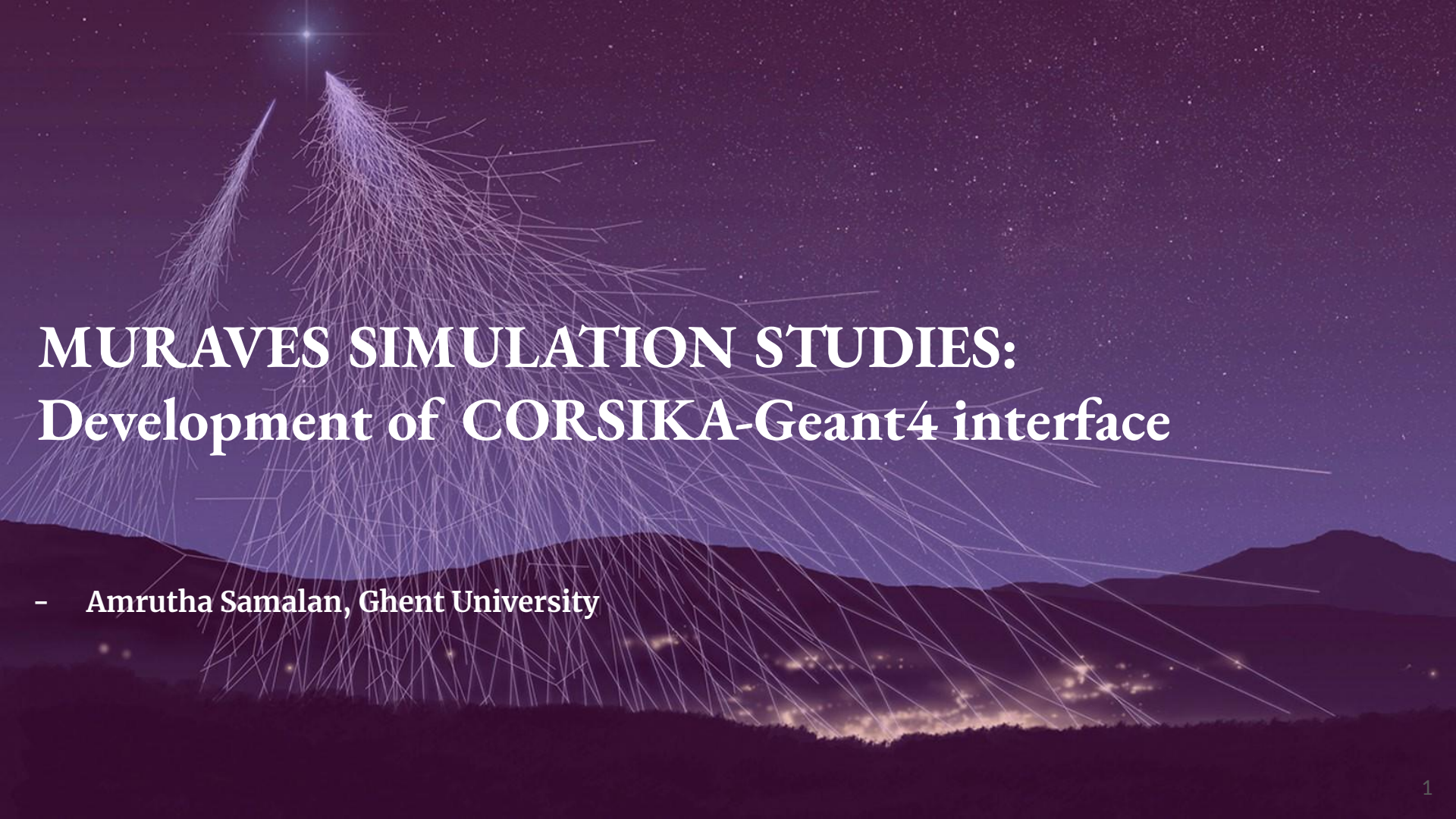

MURAVES SIMULATION STUDIES: Development of CORSIKA-Geant4 interface

- Amrutha Samalan, Ghent University

Overview

- Generating primary events in Geant4
- G4HEPEvtInterface class
- HEPEVT common block
- Convert CORSIKA binary output file to ASCII
- Geant+CORSIKA simulation sample output
- References

Generating primary events in Geant4

- In Geant4, G4VUserPrimaryGeneratorAction concrete class arranges the way primary particles are generated.
- In this class, we have to specify how a primary event should be generated
- In the constructor of your G4VUserPrimaryGeneratorAction, you should specify the primary generator(s)
- An event is generated using a method named generatePrimaries()
- This method is invoked at the beginning of each event
- Geant 4 provides three G4VPrimaryGenerator concrete classes:

1. G4ParticleGun
2. G4GeneralParticleSource (GPS)
3. G4HEPEvtInterface

Generating primary events in Geant4

G4ParticleGun

➔ **This class generates primary particle(s) with a given momentum and position**

➔ **To generate a primary with randomized energy, momentum, and/or position, G4ParticleGun provides various set methods**

- void SetParticleDefinition(G4ParticleDefinition*)
- void SetParticleMomentum(G4ParticleMomentum)
- void SetParticleMomentumDirection(G4ThreeVector)
- void SetParticleEnergy(G4double)
- void SetParticleTime(G4double)
- void SetParticlePosition(G4ThreeVector)
- void SetParticlePolarization(G4ThreeVector)
- void SetNumberOfParticles(G4int)

Generating primary events in Geant4

G4GeneralParticleSource

- ➔ **G4GeneralParticleSource is used exactly the same way as G4ParticleGun**
- ➔ **In existing applications we can simply change our PrimaryGeneratorAction by globally replacing G4ParticleGun with G4GeneralParticleSource**
- ➔ **It allows the specifications of the spectral, spatial and angular distribution of the primary source particles**
- ➔ **GPS allows the user to control the following characteristics of primary particles:**
  - Spatial sampling: on simple 2D or 3D surfaces such as discs, spheres, and boxes.
  - Angular distribution: unidirectional, isotropic, cosine-law, beam or arbitrary (user defined).
  - Spectrum: linear, exponential, power-law, Gaussian, blackbody, or piece-wise fits to data.
  - Multiple sources: multiple independent sources can be used in the same run.

Generating primary events in Geant4

G4HEPEvtInterface

- This class provides an interface to the event generators
- Almost all event generators presently in use, commonly are written in FORTRAN ; CORSIKA also
- But, in Geant4 it is not possible to link with any FORTRAN program or library because of losing many advantages of object-oriented features of C++
- Instead, Geant 4 provides an ASCII file interface for such event generators
- G4HEPEvtInterface reads an ASCII file produced by an event generator and reproduces G4PrimaryParticle objects
- It reproduces a full production chain of the event generator, starting with primary quarks, protons, etc.

G4HEPEvtInterface

```
#ifndef ExN04PrimaryGeneratorAction_h
#define ExN04PrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"
#include "globals.hh"

class G4VPrimaryGenerator;
class G4Event;

class ExN04PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
  public:
    ExN04PrimaryGeneratorAction();
    ~ExN04PrimaryGeneratorAction();

  public:
    void GeneratePrimaries(G4Event* anEvent);

  private:
    G4VPrimaryGenerator* HEPEvt;
};

#endif


#include "ExN04PrimaryGeneratorAction.hh"

#include "G4Event.hh"
#include "G4HEPEvtInterface.hh"

ExN04PrimaryGeneratorAction::ExN04PrimaryGeneratorAction()
{
  HEPEvt = new G4HEPEvtInterface("pythia_event.data");
}

ExN04PrimaryGeneratorAction::~ExN04PrimaryGeneratorAction()
{
  delete HEPEvt;
}


void ExN04PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  HEPEvt->SetParticlePosition(G4ThreeVector(0.*cm,0.*cm,0.*cm));
  HEPEvt->GeneratePrimaryVertex(anEvent);
}
```

- An example of how to use the G4HEPEvtInterface
- The constructor of G4HEPEvtInterface takes the file name
- The interaction point must be set since an event generator is not assumed to give a place of the primary particles

HEPEVT common block

- An ASCII file, which will be fed by G4HEPEvtInterface should have the following format

```
 PARAMETER (NMXHEP=4000)
 COMMON/HEPEVT/NEVHEP,NHEP,ISTHEP(NMXHEP),IDHEP(NMXHEP),
&JMOHEP(2,NMXHEP),JDAHEP(2,NMXHEP),PHEP(5,NMXHEP),VHEP(4,NMXHEP)
 DOUBLE PRECISION PHEP, VHEP
```

- This standard defines an event record structure which should make the interfacing of different event generators much simpler
- First line shows the maximum numbers of entries (particles)
- Each line in an event corresponds to a particle in the /HEPEVT/ common
- G4HEPEvtInterface converts information stored in the /HEPEVT/ common block to an object-oriented data structure
- The /HEPEVT/ common block is commonly used by almost all event generators written in FORTRAN, G4HEPEvtInterface can interface to almost all event generators currently used in the HEP community

Convert CORSIKA binary output file to ASCII

- The default output particle data file is of binary format (root file also possible if we install CORSIKA in express mode)
- In the subdirectory /src/utils, FORTRAN routines for different tasks are available

```
amrutha@amrutha-XPS-15-7590:~/CORSIKA/corsika-venus/src/utils$ ls
bcreinpcont.cpp  corsikahisto_mthin.f  cskreadme.sh     rcorsik2beok.cpp  readcskralplot.sh  readparticall.sh  shdatareduction.f     showsimprods.f          sortaugerhit.sh    summ.sh
bcreinpcont.f    corsikahisto_thin.f   gdastool         readcorsika.cpp   README             readparticles.f   shdatareduction.sh    showsimprods.sh         sumlistnkginfo.f   work.inc
coast            corsikaread.cpp       map2png.c        readcsk2asci.f    readmthinmuons.f   readparticles.i   shdatatanks.f         showsimulist.cpp        sumlistnkginfo.sh
cors2input.cpp   corsikaread.f         map2png.o        readcsk2beok.f    readmthinmuons.sh  readtimes.f       shdatatanks.pmsoutab  showsimulist.f          sumlongifiles.f
cors2input.f     corsikaread_history.f modelprint.f     readcsk2prtcls.f  readmthinprtcls.f  readtimesnew.f    shdatatanks.sh        showsimulist.sh         sumlongifiles.sh
cors2input.name  corsikaread_thin.f    plottracks3c.f   readcsk2prtcls.sh readmthinprtcls.sh readtimes.sh     showintact1st.f       showversion-corsika.sh  summ
corsikahisto.f   corsplitevts.f        plottracks3c.o   readcskralplot.f  readparticall.f    seconds2date.c    showintact1st.sh      sortaugerhit.f          summe.f
```

- readcsk2asci.f converts CORSIKA binary file into ASCII

Convert CORSIKA binary output file to ASCII

```
c  r e a d c s k 2 a s c i . f
c  --------------------------
c  write corsika shower data as readable ascii text file; it needs a
c  factor 3.5234 of disk space as of the binary particle data file;
c  the protocol output and the big ascii particle data file will be
c  written to the current directory: 2 records written, if
c  'total_number_of_files' > 0, or up to maximum 234567890 records
c  (i.e. 5.38 TBytes !!), if 'total_number_of_files' < 0;
c  ==> increase `lrecmax` data statement to print more records.
c-----------------------------------------------------------------------
c CompLink:
c          gfortran -O0 -fbounds-check readcsk2asci.f -o readcsk2asci
c          ifort -C -O0 -check bounds readcsk2asci.f -o readcsk2asci
c RunProg:
c          ./readcsk2asci < readcsk2asci.i
c-----------------------------------------------------------------------
c          RUNH = 211285.2812500000000000;
c          EVTH = 217433.0781250000000000;
c          LONG =  52815.2968750000000000;
c          EVTE =   3397.3918457031250000;
c          RUNE =   3301.3325195312500000;
c-----------------------------------------------------------------------
c input-files:
c          unit=*: number of showers and file name(s):
c          -------------------------------------------------
c                     1          'total_number_of_showers'
c                     1          'total_number_of_files'
c           '/lxdata/d2lx14/joe/DAT045216'
c                     1          '_showers_of_this_file_'
c          -------------------------------------------------
c          unit=3: current corsika particle data file.
c output-files:
c          unit=*: protocol output.
c          unit=9: ascii file named DATnnnnnn.ascithin
c                  or ....ascistnd or ....ascimthi.
c-----------------------------------------------------------------------
c                                  juergen.oehlschlaeger@kit.edu
c-----------------------------------------------------------------------
```

- Install a FORTRAN compiler
- Run any of the CompLink

```
amrutha@amrutha-XPS-15-7590:~/CORSIKA/dpmjet/src/utils$ ls
bcreinpcont.cpp        map2png.c              readmthinmuons.sh      showsimprods.f
bcreinpcont.f          modelprint.f           readmthinprtcls.f      showsimprods.sh
coast                  plottracks3c.f         readmthinprtcls.sh     showsimulist.cpp
cors2input.cpp         rcorsik2beok.cpp       readparticall.f        showsimulist.f
cors2input.f           readcorsika.cpp        readparticall.sh       showsimulist.sh
cors2input.name        readcsk2asci           readparticles.f        showversion-corsika.sh
corsikahisto.f         readcsk2asci.f         readparticles.i        sortaugerhit.f
corsikahisto_mthin.f   readcsk2asci.i         readtimes.f            sortaugerhit.sh
corsikahisto_thin.f    readcsk2asci.i.save    readtimesnew.f         sumlistnkginfo.f
corsikaread.cpp        readcsk2beok           readtimes.sh           sumlistnkginfo.sh
corsikaread.f          readcsk2beok.f         seconds2date.c         sumlongifiles.f
corsikaread_history.f  readcsk2beok.names     shdatareduction.f      sumlongifiles.sh
corsikaread_thin.f     readcsk2prtcls.f       shdatareduction.sh     summ
corsplitevts.f         readcsk2prtcls.sh      shdatatanks.f          summe.f
cskreadme.sh           readcskralplot.f       shdatatanks.pmsoutab   summ.sh
DAT000001              readcskralplot.sh      shdatatanks.sh         work.inc
DAT000001.ascistnd     README                 showintact1st.f
gdastool               readmthinmuons.f       showintact1st.sh
```

- Create a file named readcsk2asci.i

```
 GNU nano 4.3                        readcsk2asci.i
10
1
"/home/amrutha/CORSIKA/dpmjet/src/utils/DAT000001"
1
```

- Output: DATXXXXXX.ascistnd

Geant+CORSIKA simulation output

- Give this ASCII file as input to G4HEPEvtInterface
- Sample output:

```
G4WT1 > ### Run 0 starts on worker thread 1.
G4WT0 > ### Run 0 starts on worker thread 0.
G4WT1 > --> Event 0 starts with initial seeds (29353504,75991678).
G4WT0 > >>> Event: 1
G4WT1 > >>> Event: 0
G4WT1 >     0 hits stored in this event
G4WT1 > >>> Event: 2
G4WT0 >     0 hits stored in this event
G4WT0 > >>> Event: 3
G4WT1 >     0 hits stored in this event
G4WT1 > >>> Event: 4
G4WT0 >     0 hits stored in this event
G4WT0 > Thread-local run terminated.
G4WT0 > Run Summary
G4WT0 >   Number of events processed : 2
G4WT0 >   User=0.010000s Real=0.001871s Sys=0.000000s [Cpu=534.3%]
G4WT1 >     0 hits stored in this event
G4WT1 > Thread-local run terminated.
G4WT1 > Run Summary
G4WT1 >   Number of events processed : 3
G4WT1 >   User=0.010000s Real=0.003404s Sys=0.000000s [Cpu=293.8%]
```

To Do

- Setting proper primary particle position
- Try recording hit information

References

- https://geant4-userdoc.web.cern.ch/UsersGuides/IntroductionToGeant4/fo/IntroductionToGeant4.pdf
- https://web.ikp.kit.edu/corsika/usersguide/usersguide.pdf
- https://home.fnal.gov/~mrenna/lutp0613man2/node49.html

THANK YOU