Geant4 11.3.2

Geometry 3: Field Integration

Dennis Wright

Geant4 Tutorial at Jefferson Lab

20 August 2025

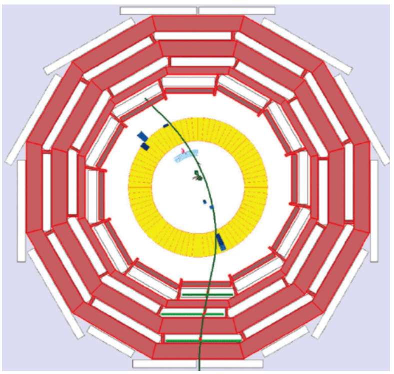
based on slides by Soon Yung Jun (Fermilab), John Apostolakis (CERN) and Makoto Asai (Jefferson Lab)

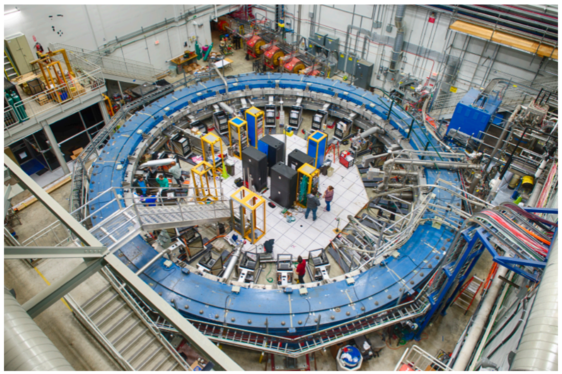
Outline

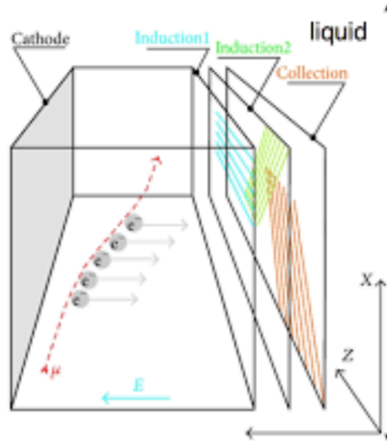
- Introduction
- Magnetic field implementation
- Field integration
- Field parameters
- Customizing field integration

Introduction

- Changed particle transport in an EM field: $F = q(\mathbf{E} + \mathbf{v} \times \mathbf{B})$
- Used for:
  - measuring momenta of charged tracks
  - bending or acceleration of charged tracks or beams
  - drifting charged particles through volumes
  - many other HEP applications







Magnetic Field Support in Geant4

- General user interfaces provide for field implementations
  - G4MagneticField (magnetic field base class)
  - G4VUserDetectorConstruction::ConstructSDandField (method to place field in detector)
  - G4UniformMagField, G4UniformElectricField (special case for simple fields)

- Field integration methods
  - Default stepper is G4DormandPrince745
  - Many other Runge-Kutta family integrators, including G4ClassicalRK4
  - QSS (quantized state system) and symplectic algorithms are used in integration

- Tunable parameters to control accuracy and performance
- Source code and examples:
  - source/geometry/magneticfield
  - source/geometry/navigation (G4PropagationInField)
  - examples/basic/B2 and B5
  - examples/extended/field

Magnetic Field Integration and Driver

- Goal: for a given step within a field, find final position and direction of charged particle within a tolerance
- Guiding principle:
  - stride in a plain (take big steps if field varies slowly)
  - crawl in a valley (small steps in a stiff, rapidly varying field)
  - → better accuracy, performance

- Elements of field integration
  - Field (E, B, …)
  - Equation of motion
  - Stepper (integration routine, e.g. RK4, …)
  - Driver (code that guides stepper, controls step size, errors, …)
  - Chord finder (splits complex path into straight segments)
  - Multi-level locator (finds intersection of track with volume boundary)
  - Propagator in field (navigates particle through a field)
  - transportation (advance the particle by integrated step)

Field Implementation

Magnetic Field Implementation

- Derive a user magnetic field class from G4MagneticField and implement the method MyField::GetFieldValue(…)
- Instantiate this field in user detector construction and pass 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 class G4UniformMagField

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

Global and Local Fields

- One field manager is associated with the World and 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" pushes 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

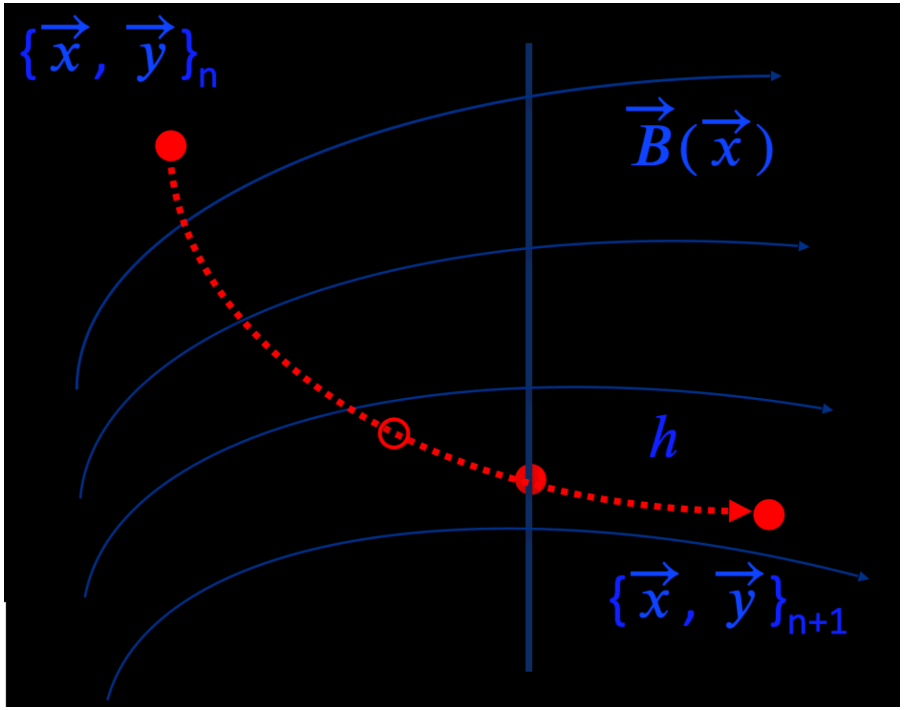
```
G4MagneticField* bField = nullptr;
logVolume->SetFieldManager(new G4FieldManager(bField));
```

Field Integration

Integrating the Equations of Motion

- Solve the equation of motion of a charged particle in a field:

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



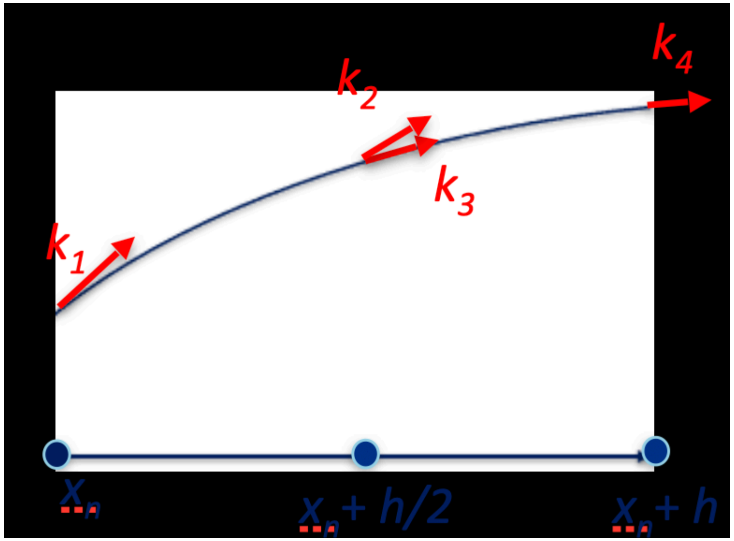
- Decompose equation along the particle trajectory, s = vt:
  - $\frac{d\mathbf{x}}{ds} = \mathbf{y}$
  - $\frac{d\mathbf{y}}{ds} = f(\mathbf{x}, \mathbf{y})$

- Use an integration method to find x_{n+1}, y_{n+1} for any given step size:
  - $x_{n+1} = x_n + h$
  - $y_{n+1} = y_n + hf(x_n, y_n)$

Explicit Runge-Kutta Integration

- Taylor expansion of right hand side at a set of intermediate points $h_i = c_i h$ $(i = 1,\ldots,s)$ where s is the number of stages, subject to $\Sigma_i b_i = 1$ and $\Sigma_j a_{ij} = c_i$:
  - $y_{n+1} = y_n + \Sigma_{i=1}^{s} b_i k_i + O(h^{s+1})$
    - $k_i = hf(x_n + c_i h, y_n + h\Sigma_{j=1}^{i-1} a_{ij} k_j)$

- Classical 4th order (4-stage) Runge-Kutta:
  - $y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) + O(h^5)$
    - $k_1 = hf\left(x_n, y_n\right)$
    - $k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$
    - $k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right)$
    - $k_4 = hf\left(x_n + h, y_n + k_3\right)$



- Adaptive step control by truncation error of difference between two small steps and one big step: total of 11 evaluations of right hand side of equation of motion
  - If error is bigger than a given tolerance, propose new substep h_i and repeat until $h = \Sigma_i h_i$

Dormand-Prince RK5(4)7M

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

  - $y_{n+1} = y_n + \Sigma_{i=1}^{7} b_i k_i + O(h^6)$
  - $y_{n+1}^* = y_n + \Sigma_{n=1}^{6} b_i^* k_i + O(h^5)$
  - $y_{err} = y_{n+1} - y_{n+1}^* \;=\; \Sigma_{n=1}^{7} (b_i^* - b_i) k_i$

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

  - Uses 6 field evaluations per integration because it provides the derivative at the end point
  - RK5(4)7M is the most efficient and stable among algorithms

- Other steppers available:
  - BogackiShampine45 (and 23)
  - Runge-Kutta Felberg (4th order embedded solution)
  - Cash-Karp (4th order)
  - And more

Field Parameters

Tunable Parameters

- 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 parameter delta one step (δ_1-step) is the accuracy for the endpoint of integration steps that do not intersect a volume boundary
  - It also limits the estimated error of the endpoint of each physics step (essentially $< 1000\ \delta_1$-step)
  - Values of δ-intersection and δ_1-step should be within one order of magnitude

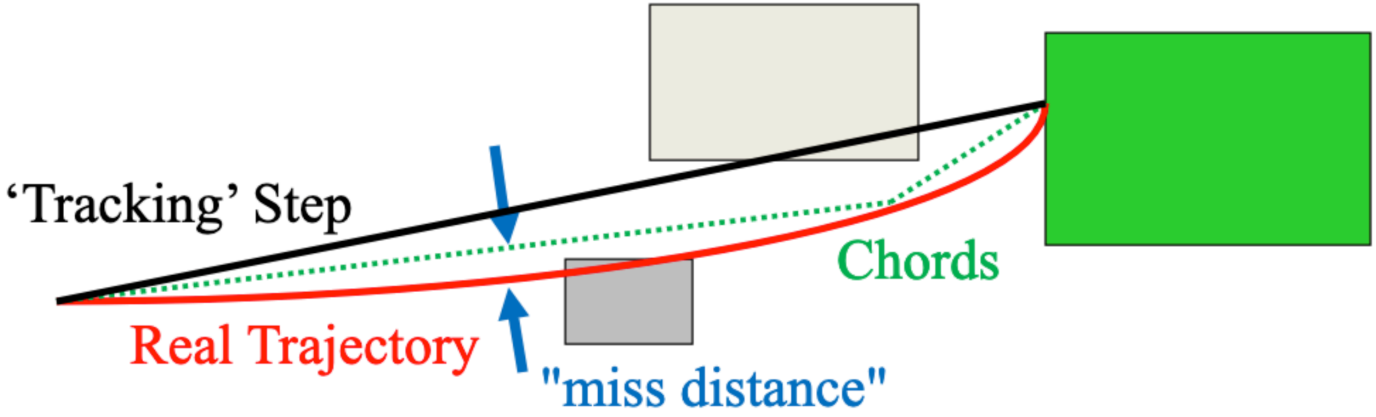
```
// Tunable parameters can be set by, for an example
fChordFinder->SetDeltaChord(miss_distance);

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

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

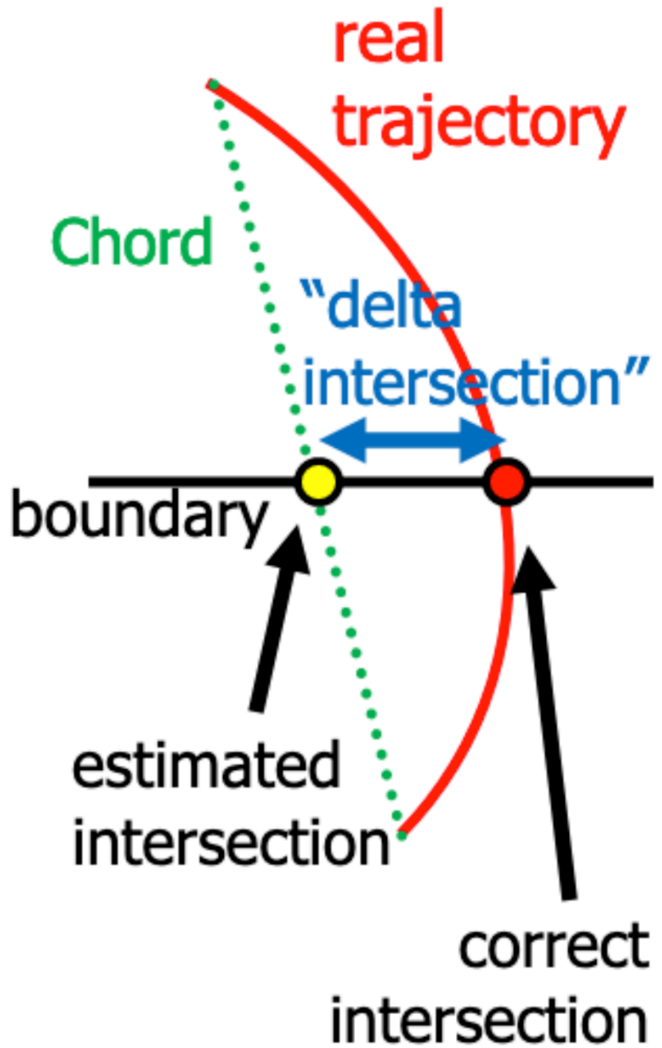
Miss Distance

- Depending on error from integration, Geant4 breaks up curved path into linear chord segments, which approximate the path
- Chords are used 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 an indicator of whether or not approximate track intersects a volume
  - CPU performance is sensitive to this value



Delta Intersection

- Parameter δ_{int} is the accuracy to which an intersection with a volume boundary is calculated
- Especially important because it is used to limit a bias that our boundary crossing algorithm exhibits
- Intersection point is always on the "inside" of that curve
- By setting a value for this parameter that is much smaller than some acceptable error, user can limit the effect of the bias
- User can set this parameter to adjust the accuracy and performance of charged particle tracking in a field



Customizing Field Integration

Choosing a Stepper

- Runge-Kutta integrations are used to trace a charged particle in a general field
  - Many steppers to choose from
  - And specialized steppers for pure magnetic fields

- Default: G4DormandPrinceRFK45
  - Embedded 4th-5th order RK stepper (embedded = compares 4th and 5th order to estimate error)
  - If field is very smooth, may consider higher-order steppers
  - Of most interest in large volumes filled with gas or vacuum

- If field calculated from field map, use a lower-order stepper
  - The less smooth the field, the lower-order the stepper
  - Some low-order steppers:
    - G4SimpleHeum (3rd order)
    - G4ImplicitEuler and G4SimpleRunge (2nd order)
    - G4ExplicitEuler (1st order) - useful only for very rough fields
    - For intermediate (somewhat smooth fields) choice between 2nd and 3rd order is made by trial and error

Example: Setting Up Your Own Stepper/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: default is fnVariables = 6 (x, y, z, px, py, pz) but can be extended to include time, or polarization (spin) components

Basic and Extended Field Examples

- examples/basic
  - B2: use G4GlobalMagFieldMessenger to create global, uniform magnetic field
  - B5: create a custom magnetic field and assign it to a field

- examples/extended
  - field01: exploration of integration methods
  - field02: combined E+B (electric and magnetic field)
  - field03: define 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 gravitational field
  - BlineTracer: trace and visualize magnetic field lines

Summary

- Geant4 supports general user interfaces for field implementation
  - G4MagneticField::GetFieldValue()
  - G4VDetectorConstruction::ConstructSDandField()

- Runge-Kutta (RK) integration is used to track a charged particle in any magnetic, electric, combined EM, gravitational or mixed field
  - Many general steppers are available/applicable for any equation/field

- Default in Geant4 is the general-purpose G4DormandPrince745 which is a 5th order RK stepper with a 4th order embedded solution

- Different types of integration methods are available and Geant4 produces interfaces to control field parameters for accuracy and performance tunings and to customize user field integration