Field propagation in Geant4

John Apostolakis, CERN

Ecole Geant4 2007
7 June 2007, Paris

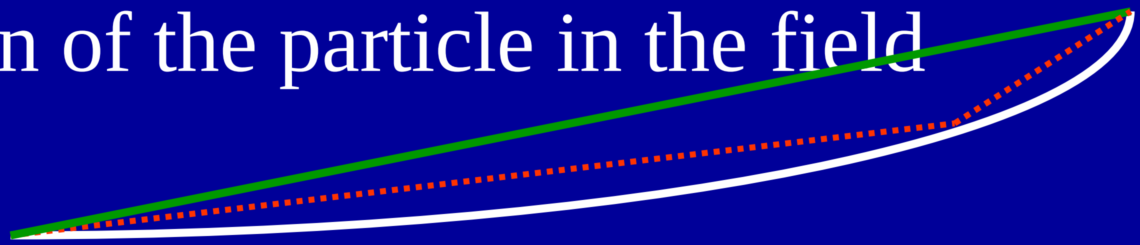
25th May 2005

Ver .ε

Contents

Magnetic field: overview

- To propagate a particle in a field (e.g. magnetic, electric or other), we solve the equation of motion of the particle in the field



- Using this solution we break up this curved path into linear chord segments
  - We determine the chord segments so that they closely approximate the curved path.
  - each chord segment will be ‘intersected’ so see it crosses a volume boundary.

Magnetic field: a first example

Part 1/2

Create your Magnetic field class

– Uniform field :

- Use an object of the G4UniformMagField class

```
#include "G4UniformMagField.hh"
#include "G4FieldManager.hh"
#include  "G4TransportationManager.hh“

G4MagneticField* magField= new
  G4UniformMagField( G4ThreeVector(1.0*Tesla, 0.0,
  0.0 ) );
```

– Non-uniform field :

- Create your own concrete class derived from G4MagneticField (see eg ExN04Field in novice example N04)

Magnetic field: a first example

Set your field as the ‘global’ field

Part 2/2

- Find the global Field Manager

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

- Set the field for this FieldManager,

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

- and create a Chord Finder.

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

In practice: exampleN03

From ExN03DetectorConstruction.cc,

which you can find also in geant4/examples/novice/N03/src

In the class definition

```
G4UniformMagField* magfield;
```

In the method SetMagField(G4double fieldValue):

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

// create a uniform magnetic field along Z axis
    magField = new G4UniformMagField(G4ThreeVector(0.,0.,fieldValue));

// Set this field as the global field
   fieldMgr->SetDetectorField(magField);
// Prepare the propagation with default parameters and other choices.
   fieldMgr->CreateChordFinder(magField);
```

Beyond your first field

- Create your own field class
  - To describe your setup's EM field
- Global field and local fields
  - The world or detector field manager
  - An alternative field manager can be associated with any logical volume
    - Currently the field must accept position global coordinates and return field in global coordinates
- Customizing the field propagation classes
  - Choosing an appropriate stepper for your field
  - Setting precision parameters

Creating your own field

Create a class, with one key method – that calculates the value of the field at a Point

Point [0..2]	position
Point[3]	time

```
void ExN04Field::GetFieldValue(
  const double Point[4],
    double *field) const
{
  field[0] = 0.;
  field[1] = 0.;
  if(abs(Point[2])<zmax &&
  (sqr(Point[0])+sqr(Point[1]))<rmax_sq)
  { field[2] = Bz; }
  else
  { field[2] = 0.; }
}
```

Global and local fields

- One field manager is associated with the ‘world’
  - Set in G4TransportationManager
- Other volumes can override this
  - By associating a field manager with any logical volume
    - By default this is propagated to all its daughter volumes

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

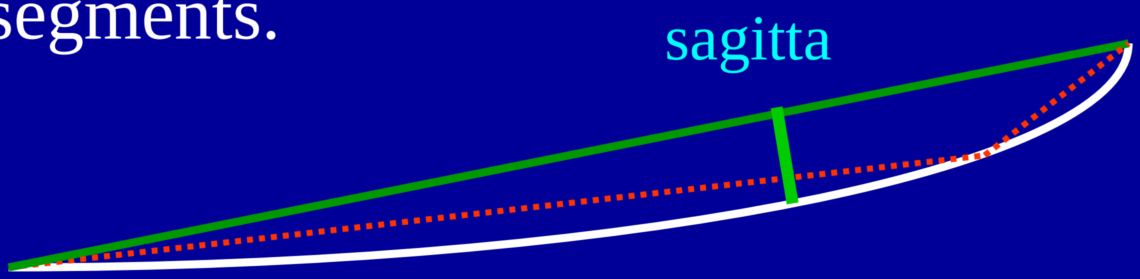
where ‘`true`’ makes it push the field to all the volumes it contains.

Solving the Equation of Motion

- 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.
- We use a Runge-Kutta method for the integration of the ordinary differential equations of motion.
  - Several Runge-Kutta 'steppers' are available.
- In specific cases other solvers can also be used:
  - In a uniform field, using the analytical solution.
  - In a nearly uniform field (BgsTransportation/future)
  - In a smooth but varying field, with new RK+helix.

Splitting the path into chords

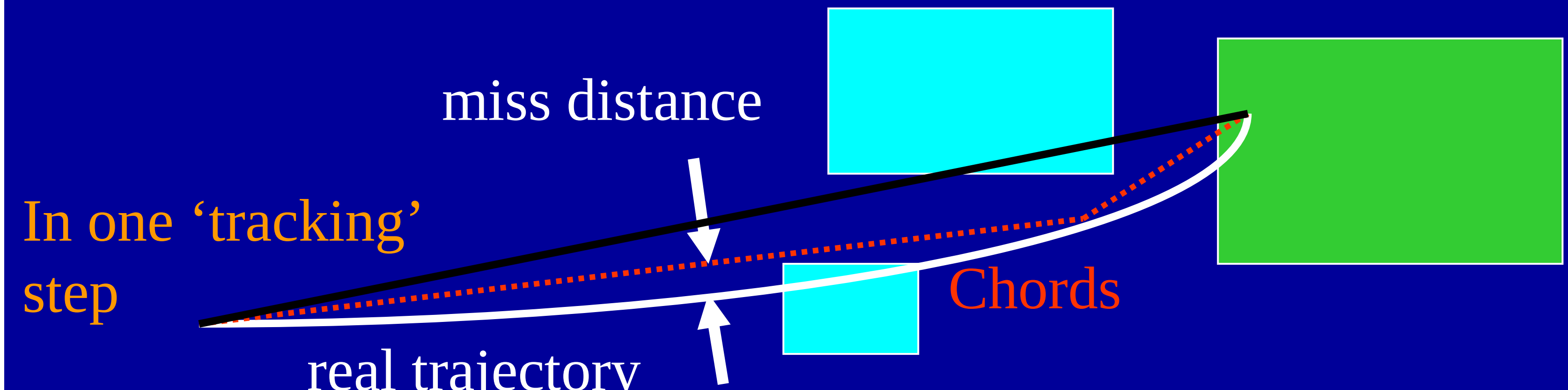
- Using the method to calculate the track's motion in a field, Geant4 breaks up this curved path into linear chord segments.



- Choose the chord segments 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.
- We use the chords to interrogate the Navigator, to see whether the track has crossed a volume boundary.

Stepping and accuracy

- You 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.
    - Default "miss distance" is 0.25 mm (used to be 3.0 mm).
- One physics/tracking step can create several chords.
  - In some cases, one step consists of several helix turns.



Precision parameters

- Errors come from
  - Break-up of curved trajectory into linear chords
  - Numerical integration of equation of motion
    - or potential approximation of the path,
  - Intersection of path with volume boundary.
- Precision parameters enable the user to limit these errors and control performance.
  - The following slides attempt to explain these parameters and their effects.

Imprecisions

- Due to approximating the curved path by linear sections (chords)
  - Parameter to limit this is maximum sagitta δ_{chord}
- Due to numerical integration, ‘error’ in final position and momentum
  - Parameters to limit are $\varepsilon_{integration}$ max, min
- Due to intersecting approximate path with 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-define. One possible choice is driven by the typical size of your geometry (size of smallest volume)

Where to find the parameters

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

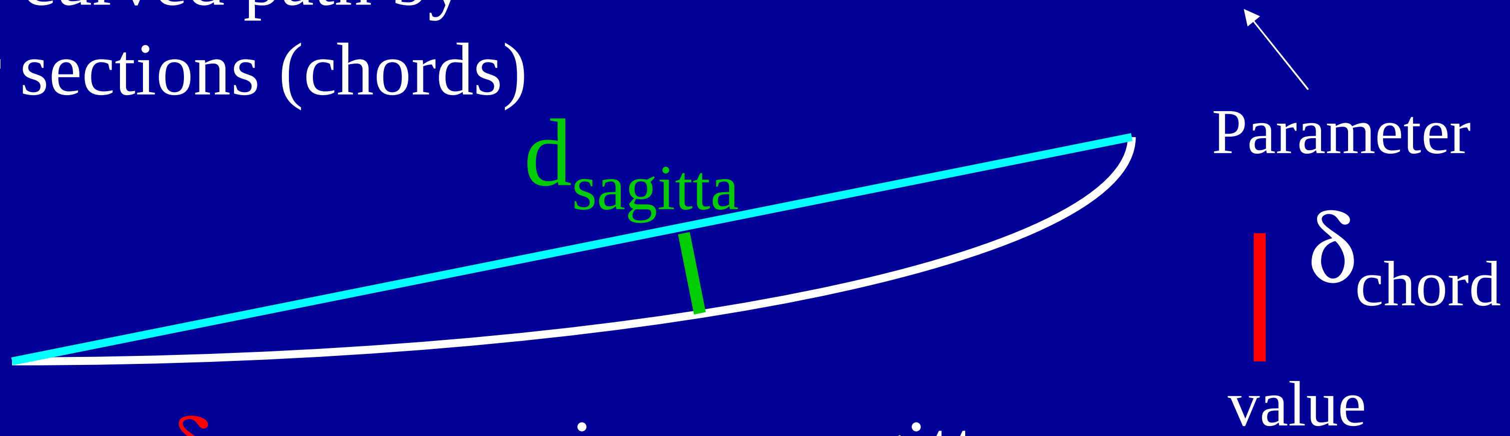
Details of Precision Parameters

For further/later use

Volume miss error

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

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



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

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

so long as $s^{propagator} \leq s^{phys}$ 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:

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

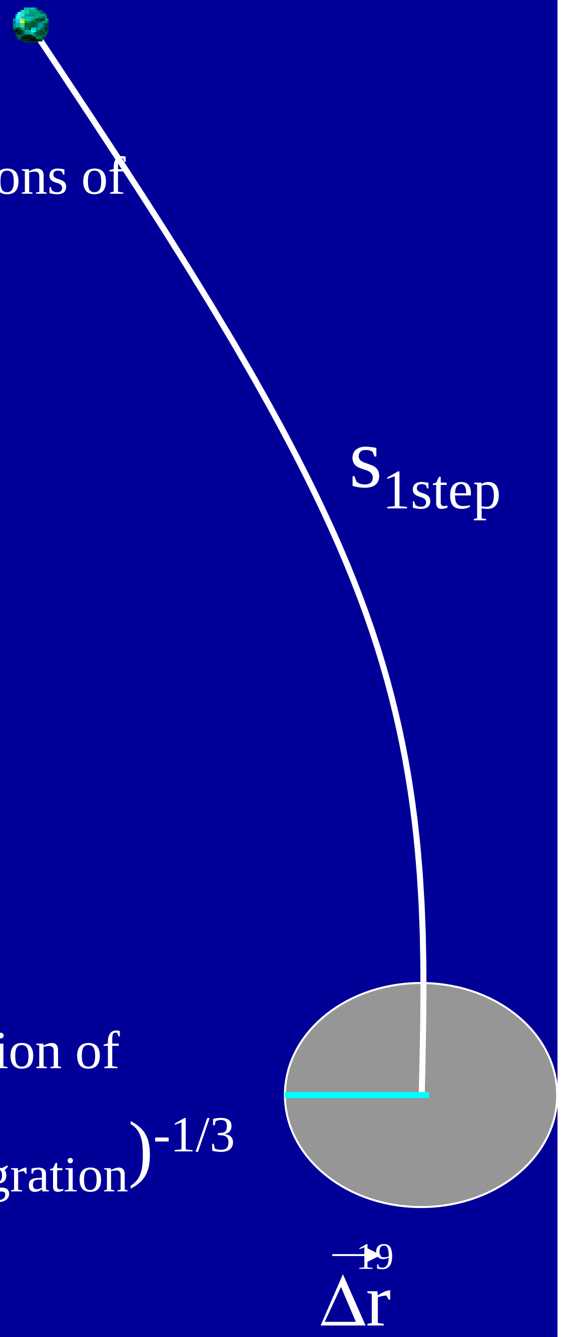
- For ClassicalRK4 Stepper

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

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 errors (cont.)

- In practice

  $\varepsilon_{integration}$ is currently represented by 3 parameters — Defaults
  - epsilonMin, a minimum value (used for big steps) — $0.5*10^{-7}$
  - epsilonMax, a maximum value (used for small steps) — 0.05
  - DeltaOneStep, a distance error (for intermediate steps) — 0.25 mm

$$\varepsilon_{integration} = \delta_{one\ step} / s_{physics}$$

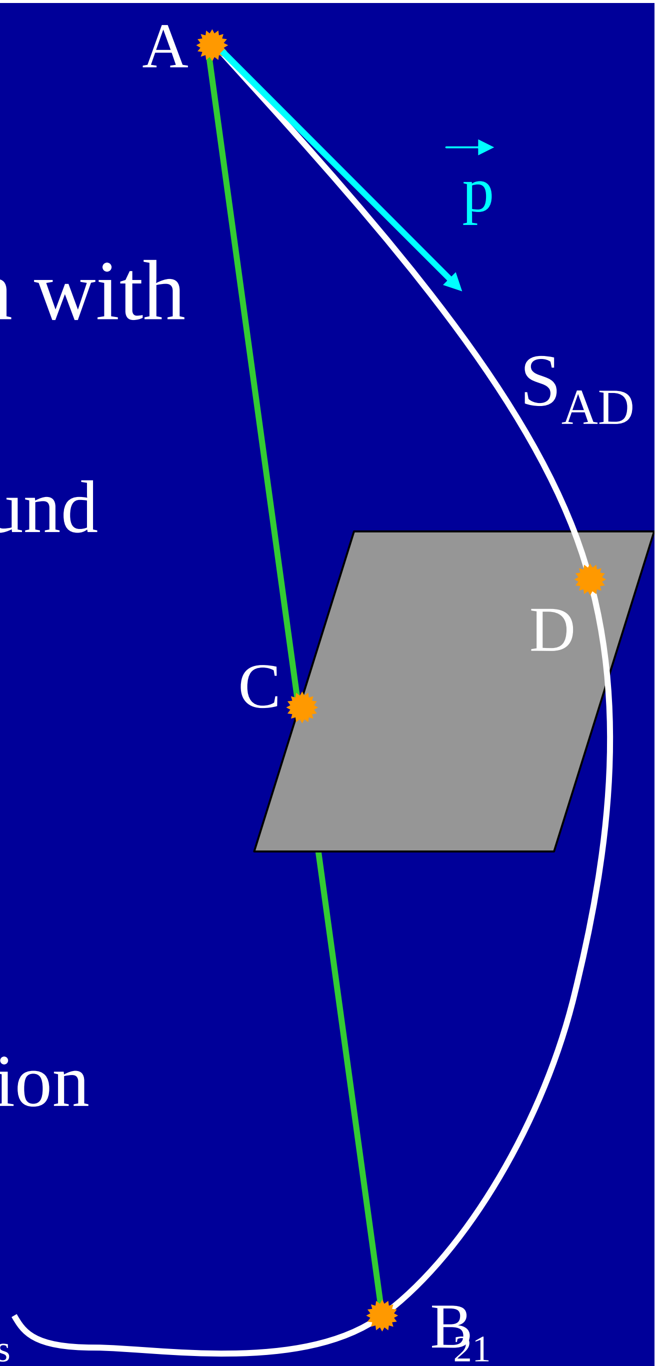
- Determining a reasonable value
  - I suggest it should be the minimum of the ratio (accuracy/distance) between sensitive components, ..
- Another parameter — Default
  - d_{min} is the minimum step of integration — 0.01 mm
    - (newly enforced in Geant4 4.0)

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

    $$S_{AD} = S_{AB} * |AC| \ / \ |AB|$$

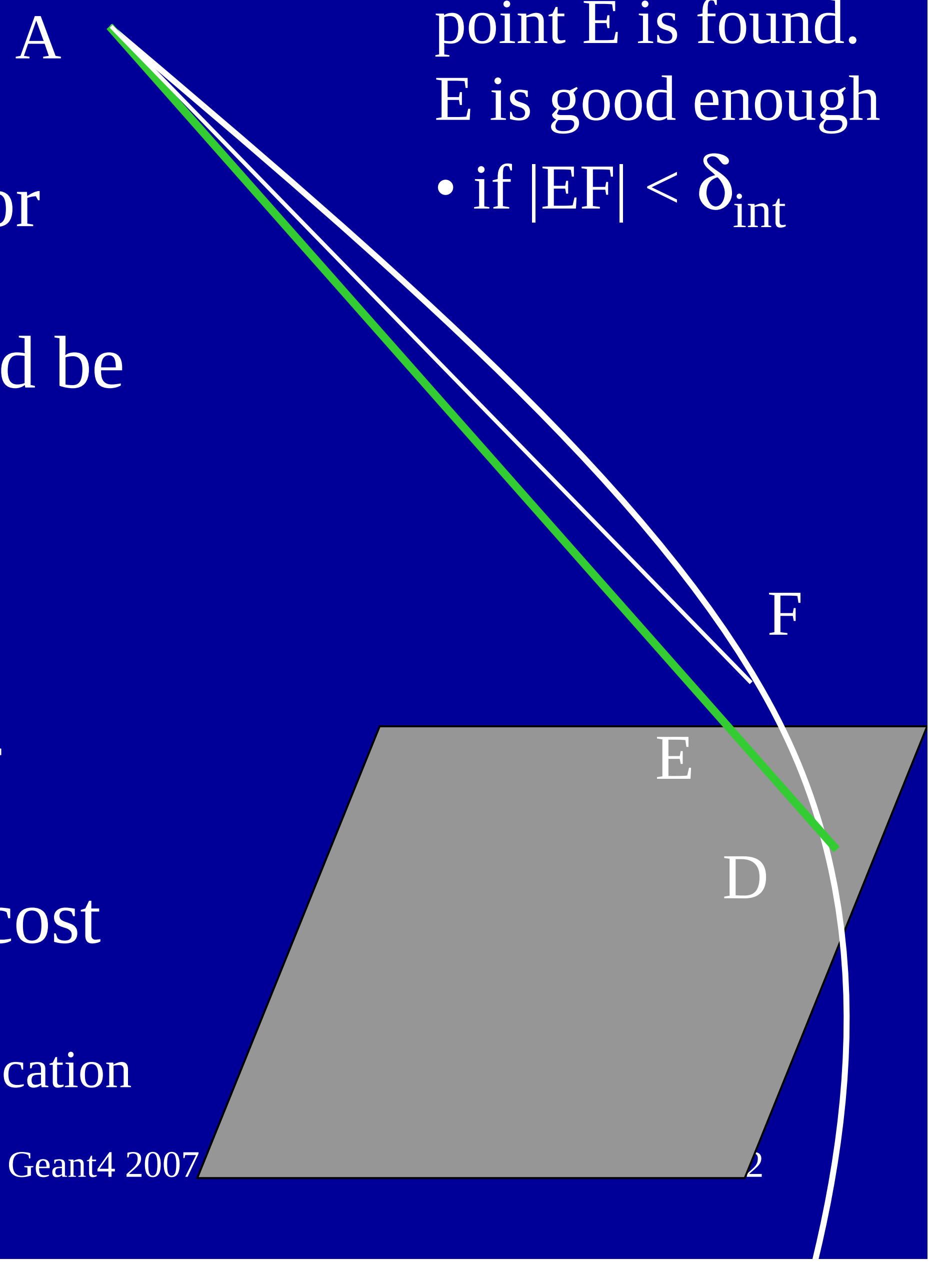
  - If $|CD| < \delta_{intersection}$
    - Then C is accepted as intersection point.
  - So δ_{int} is a position error/bias



Intersection error (cont)

- So 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
  - Is proportional to the number of boundary crossings – not steps.
- Quicker convergence / lower cost
  - Possible with optimization
    - adding std algorithm, as in BgsLocation

If C is rejected,
a new intersection
point E is found.
E is good enough
- if $|EF| < \delta_{int}$



The ‘driving force’

- Distinguish cases according to the factor driving the tracking step length
  - ‘physics’, eg in dense materials
  - fine-grain geometry
- Distinguish the factor driving the propagator step length (if different)
  - Need for accuracy in ‘seeing’ volume
  - Integration inaccuracy
    - Strongly varying field

Potential Influence

G4 Safety improvement

Other Steppers, tuning d_{min}

What if time does not change much?

- If adjusting these parameters (together) by a significant factor (10 to 100) does not produce results,
  - Then field propagation may not the dominant (most CPU intensive) part of your program.
  - Look into alternative measures
    - modifying the physics 'cuts' – ie production thresholds
      - To create fewer secondaries, and so track fewer particles
    - determining the number of steps of neutral vs charged particles,
      - To find whether neutrons, gammas 'dominate'
    - profiling your application
      - You can compile using G4PROFILE=yes, run your program and then use "gprof" to get an execution profile.

Contributors to Field sub-category

Current Contributors

- John Apostolakis
- Tatiana Nikitina
- Vladimir Grichine

Past contributors

- Simone Giani
- Wolfgang Wander

With thanks to users contributing significant feedback

- including Pedro Arce, Alberto Ribon, Stefano Magni, …

and to David C. Williams for feedback & discussions