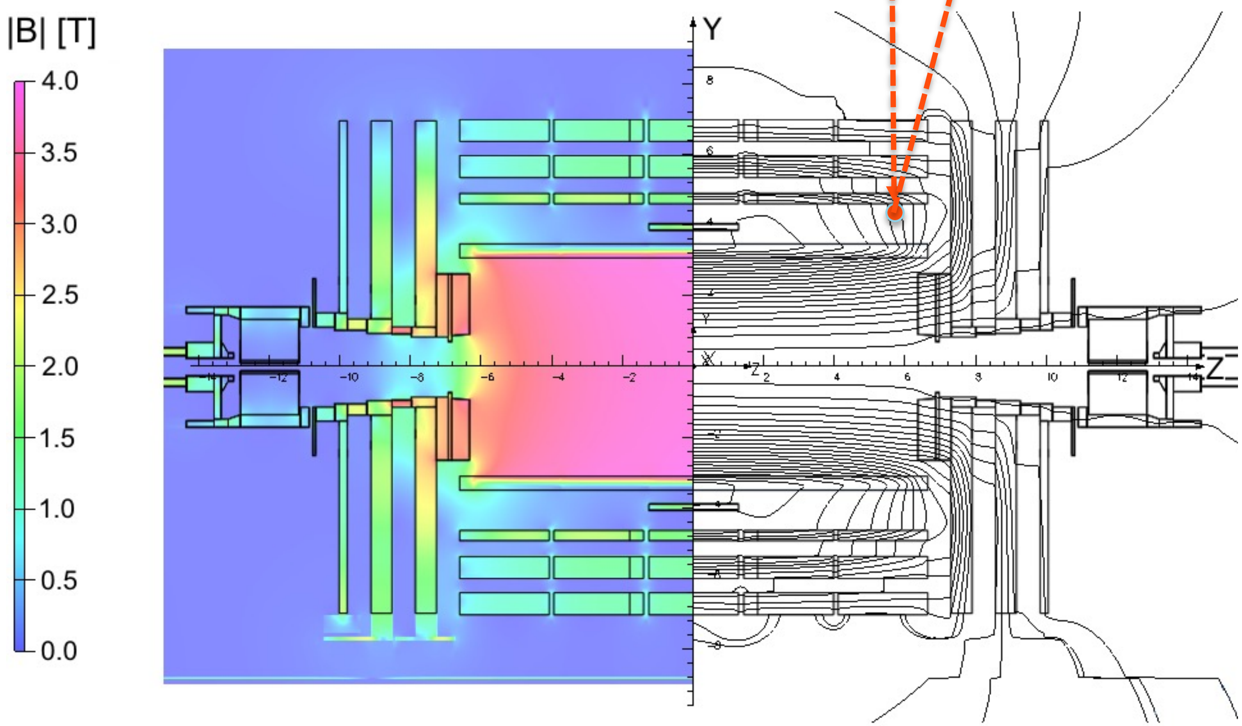
Field Parameters

- Geant4 provides parameters to control accuracy and performance for a charged particle transport in various types of fields



Field Example: CMS Magnetic Field

- A user task: implement *GetFieldValue(point, B)* which returns the magnetic field value ***B*** in the global coordinate system for the given point (position for a static field)



CMS magnetic field: value of |B| (left) and field lines (right)
Chatrchyan S. *et al* arXiv:0910.5530; CMS-CFT-09-01

Example of Magnetic Field Implementation

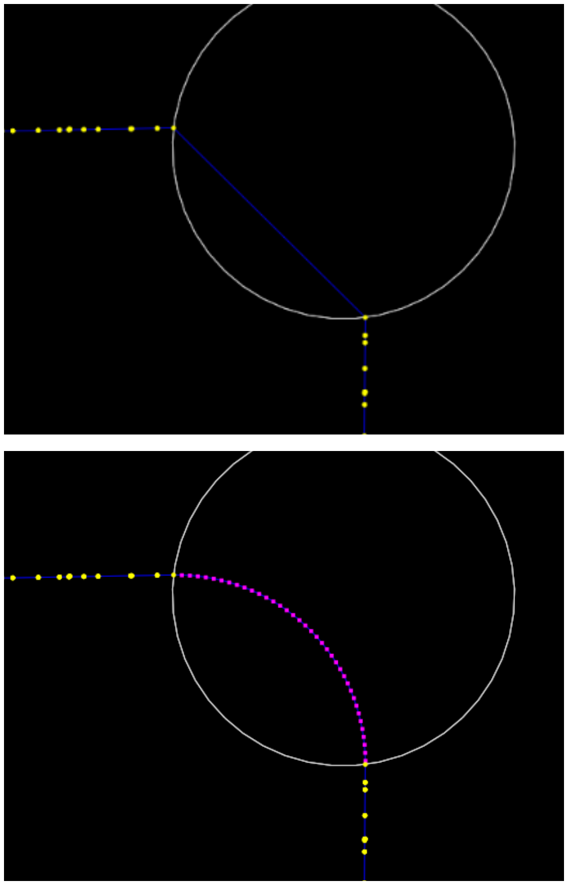
- examples/basic/B5 is a good starting point

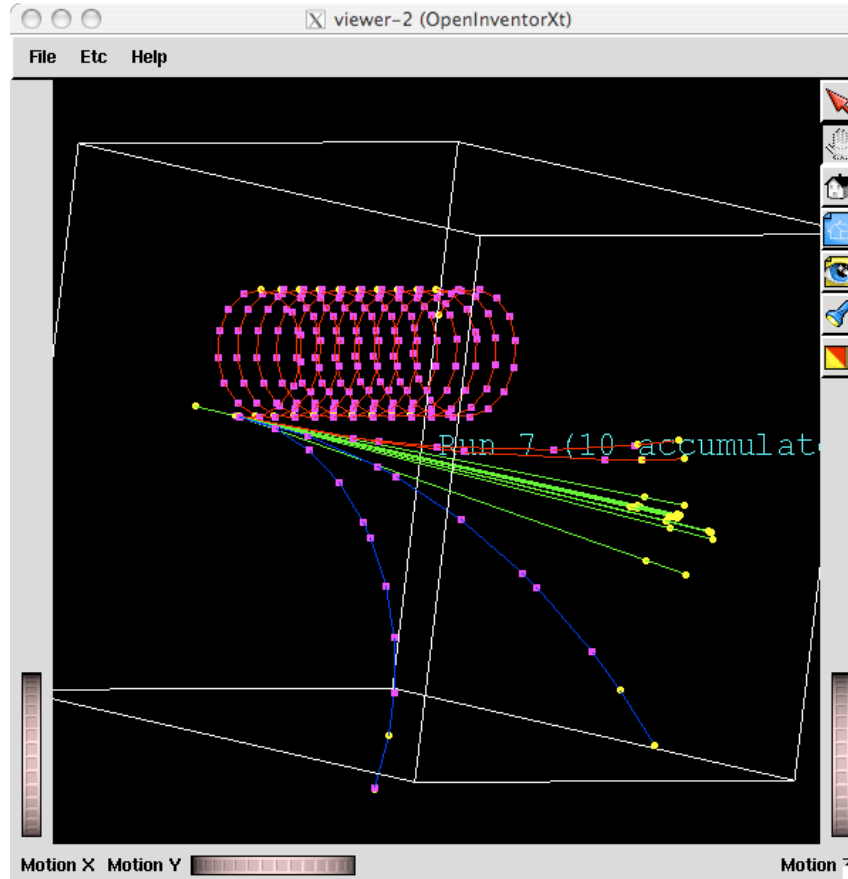
```
//------------------------------------------------------------------------//
/*!
 *  Construct sensitive detectors and a user magnetic field:
 *  Please see examples/basic/B5/src/DetectorConstruction.cc
 */
void DetectorConstruction::ConstructSDandField()
{
  // Sensitive detectors:
  ;;;

  // Create a user MagneticField and set it to the thread local G4FieldManager
  fMagneticField = new MagneticField();
  fFieldMgr = new G4FieldManager();
  fFieldMgr->SetDetectorField(fMagneticField);
  fFieldMgr->CreateChordFinder(fMagneticField);
  G4bool forceToAllDaughters = true;
  fMagneticLogical->SetFieldManager(fFieldMgr, forceToAllDaughters);

  // Register the field and its manager for deleting
  G4AutoDelete::Register(fMagneticField);
  G4AutoDelete::Register(fFieldMgr);
}
```

  - fMagneticLogical is a *G4LogicalVolume* of a certain *G4VSolid* in which the local magnetic field, fMagneticField is assigned

Regular versus Smooth Trajectory

- Yellow dots are actual step points by Geant4 and magenta dots are auxiliary pointed added just for the purpose of visualization

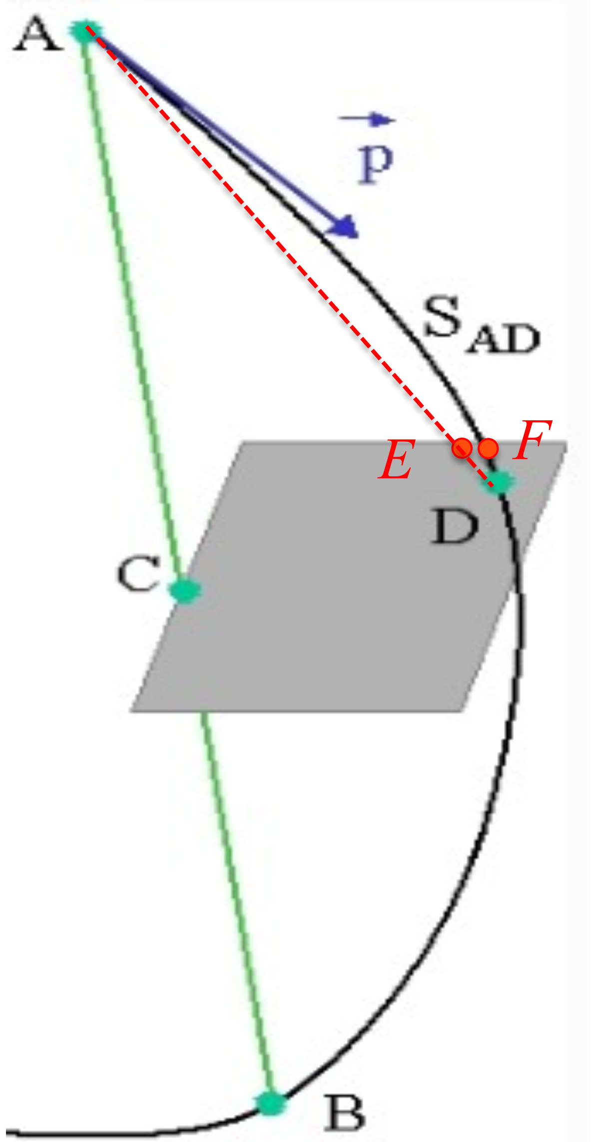




- Smooth trajectory makes big difference for trajectories loop in magnetic field

Estimation of Intersection Point

- In intersecting approximate path with volume boundary
  - In trial step AB, an intersection is found with a volume at C
  - Step is broken up, choosing D, so
    $S_{AD} = S_{AB} * |AC| / |AB|$,
  - If $|CD| < \delta_{int}$, then C is accepted as intersection point
  - If C is rejected and D is not in the volume of A, a new intersection E is tried,
    $S_{AF} = S_{AD} * |AE| / |AD|$
    Otherwise, use B instead of A (backward integration)
  - Test $|EF| < \delta_{int}$, and repeat this until the condition is satisfied
  - δ_{int} is the maximum bias for the position



Customizing Field Integration

- Trying a few different types of steppers for a particular field or application is suggested if maximum performance is a goal
  - What is the most performant option may be different in different regions
  - e.g., depending on whether the field is varying greatly
- Specialized steppers for pure magnetic fields are also available. They take account of the fact that a local trajectory in a slowly varying field will not vary significantly from a helix stepper
  - Combining this in a variation, Runge-Kutta methods can provide higher accuracy at lower computational cost when large steps are possible.
  - Suggested are *G4HelixSimpleRunge* and *G4HelixSimpleHeum*
- To change the stepper

```
// Change the stepper
G4HelixSimpleRunge *newStepper = new G4HelixSimpleRunge(equation_rhs);
fChordFinder->GetIntegrationDriver()->RenewStepperAndAdjust(newStepper)
```

- Further details are described in Section 4.3 (Electromagnetic Field) of the Application Developers Manual.

Other Types of Field

- For pure electric field, Geant4 provides *G4ElectricField* (base) and the simple *G4UniformElectricField* (concrete) classes.
- *G4ElectroMagneticField* is the base for combined electro-magnetic fields and the equation of motion for it is *G4EqMagElectricField*

```
//----------------------------------------------------------------------------//
/*!
 *  An example of how to use other types of field, electric or combined E-B field
 */
void MyDetectorConstuction::ConstructSDandField()
{
  // Create a user defined EM field which is derived from G4Field or use the
  // uniform electric field G4UniformElectricField(const G4ThreeVector& efield);
  G4ElectricMagneticField* emField = new G4ElectroMagneticField();

  // Create the equation of motion
  auto equation = new G4EqMagElectriField(emField);

  // The rest of code is same as one in the previous page
  ...

}
```

  - In a combined EM field, the return values of the *GetFieldValue* are fieldValue[0-2] = {Bx, By, Bz} and fieldValue[3-5] = {Ex, Ey, Ez}
  - A user can create their own type of field, inheriting from *G4Field*, and must create an associated equation of motion (inherits from G4EqRhs)