GEANT4E:
Error propagation for track reconstruction inside the GEANT4 framework

Pedro Arce (CIEMAT)

CHEP 2006, Mumbai, 13-17th February 2006

Outlook

- What is GEANT4E
- GEANT4E components
  - Trajectory state
  - Target to propagate
  - Track propagator manager
  - Physics
  - Magnetic field
  - Track error propagation
- Comments on backwards tracking
- GEANT4E example
  - Results comparison with GEANE
  - CPU time comparison with GEANE
- How to make GEANT4E faster?
- Summary and plans

What is GEANT4E

- Track reconstruction needs to match signals in two detector parts
  - Propagate tracks from one detector part to another and compare with real measurement there
  - Make the average between the prediction and the real measurement

  $\Rightarrow$ it needs the track parameter errors

- Many experiments have used in the past GEANE (based on GEANT3) or their 'ad hoc' solution

GEANT4e provides this functionality for the reconstruction software in the context of GEANT4

Trajectory state: G4eTrajState

- User defines the initial track parameters in a given point of the trajectory: **G4eTrajState**
  - Particle type
  - Position
  - Momentum
  - Track errors (5x5 HepSymMatrix)
  - Initial surface where parameters are defined
- Two different trajectory states:
  - G4eTrajState**Free**:
    - 1/p, λ, ϕ, y_perp, z_perp (p_x = p cos(λ) cos(ϕ), p_y = p cos(λ) sin(ϕ), p_z = p sin(λ), x_perp || trajectory, y_perp parallel to x-y plane)
  - G4eTrajState**OnSurface**: parameters on a plane in an arbitrary direction
    - 1/p, v', w', v, w (u,v,w is any orthonormal coordiante system, v, w on the plane)

End of propagation: G4eTarget

- User defines up to where the propagation must be done: the target

- G4eTargetSurface
  - Track is propagated until the surface is reached
  - The surface is not part of GEANT4 geometry
    - Using a ghost geometry would mean that propagation in field is done twice
    - **G4eNavigator** takes care of the double navigation: on the full geometry and checking if surface is reached
      - overwrites ComputeStep() and ComputeSafety() to stop the navigation when the surface si reached
  - Several types defined
    - G4eTargetPlaneSurface: infinite plane
    - G4eTargetCylindrycalSurface: infinite length cylindrical surface
    - ....
- G4eTargetTrackLength
  - Track is propagated until a certain track length is reached
  - Implemented as a G4VDiscreteProcess

End of propagation: G4eTarget (2)

- G4eTargetVolumeG4
  - Track is propagated until the surface of a GEANT4 volume
    - Track enters
    - or track exits
    - or both
  - User can choose if volume refers to one or many G4LogicalVolume's, G4VPhysicalVolume's or G4VTouchable's, with a simple syntax:
    - `G4eTargetVolumeG4("MuonCell")` G4LogicalVolume
    - `G4eTargetVolumeG4("MuonCell#1")` G4VPhysicalVolume
    - `G4eTargetVolumeG4("MuonChamber#3/MuonCell#2")` G4VTouchable
- G4eTargetVolumeUserDefined (TO BE DONE)
  - Track is propagated until the surface of a user-defined volume (outside the GEANT4 geometry)

Managing tracks: G4ePropagator

- User needs to propagate just one track

  ⇒ no need of run and events

- ✓ G4ePropagator creates a track and manages the step propagation
  - Creates a G4Track from the information given in the G4eTrajState
  - Invokes G4SteppingManager to propagate one step

    ```
    fpSteppingManager::Stepping();
    ```

  - And propagates the track errors for this step

    ```
    G4ePropagator::PropagateError( aTrack );
    ```

  - Stops when G4Track stops or when the target is reached
    - If defined target is not reached it returns an error

- ❖ User can choose two ways of propagetion
  - Propagate until target is reached
  - Propagate step by step and return control after each step

Physics: G4ePhysics

- Reconstruction software wants the average trajectory followed by the particle: G4ePhysicsList
  - No multiple scattering
  - No secondaries allowed
  - No random fluctuations for energy loss
  - No hadronic processes
  - Huge cuts by default (User can change them with standard GEANT4 methods)
  - Negative energy loss when propagation is backwards
    - G4e/mu/hIonisation redefined through a templated class: G4ePhysicalProcesses<class Tionisation>

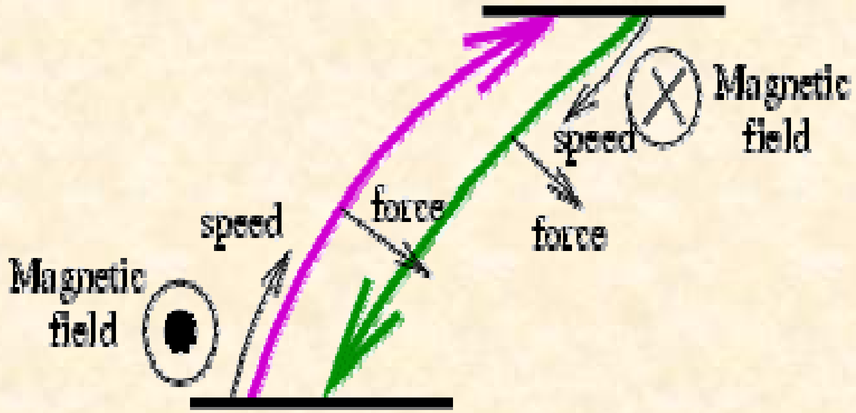
- User could define its own physics list (simply add it to the G4RunManager)
  - But it should account for backwards tracking

- Simple energy loss can be chosen: faster but less precise
  - But time in calculating energy loss is ~1 %

Magnetic field: G4eMagneticField

- User defines the magnetic field in the standard GEANT4 way
- But GEANT4e has to handle the backwards propagation

  ⇒ Magnetic field has to be reversed



- G4ePropagatorG4 takes care of replacing G4Mag_UsualEqRhs by G4eMag_UsualEqRhs, that overwrites EvaluateRhsGivenB() to reverse the field

Track error propagation

- ❖ Based on the equations of the European Muon Collaboration (same as GEANE)
  - ✓ Error from curved trajectory in magnetic field
  - ✓ Error from multiple scattering
  - ✓ Error from ionisation

- ➢ Formulas assume propagation along an helix
  - Need to make small steps to assure magnetic field constantness and not too big energy loss ⇒ makes it slower

- ❑ Another approach to be studied: propagate the error together with the solving of the Runge-Kutta equations
  - Probably slower per step but might not need so many steps

E loss in backwards tracking

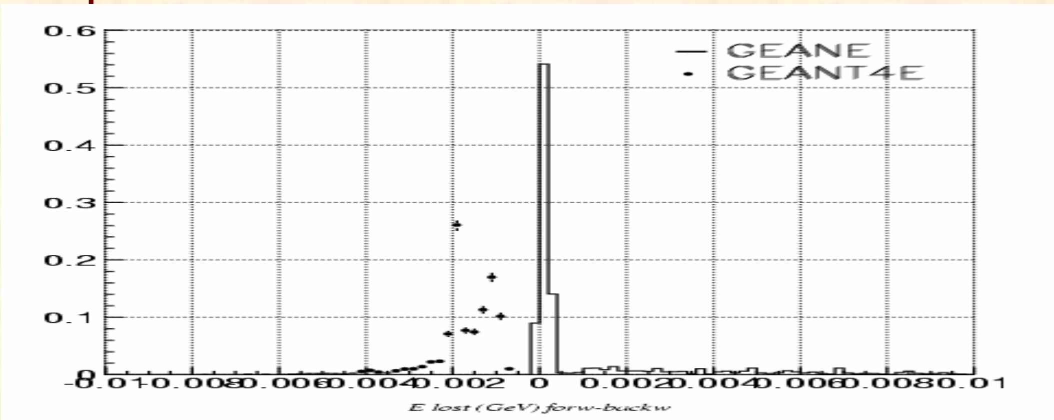
- When reconstruction software wants to know the trajectory that a track has described from a detector part to another, often the track has to be propagated **backwards**
  - ✓ The track has to gain energy instead of losing it
  - ✓ The value of the magnetic field has to be reversed

- But the energy lost (or gained) in one step is calculated
  - o Forward tracking: using the energy at the beginning of the step
  - o Backward tracking: using **the energy at the end of the step**
  - ❑ And similarly for the curvature in magnetic field

E loss in backward tracking (2)

Geant 4 Ciemat

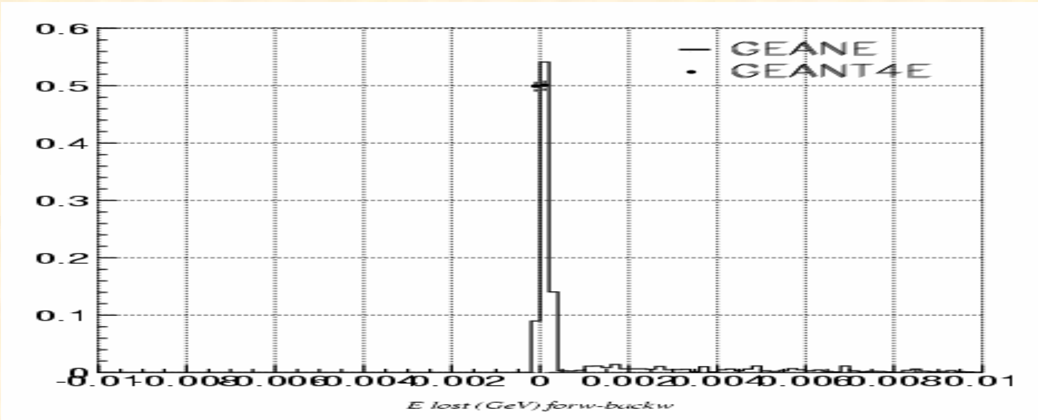
☹ This means that if you propagate a particle forwards and then backwards it would not recover the original energy



Difference in energy when a 20 GeV track is propagated forwards and then backwards

NO CORRECTION

☺ A correction is applied: dEdx is calculated with the energy at the end of step, then half of this energy is added and dEdx is recalculated again



Difference in energy when a 20 GeV track is propagated forwards and then backwards

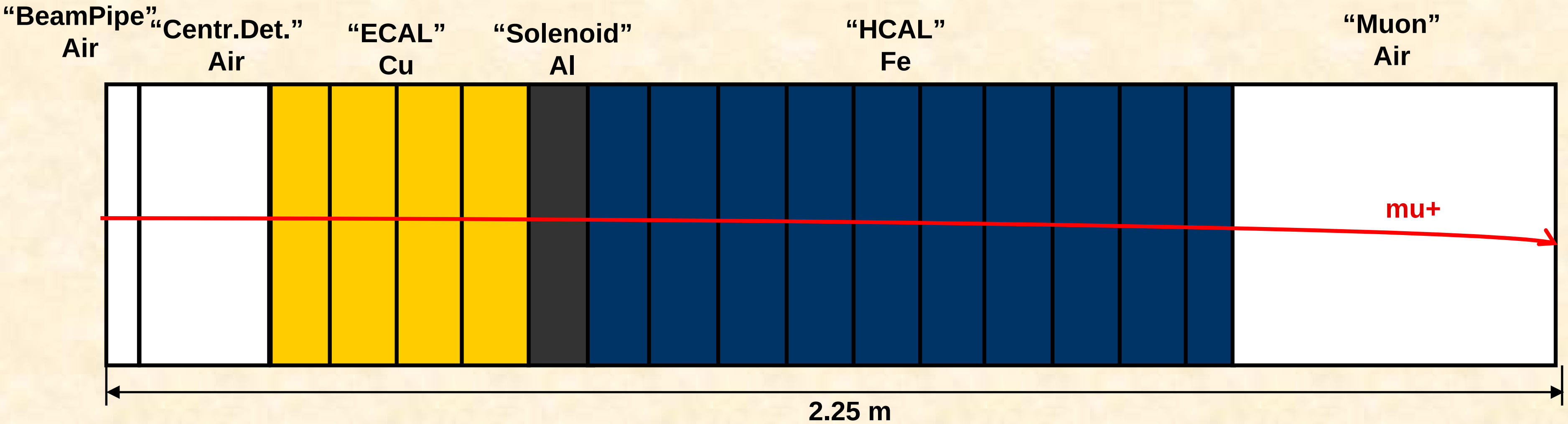
CORRECTED

- Something similar should be done for the propagation in magnetic field (under discussion with GEANT4 experts)

GEANTE example

Same example is implemented in GEANE and GEANT4E

- Simple detector:



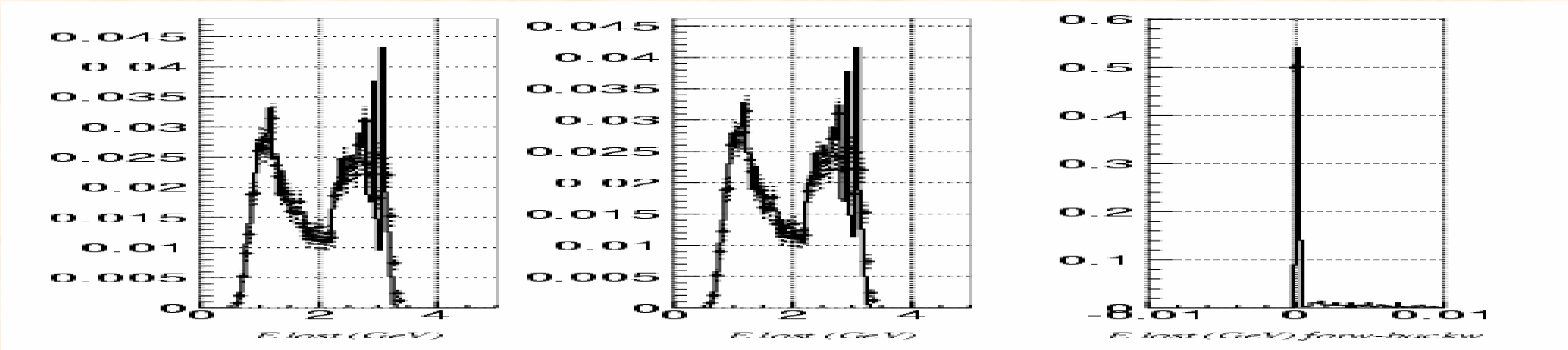
- Magnetic field 10 kGauss (0.1 Tesla)
- A track is propagated from the origin along all detectors until a plane surface, and then from the end point it is propagated backwards
  - Several variables are compared GEANE vs GEANT4E
    - Energy lost, deviation in position and angle for forward, backward and forward+backward tracking (this last one should be 0)
    - Trajectory errors for forward and backward

Comparison with GEANE

10000 mu+: 5-100 GeV, along X ± 10 degrees:

- Energy lost (GeV)

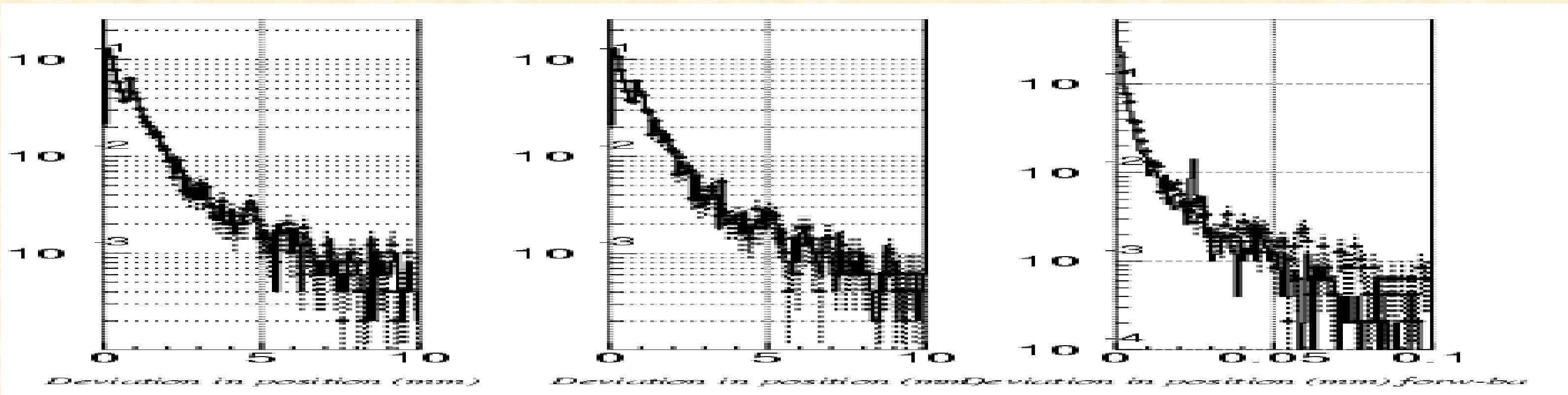
forwards **backwards** **forwards+backwards**



- GEANE
• GEANT4E

- Deviation in position (mm)

forwards **backwards** **forwards+backwards**



- GEANE
• GEANT4E

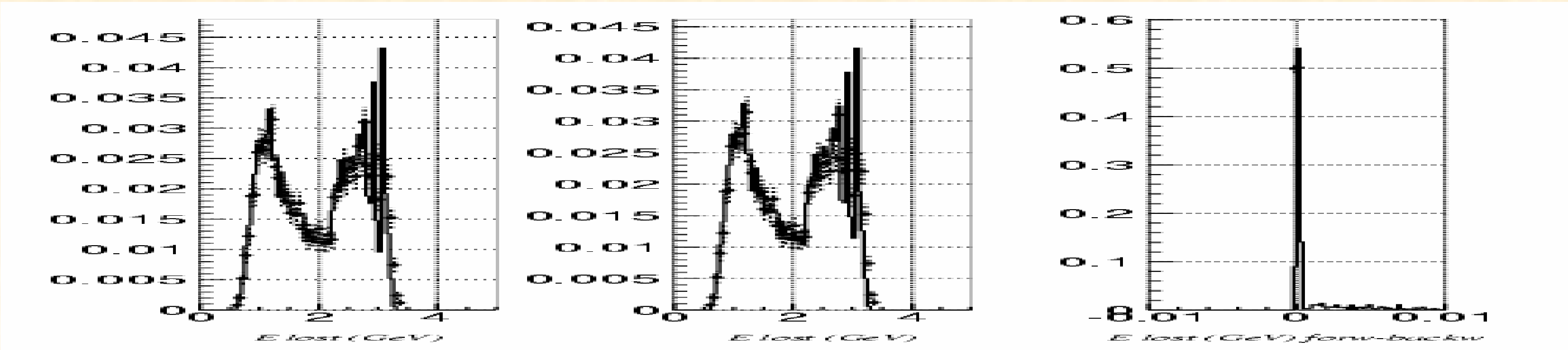
Not the same because GEANT4 propagation is more precise

Comparison with GEANE (2)

10000 mu+: 5-100 GeV, along X ± 10 degrees:

- Deviation in angle (mrad)

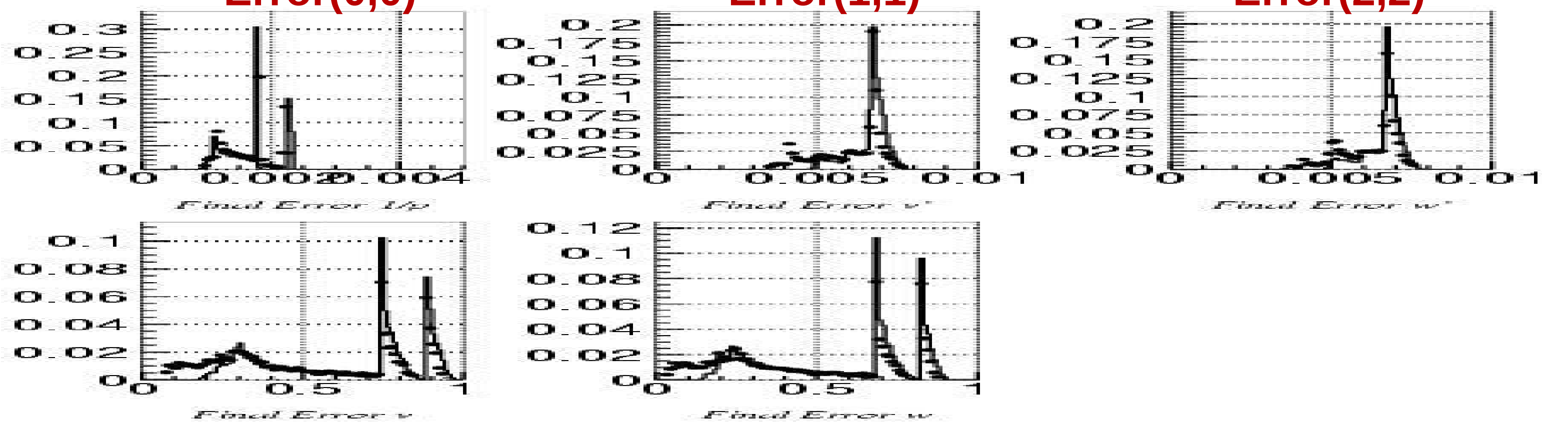
forwards **backwards** **forwards+backwards**



GEANE
GEANT4E

- Trajectory errors (if target is reached)

Error(0,0) **Error(1,1)** **Error(2,2)**



Error(3,3) **Error(4,4)**

- GEANE
• GEANT4E

Timing GEANE vs GEANT4E

10k mu+ 20 GeV cross all the detector (time in msec/evt CPU: Athlon 1 GHz)

- Same number of steps in GEANT3 and GEANT4

GEANT3		GEANT4	
GEANT3	0.39	GEANT4	1.22
GEANE: Forward or backward	**0.45**	**GEANT4E: Forward or backward**	**1.65**
GEANE: no error Forward or backward	0.28	GEANT4E: no error Forward or backward	1.30

- GEANT4 is 2.5 times slower than GEANT3
- GEANT4E is 3.5 times slower than GEANE
- Most of the time is taken by GEANT4 field propagation
- Error propagation is ~1/3 of total time

☺ Results have been checked by profiling

VERY PRELIMINARY:

Time in full CMS:

- GEANE: 55 msec/track
- GEANT4E: 44 msec/track

! But 3.5 X more steps in GEANE

How to make GEANT4E faster?

➢ The problem is indeed how to make it faster keeping the desired precision

Some ideas:

- Tune the step length to your desired precision
  - Define a fixed step length
  - Define the allowed variation in magnetic field
  - Define the allowed proportion of energy loss
- Tune propagation in magnetic field
  - Choose IntegratorStepper
  - Choose precision parameters
- Simplify geometry
  - Probably you do not need so much precision for reconstruction as for simulation
- Optimize the error propagation
  - Try different matrix class
- Propagate error with Runge-Kutta equations to make bigger steps

Summary and plans

- First prototype of GEANT4E is ready with similar functionality as GEANE
- Simple example shows it is 3.5 times slower as GEANE (0.8 in real detector, although with bigger steps)
- Many optimisation options available

Next steps:

- Release in GEANT4
- Check in a real detector reconstruction (CMS)
  - Baseline software for track error propagator in CMS Cosmic Challenge (May 2006)
- Try different optimisation options