Simulation Steering with DDSim

André Sailer

CERN-EP-LCD

ILD Software Workshop
February 24, 2016

Table of Contents

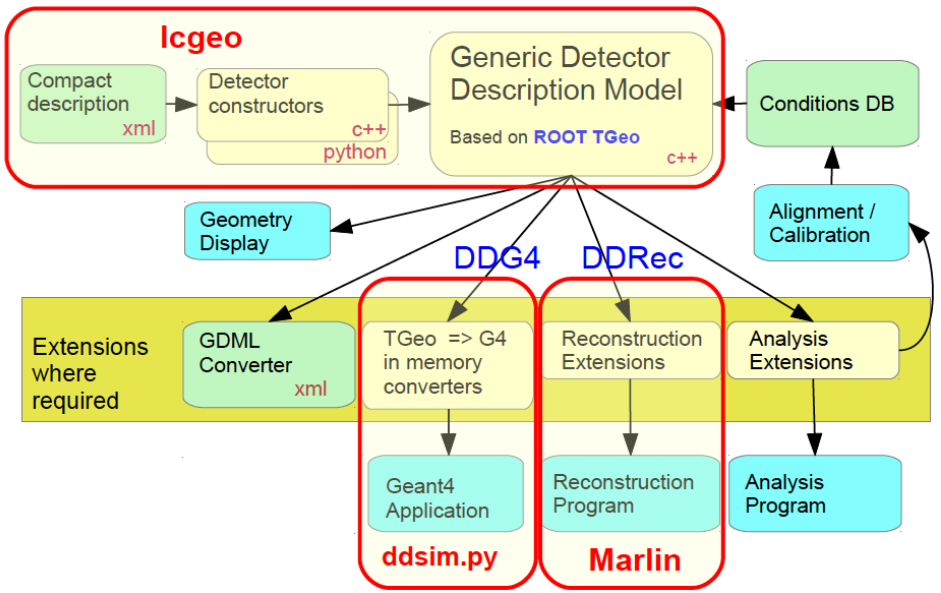
1 DD4hep and DDG4
- DDG4 Features

2 Running the Simulation

3 Conclusion

DD4hep

- DD4hep: Single source of Geometry for Simulation, Reconstruction, Analysis, . . .
- lcgeo: Collection of linear collider detector (CLICdp,ILD,SiD) and testbeam (FCal) geometry drivers
- ddsim(.py): The program to run simulation with Geant4
  - Should be easy to use for users
  - Provide the same functionality as Mokka and SLIC (and more)



Monte Carlo Generator Information and Handling

Reading files from MC Generators

- HepMC, lcio, HEPEvt, stdhep
- Skipping of events at start of file
- Treating particles unknown to Geant (e.g.: some B-Baryons with non-zero decay length)

Boost and Smearing of the vertex

- Lorentz-Boost for the crossing angle
- Smearing or moving of initial vertex position, if desired

MC-Truth Information during the simulation

- Linking hits and particles that caused them

Simulation Parameters

- PhysicsList: All physics lists from Geant4 are available
  - Also allows electromagnetic extension to be chosen
- Range-Cut: Material dependent energy threshold for production of secondaries
  - Global Range-Cut in simulation setup
  - Local Range-Cut via regions in the detector XML

Magnetic Field Configuration

Tested configurations

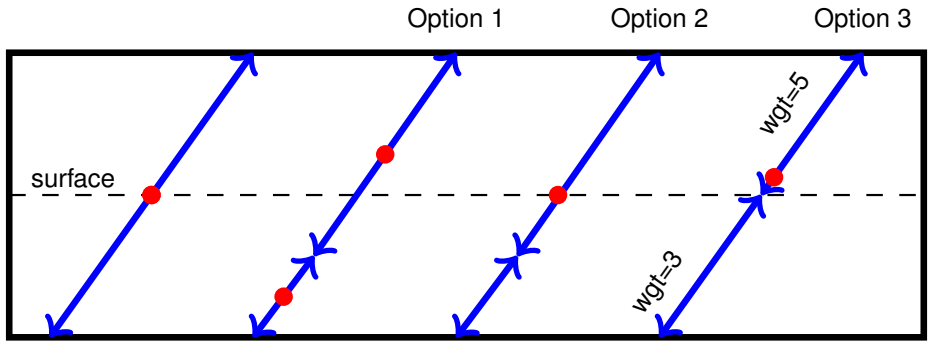
- Constant (Solenoid) Field
- 2D Field Map: B_r, B_z for given r, z

Stepping configuration

- Configuration of the propagator of particles in the field
- E.g.: Largest Acceptable Step: Default 1 km, lead to random crashes in Mokka
- Can also control type of stepper and parameters for its precision

Sensitive Detector: Tracker

- Many different options how to store the particles passing through the sensor



1. Store each *step* as a *hit*: Many hits for one particle in one volume. Each hit is placed at the average of each step
2. Sum the *steps* in the sensitive volume as a single *hit*, hit is placed at the average of all steps
3. Sum the *steps* and place the hit at the energy weighted position of the steps

- Hits need to be on the surface during reconstruction:
  - Option 2 guarantees hit on the surface
  - Option 1 and 2 need projection onto the surface
  - Option 1 needs some kind of hit merging during digitisation

ddsim

ddsim python executable part of lcgeo and the latest iLCsoft release

- To get started: `ddsim --help`
- Get a steering file `ddsim --dumpSteeringFile > mySteer.py`
  - Steering file includes documentation for parameters and examples
  - This is an actual python file, and will be interpreted as such
  - It has to contain a `DD4hepSimulation` object at global scope

```
from DDSim.DD4hepSimulation import DD4hepSimulation
from SystemOfUnits import mm, GeV, MeV, keV

SIM = DD4hepSimulation()
SIM.compactFile = "CLIC_o3_v04.xml"
SIM.runType = "batch"
SIM.numberOfEvents = 2
SIM.inputFile = "electrons.HEPEvt"

SIM.part.minimalKineticEnergy = 1*MeV

SIM.filter.filters['edep3kev'] =
  dict(name="EnergyDepositMinimumCut/3keV",
       parameter={"Cut": 3.0*keV} )
```

Command Line Options

Most options can be defined in the steering file and overwritten on the command line

- `ddsim --steeringFile mySteer.py --numberOfEvents 10`
- dumping a steering file when options are defined will change the values in the dumped steering file
- Most examples in the next slides are part of the documentation in the dumped steering file

Tab Completion

- For automatic completion of command line parameters: bash **bash-completion**, python: argcomplete `eval "$(register-python-argcomplete ddsim)"`
- Much more convenient interactive use

```
$ ddsim
--action.calo                --filter.tracker             --part.keepAllParticles
--action.mapActions          -G                           --part.minimalKineticEnergy
--action.tracker             --gun.direction              --part.printEndTracking
--compactFile                --gun.energy                 --part.printStartTracking
--crossingAngleBoost         --gun.isotrop                --part.saveProcesses
--dump                       --gun.multiplicity           --physics.decays
--dumpParameter              --gun.particle               --physics.list
--dumpSteeringFile           --gun.position               --physicsList
--enableDetailedShowerMode   -h                           --physics.rangecut
--enableGun                  --help                       --printLevel
--field.delta_chord          -I                           --random.file
--field.delta_intersection   --inputFiles                 --random.luxury
--field.delta_one_step       -M                           --random.replace_gRandom
--field.eps_max              --macroFile                  --random.seed
--field.eps_min              -N                           --random.type
--field.equation             --numberOfEvents             --runType
--field.largest_step         -O                           -S
--field.min_chord_step       --outputFile                 --skipNEvents
--field.stepper              --output.inputStage          --steeringFile
--filter.calo                --output.kernel              -v
--filter.filters             --output.part                --vertexOffset
--filter.mapDetFilter        --output.random              --vertexSigma
```

Example: DetectorModels, Geant4 Shell

First one needs a geometry model described by DD4hep: *compact file*

- `ddsim --compactFile $lcgeo_DIR/ILD/compact/ILD_o1_v05/ILD_o1_v05.xml`

What to do with it

- `ddsim --compactFile $lcgeo_DIR/ILD/compact/ILD_o1_v05/ILD_o1_v05.xml --runType vis # For interactive Geant4 shell with optional visualisation`

Example: Simulation with the Particle Gun

- ddsim
  --compactFile $lcgeo_DIR/ILD/compact/ILD_o1_v05/ILD_o1_v05.xml
  --runType batch
  --enableGun
  --gun.particle kaon0 # whatever particle name is known to geant
  --gun.energy 10*GeV # whatever unit
  --gun.direction "1 1 1" # quotes mandatory here
  --numberOfEvents 10 # or -N 10

Example: Sensitive Detectors

Changing the sensitive detector:

- Most of this can also be done via the command line, it is however recommended to do the mapping via a steering file. The parsing of the tuple of Name and parameter dictionary is only reasonable to do via steering file
- The list of available sensitive detectors should appear, e.g., here in dd4hep doxygen: `http://test-dd4hep.web.cern.ch/test-dd4hep/doxygen/html/group___plugins.html`
- Change the default Sensitive detector for calo
  `SIM.action.calo = "Geant4CalorimeterAction"`
- Use the TPCSDAction sensitive detector for the TPC:
  `SIM.action.mapActions['tpc'] = "TPCSDAction"`
- If there are parameters needed for a sensitive detector:
  `SIM.action.mapActions['ecal'] = ( "CaloPreShowerSDAction", "FirstLayerNumber": 1`

Example: Filters

Sensitive detectors describe what is done with energy deposits, *filters* are used to decide which energy deposits to store

- This was combined in Mokka

New filters can be defined (in the steering file), additional filter plugins can be written if needed.

Default available filters are:

- Geantino Rejector: Do not store hits by Geantinos (not so useful)
- 1 keV minimum energy (edep1kev)
- larger than 0 energy deposit (edep0)
- New filter:
  ```
  SIM.filter.filters['edep3kev'] =
  dict(name="EnergyDepositMinimumCut/3keV", parameter={"Cut":
  3.0*keV} )
  ```
- Assign filter for detectors:
  ```
  SIM.filter.mapDetFilter['FTD'] = "edep1kev"
  ```
- multiple filters
  ```
  SIM.filter.mapDetFilter['FTD'] = ["edep1kev", "geantino"]
  ```
- Disable all filters for certain detectors
  ```
  SIM.filter.mapDetFilter['TPC'] = None # or "" or []
  ```

DDSim on the GRID

- New interface, and work-flow module necessary for ddsim
- Is in latest iLCDirac release
  - Some issues to be resolved in v25r0p1
- Very similar to the Mokka and SLIC interfaces

```
from ILCDIRAC....Applications import Mokka

mo = Mokka()
mo.setInputFile("muons.HEPEvt")
mo.setDetectorModel("CLIC_ILD_CDR")
mo.setSteeringFile("mysteer.steer")
mo.setMacFile('MyMacFile.mac')
mo.setStartFrom(10)
mo.setNumberOfEvents(30)
```

```
from ILCDIRAC....ns import DDSim as Mokka

mo = Mokka()
mo.setInputFile("muons.HEPEvt")
mo.setDetectorModel("CLIC_o3_v04")
mo.setSteeringFile("mysteer.steer")
#mo.setMacFile('MyMacFile.mac')
mo.setStartFrom(10)
mo.setNumberOfEvents(30)
```

DDSim on the GRID

- New interface, and work-flow module necessary for ddsim
- Is in latest iLCDirac release
  - Some issues to be resolved in v25r0p1
- Very similar to the Mokka and SLIC interfaces

```
from ILCDIRAC....Applications import Mokka

mo = Mokka()
mo.setInputFile("muons.HEPEvt")
mo.setDetectorModel("CLIC_ILD_CDR")
mo.setSteeringFile("mysteer.steer")
mo.setMacFile('MyMacFile.mac')
mo.setStartFrom(10)
mo.setNumberOfEvents(30)
```

```
from ILCDIRAC....Applications import DDSim

dd = DDSim()
dd.setInputFile("muons.HEPEvt")
dd.setDetectorModel("CLIC_o3_v04")
dd.setSteeringFile("mysteer.steer")
dd.setStartFrom(10)
dd.setNumberOfEvents(30)
```

Conclusion

- dd4hep based simulation is ready to be used
  - More users means more bugs found, missing features discovered