Geant 4

Detector Description:

Sensitive Detector & Field

http://cern.ch/geant4

PART III

Magnetic Field

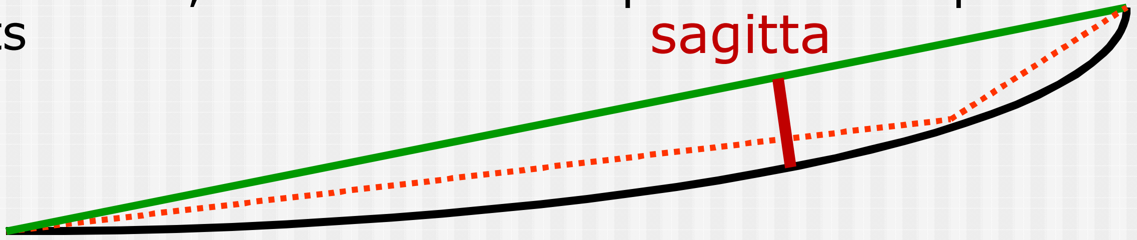
- *Field Propagation & accuracy*
- *Global & Local Field*
- *Tunable parameters*
- *Field Integration*

Field Propagation

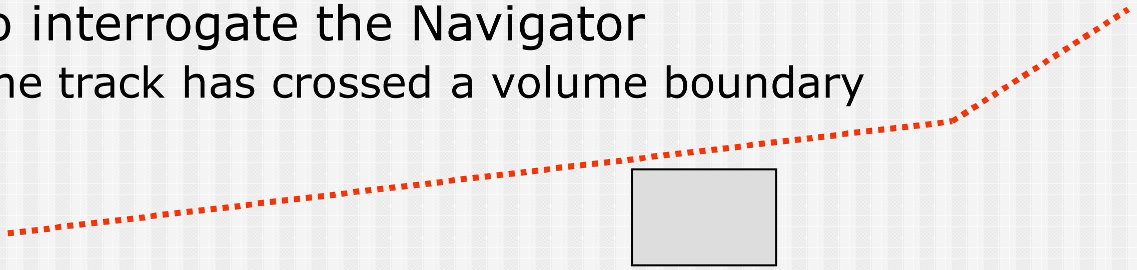
- In order to propagate a particle inside a field (e.g. magnetic, electric or both), we integrate the equation of motion of the particle in the field
- In general this is best done using a **Runge-Kutta** (RK) method for the integration of ordinary differential equations
  - Several RK methods are available
- In specific cases other solvers can also be used:
  - In a uniform field, using the known analytical solution
  - In a nearly uniform but varying field, with RK+Helix

Chords

- Once a method is chosen that allows Geant4 to calculate the track's motion in a field, Geant4 breaks up this curved path into linear chord segments

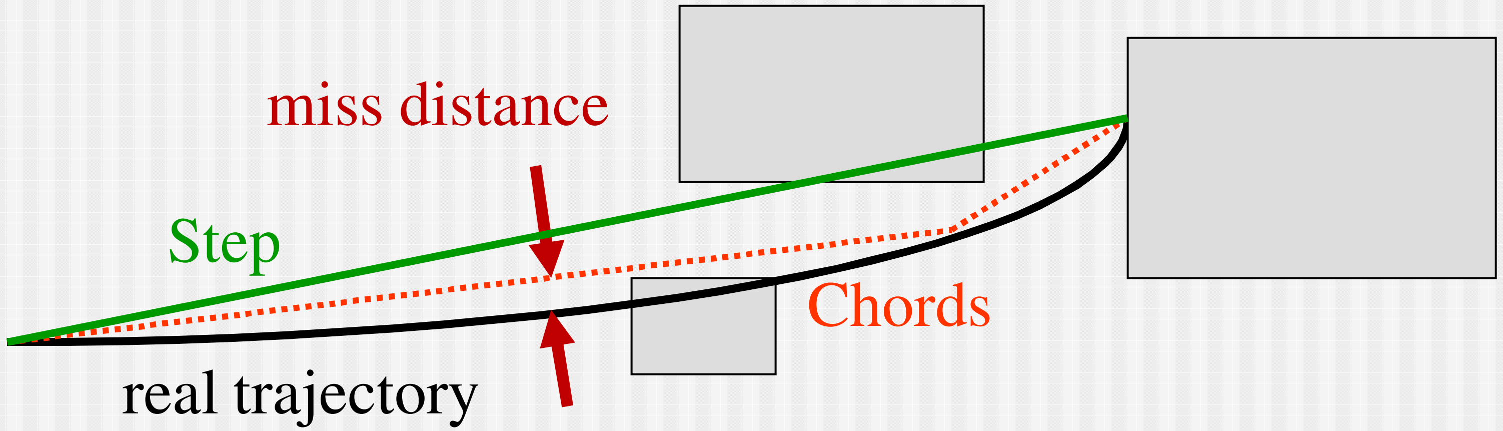


- The chord segments are determined so that they closely approximate the curved path; they're chosen so that their sagitta is small enough
  - The *sagitta* is the maximum distance between the curved path and the straight line
  - Small enough: is smaller than a user-defined maximum
- Chords are used to interrogate the Navigator
  - to see whether the track has crossed a volume boundary



Intersection accuracy

- The accuracy of the volume intersection can be tuned
  - by setting a parameter called the "miss distance"
    - The *miss distance* is a measure of the error resolution by which the chord may intersect a volume
    - Default *miss distance* is 0.25 mm
    - Setting small *miss distance* may be highly CPU consuming
- One step can consist of more than one chord
  - In some cases, one step consists of several turns



Create a Magnetic Field

Field classes are in source/geometry/magneticfield

- How to create Uniform field ?
  - Use the constructor of **`G4UniformMagField`**

    G4UniformMagField(const G4ThreeVector& FieldVector);

```
G4ThreeVector  fieldV ( 0.1 * Tesla, 1.0*Gauss, 0.0 );
G4MagneticField *magField= new G4UniformField( fieldV );
```

- Non-uniform field
  - Concrete class derived from **`G4MagneticField`**
    - Must define the method
      - `void  GetFieldValue( const G4double Point[4], G4double *Bfield ) const;`
  - ..

“Packaging” the Field

- A field is packaged together with properties and accuracy parameters into a field manager
  - Field object
  - Object that holds the Solver (`G4ChordFinder`)
  - Accuracy parameters
- To create a FieldManager
  - `G4FieldManager* localFieldMgr = new G4FieldManager(magField);`
  -

3 steps to setting a global field

- Get the global **`G4FieldManager`**

```
G4FieldManager* globalFieldM=
  G4TransportationManager::GetTransportationManager()
        ->GetFieldManager();
```

- Make it use your field:

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

- Create a **`G4ChordFinder`**
  - Let the field manager create it (with default parameters)

```
globalFieldM->CreateChordFinder(magField);
```

  - Explicitly create it

```
G4ChordFinder myChordFinder= new G4ChordFinder( … );
globalFieldM->SetChordFinder(myChordFinder);
```

Creating a ChordFinder: detail

- Create a **G4ChordFinder**

```
G4FieldManager *pFieldMgr= … ; // Get or create it
G4ChordFinder  *pChordFinder=0;
G4MagIntegratorStepper *pStepper;
pStepper = … ;

pChordFinder = new G4ChordFinder(  pField,
                                          1.0e-2 * mm,   // Min
   step
                                          pStepper );

pFieldMgr->SetChordFinder( pChordFinder );
```

Local Fields

- One field manager is associated with the 'world'
- Other volumes/regions in the geometry 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
  - The assigned field is propagated to all new daughter volumes.

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

Using '`true`' makes it *push* the field to all existing daughter volumes (and their daughters and so on) – unless a daughter has its own field manager.

Customizing field integration

- Trying a few different types of steppers for a particular field or application is suggested if maximum performance is a goal
- Specialized steppers for pure magnetic fields are also available
  - They take into account the fact that a local trajectory in a slowly varying field will not vary significantly from a helix
  - Combining this in with a variation, the Runge-Kutta method can provide higher accuracy at lower computational cost when large steps are possible
- To change the stepper:

```
theChordFinder
   ->GetIntegrationDriver()
   ->RenewStepperAndAdjust( newStepper );
```

Creating a Stepper: Example

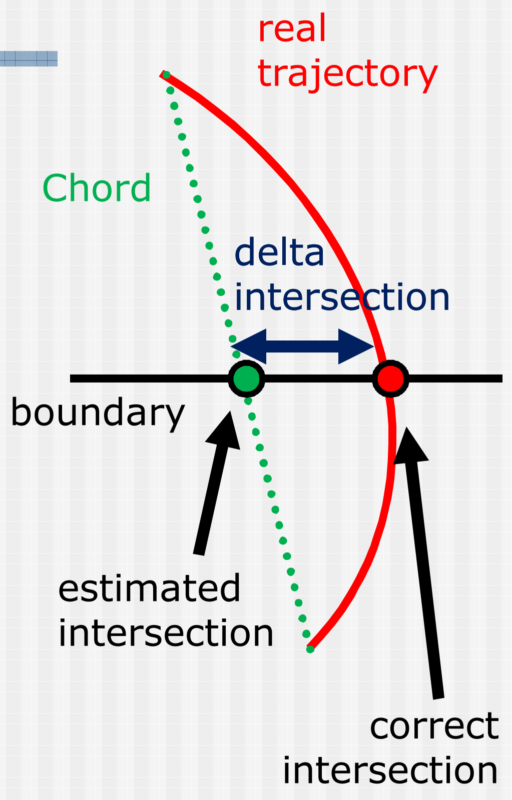
```
#include "G4SimpleRunge.hh"
…
G4Mag_UsualEqRhs *fEquation = new
   G4Mag_UsualEqRhs(&myMagField);
G4MagIntegratorStepper *pStepper;
// Can choose one of the following Steppers
pStepper = new G4SimpleRunge( fEquation );   // 2nd
pStepper = new G4SimpleHeum( fEquation );    // 3rd
pStepper = new G4ClassicalRK4( fEquation ); // 4th
pStepper = new G4HelixExplicitEuler( fEquation );
pStepper = new G4CashKarpRKF45( fEquation );
pStepper = new G4NystromRK4( fEquation );    // New!
```

Accuracy and performance

- You can customise the propagation to get
  - Higher accuracy for key particles, or
  - Faster – fewer CPU cycles
- How to tailor it to your needs:
  - Choose a stepper for the field
  - Set precision parameters

Tunable Parameters

- In addition to the "miss distance" there are two more parameters which can be set in order to adjust the accuracy (and performance) of tracking in a field
  - Such 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 the 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, one can limit the effect of this bias



Tunable Parameters

- The "MaximumEpsilonStep" parameter is the relative accuracy for the endpoint of 'ordinary' integration steps, those which do not intersect a volume boundary
  - Limits estimated error $|\Delta x|$ of endpoint of each physics step (of length `len`):

    $|\Delta x| < \varepsilon$ * `len`
- Parameters can be set by:

```
myFieldManager->SetMaximumEpsilonStep( eps_max );
myFieldManager->SetMaximumEpsilonStep ( eps_min );
myFieldManager->SetDeltaIntersection ( delta_intersection );
theChordFinder->SetDeltaChord ( miss_distance );
```

Imprecisions …

- … are due to approximating the curved path by linear sections (chords)
  - Parameter to limit this is maximum sagitta δ_{chord}
- … are due to numerical integration, 'error' in final position and momentum
  - Parameters to limit are ε_{max}, ε_{min}
- … are due to intersecting approximate path with the volume boundary
  - Parameter is $\delta_{intersection}$

Key elements

- Precision of track required by the user relates primarily to:
  - The precision (error in position) e_{pos} after a particle has undertaken track length s
  - Precision DE in final energy (momentum) $\delta_E = \Delta E/E$
  - Expected maximum number N_{int} of integration steps
- Recipe for parameters:
  - Set $\varepsilon_{\text{integration (min, max)}}$ smaller than
    - The minimum ratio of e_{pos} / s along particle's trajectory
    - δ_E / N_{int} the relative error per integration step (in E/p)
  - Choosing how to set δ_{chord} is less well-defined. One possible choice is driven by the typical size of the geometry (size of smallest volume)

Where to find the parameters …

Parameter	Name	Class	Default value
δ_{miss}	`DeltaChord`	`G4ChordFinder`	`0.25 mm`
d_{min}	`stepMinimum`	`G4ChordFinder`	`0.01 mm`
$\delta_{intersection}$	`DeltaIntersection`	`G4FieldManager`	`1 micron`
ε_{max}	`epsilonMax`	`G4FieldManager`	`0.001`
ε_{min}	`epsilonMin`	`G4FieldManager`	$5\ 10^{-5}$
~~$\delta_{one\ step}$~~	~~`DeltaOneStep`~~	`G4FieldManager`	`0.01 mm`

Other types of field

- It is possible to create any specialised type of field:
  - inheriting from **`G4VField`**
  - Associating an *Equation of Motion* class (inheriting from **`G4EqRhs`**) to simulate other types of fields
  - Fields can be time-dependent
- For pure electric field:
  - **`G4ElectricField`** and **`G4UniformElectricField`** classes
- For combined electromagnetic field:
  - **`G4ElectroMagneticField`** class
- The *Equation of Motion* class for electromagnetic field is **`G4MagElectricField`**.

Example code for E field

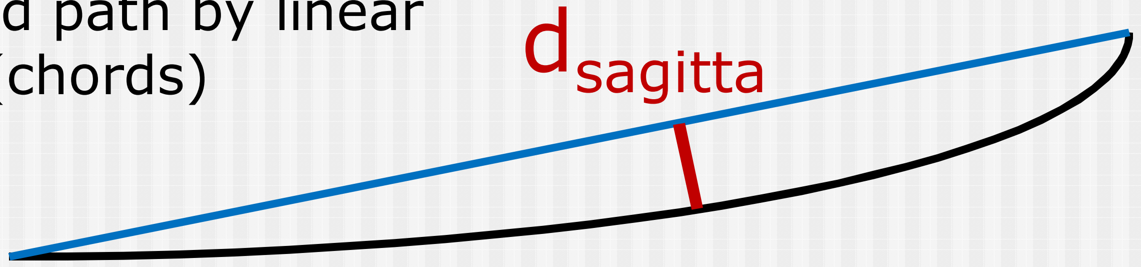
```
G4ElectricField* fEMfield
   = new G4UniformElectricField( G4ThreeVector(0.,
   100000.*kilovolt/cm, 0.) );
G4EqMagElectricField* fEquation = new G4EqMagElectricField(fEMfield);

G4int nvar= 8; // Integrate position(3), momentum(3), energy, time
G4MagIntegratorStepper* fStepper = new G4ClassicalRK4( fEquation, nvar
   );
G4FieldManager* fFieldMgr
   = G4TransportationManager::GetTransportationManager()->
   GetFieldManager();
fFieldManager->SetDetectorField( fEMfield );

G4MagInt_Driver* fIntgrDriver
   = new G4MagInt_Driver(fMinStep, fStepper, fStepper-
   >GetNumberOfVariables() );
G4ChordFinder* fChordFinder = new G4ChordFinder(fIntgrDriver);
```

DETAILS

Volume miss error

- Due to the approximation of the curved path by linear sections (chords)

$d_{sagitta} < \delta_{chord}$



Parameter δ_{chord} value

- Parameter δ_{chord} = maximum sagitta
- Effect of this parameter as $\delta_{chord} \longrightarrow 0$

$$s_{1step}^{propagator} \sim (8\,\delta_{chord}\,R_{curv})^{1/2}$$

so long as $s^{propagator} \Leftarrow s^{phys}$ and $s^{propagator} > d_{min}$(integr)

Integration error

Due to error in the numerical integration (of equations of motion)

Parameter(s): $\varepsilon_{integration}$

- The size *s* of the step is limited so that the estimated errors of the final position Δr and momentum Δp are both small enough:

$$\max\left(\| \Delta \vec{r} \| / s \, , \ \|\Delta \vec{p}\| / \|\vec{p}\| \right) < \varepsilon_{integration}$$

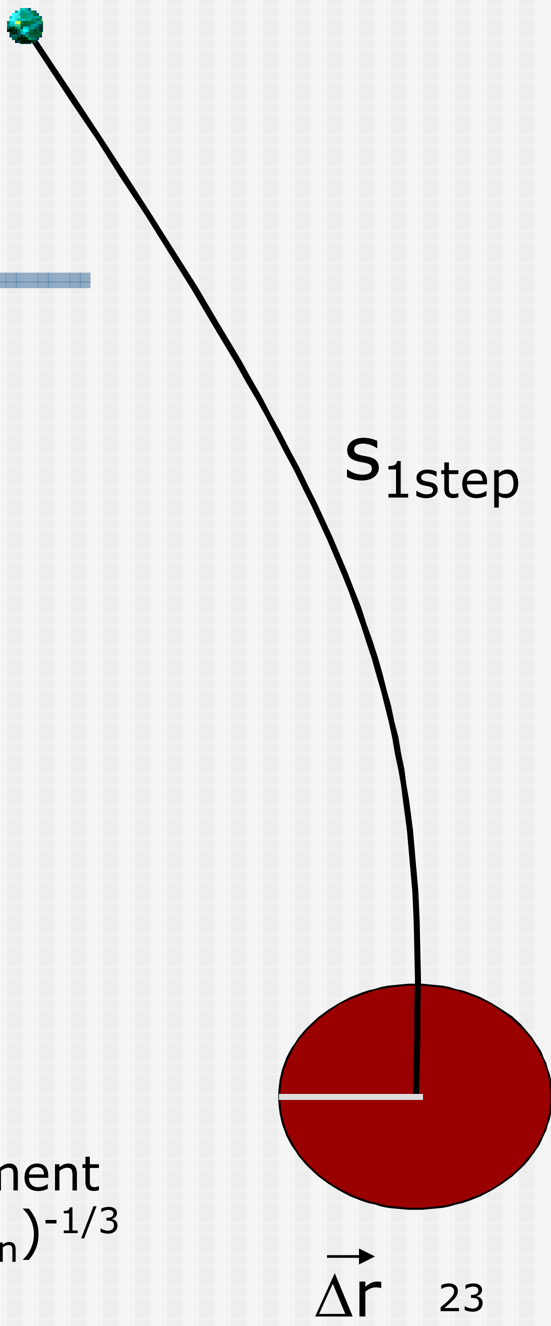
- For ClassicalRK4 Stepper

$$s_{1step}^{integration} \sim (\varepsilon_{integration})^{1/3}$$

  for small enough $\varepsilon_{integration}$

- The integration error should be influenced by the precision of the knowledge of the field (measurement or modeling).

$$N_{steps} \sim (\varepsilon_{integration})^{-1/3}$$



Integration error - 2

- $\varepsilon_{\text{integration}}$ is currently represented by 3 parameters
  - **`epsilonMin`**, a minimum value (used for big steps)
  - **`epsilonMax`**, a maximum value (used for small steps)
  - **`DeltaOneStep`**, a distance error (for intermediate steps)

	Defaults
`epsilonMin`	*$0.5*10^{-5}$*
`epsilonMax`	*0.001*
`DeltaOneStep`	*0.01 mm*

$$\varepsilon_{\text{integration}} = \delta_{\text{one step}} / \mathrm{S}_{\text{physics}}$$

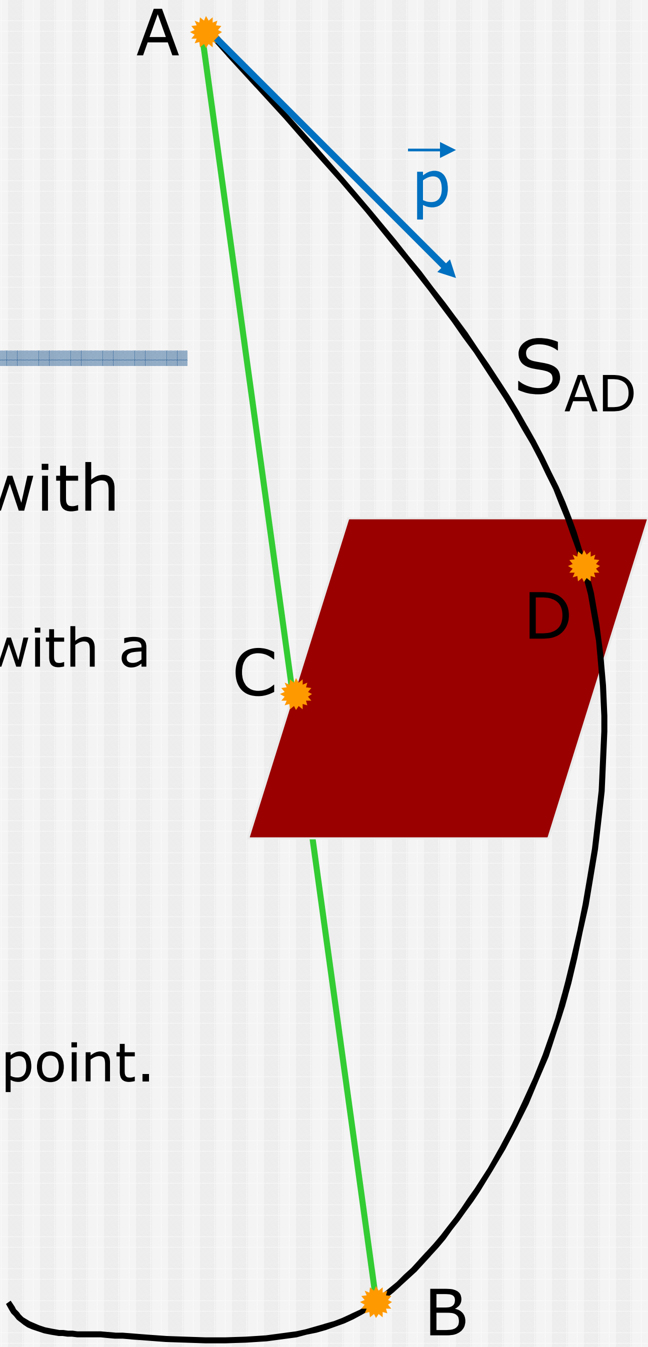
- Determining a reasonable value
  - Suggested to be the minimum of the ratio (accuracy/distance) between sensitive components, …
- Another parameter
  - d_{min} is the minimum step of integration

	Default
d_{min}	*0.01 mm*

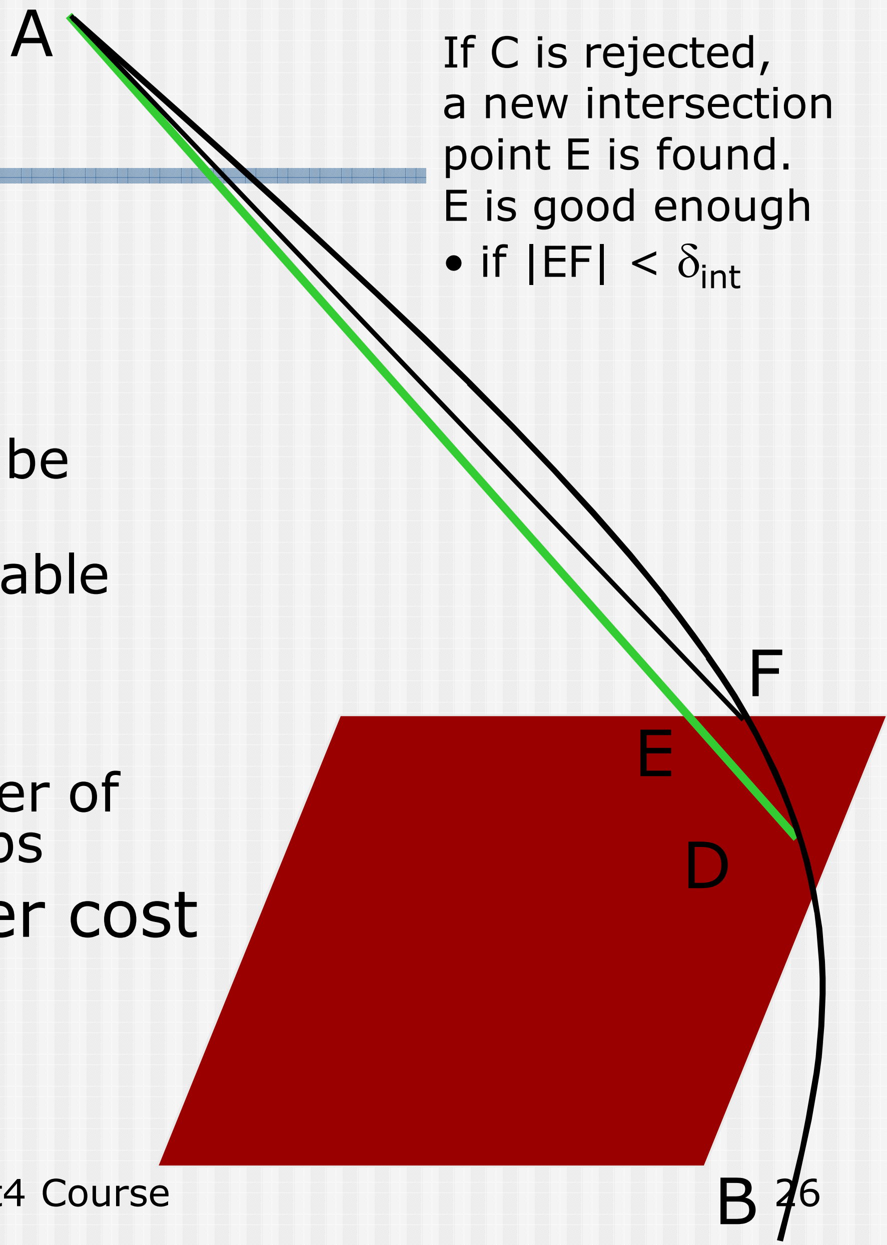
Intersection error

- In intersecting approximate path with volume boundary
  - In trial step AB, intersection is found with a volume at C
  - Step is broken up, choosing D, so

    $S_{AD} = S_{AB} * |AC| \;/\; |AB|$
  - If $|CD| < \delta_{\text{intersection}}$
    - Then C is accepted as intersection point.
  - So δ_{int} is a position error/bias



Intersection error - 2



If C is rejected, a new intersection point E is found. E is good enough

- if |EF| < δ_{int}

- δ_{int} must be small
  - compared to tracker hit error
  - its effect on reconstructed momentum estimates should be calculated
    - … and limited to be acceptable
- Cost of small δ_{int} is less
  - than making δ_{chord} small
  - it is proportional to the number of boundary crossings – not steps
- Quicker convergence / lower cost
  - Possible with optimization

Customizing field integration

- **Runge-Kutta** integration is used to compute the motion of a charged track in a general field. There are many general steppers from which to choose
  - Low and high order, and specialized steppers for pure magnetic fields
- By default, Geant4 uses the classical fourth-order **Runge-Kutta** stepper (`G4ClassicalRK4`), which is general purpose and robust.
  - If the field is known to have specific properties, lower or higher order steppers can be used to obtain the results of same quality using fewer computing cycles

Steppers for ‘rough’ fields

- If the field is calculated from a field map using a linear interpolation, a lower order stepper is recommended
  - The less smooth the field is, the lower the order of the stepper that should be used
  - The choice of lower order steppers includes the third order stepper (`G4SimpleHeum`) the second order (`G4ImplicitEuler` and `G4SimpleRunge`), and the first order (`G4ExplicitEuler`)
    - A first order stepper is hardly ever useful – potentially only for very rough fields
    - For most field approximations, the choice of order, e.g. between second and third order steppers, should be made by trial and error.