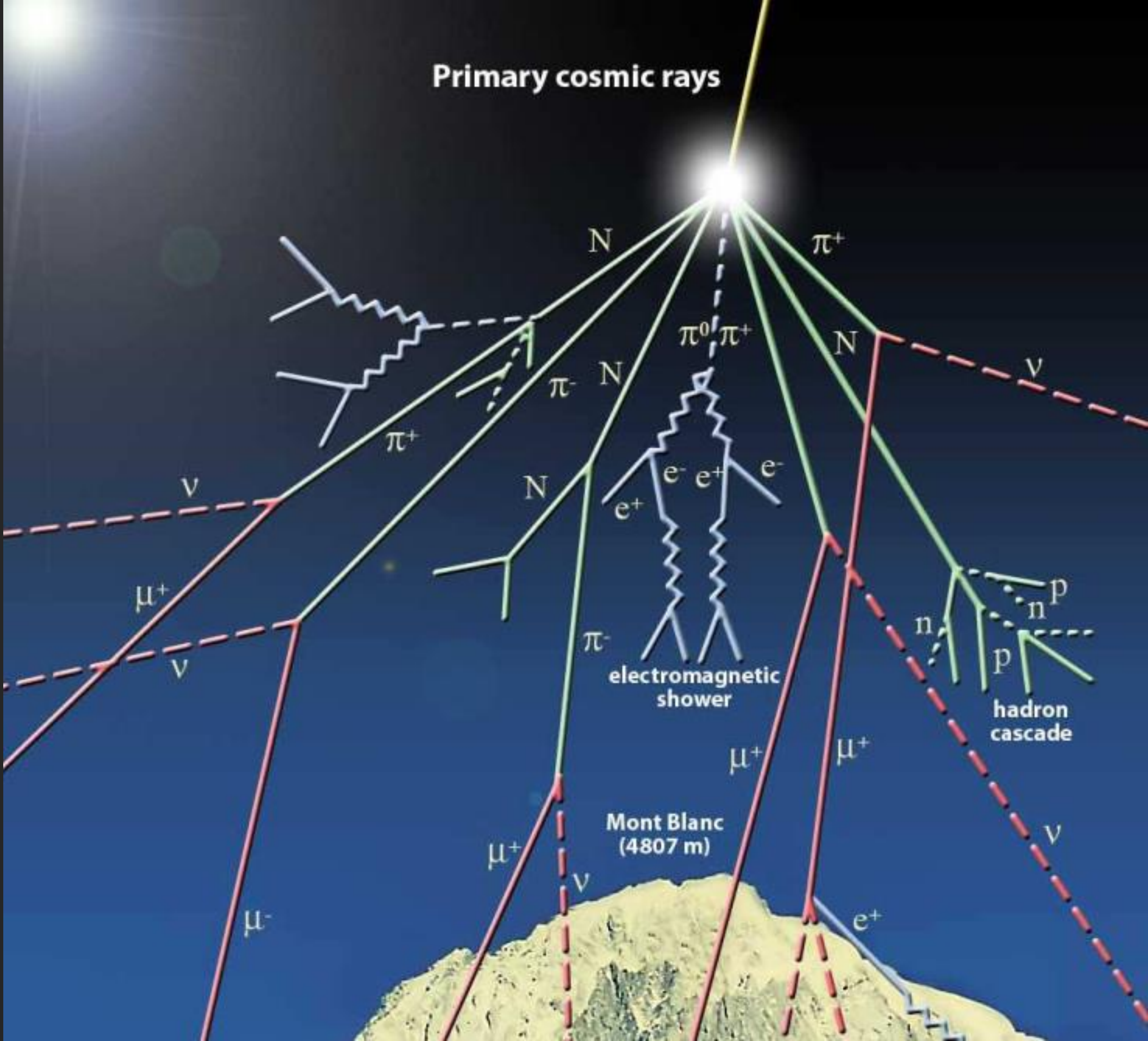
ATMOSPHERIC AND DETECTOR SIMULATIONS

A. SERIPIENLERT AND A. PAGWAN

*Based on P.-S. Mangeard codes and scripts



ATMOSPHERIC SIMULATION

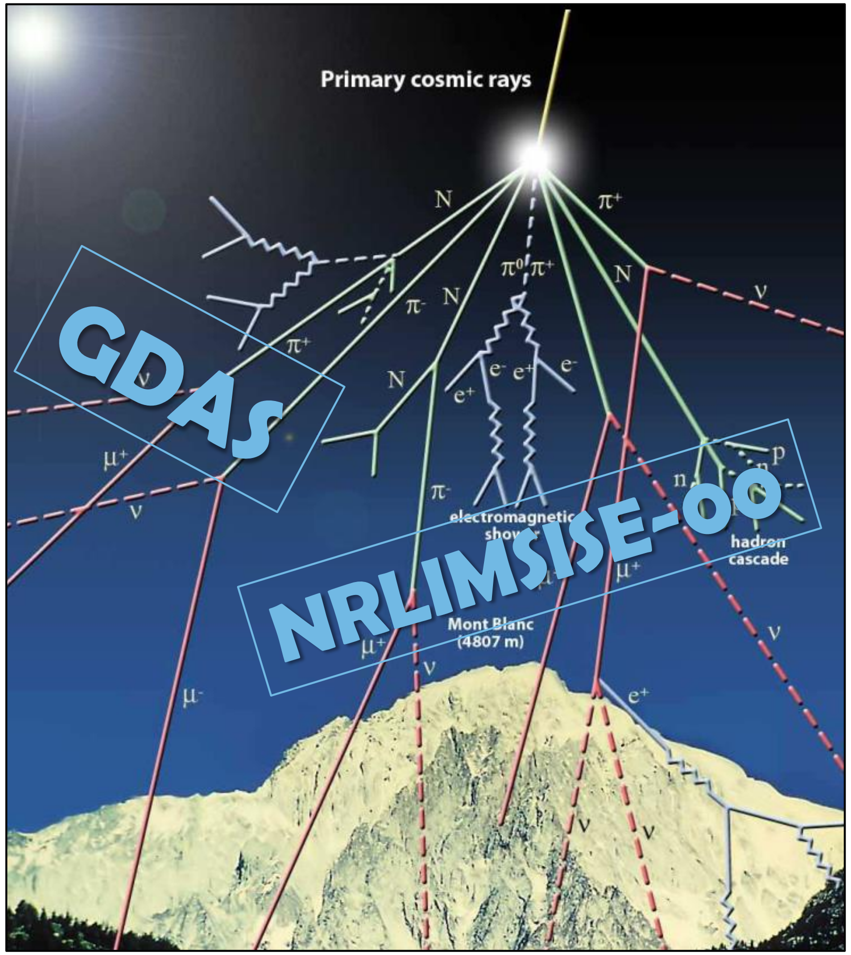
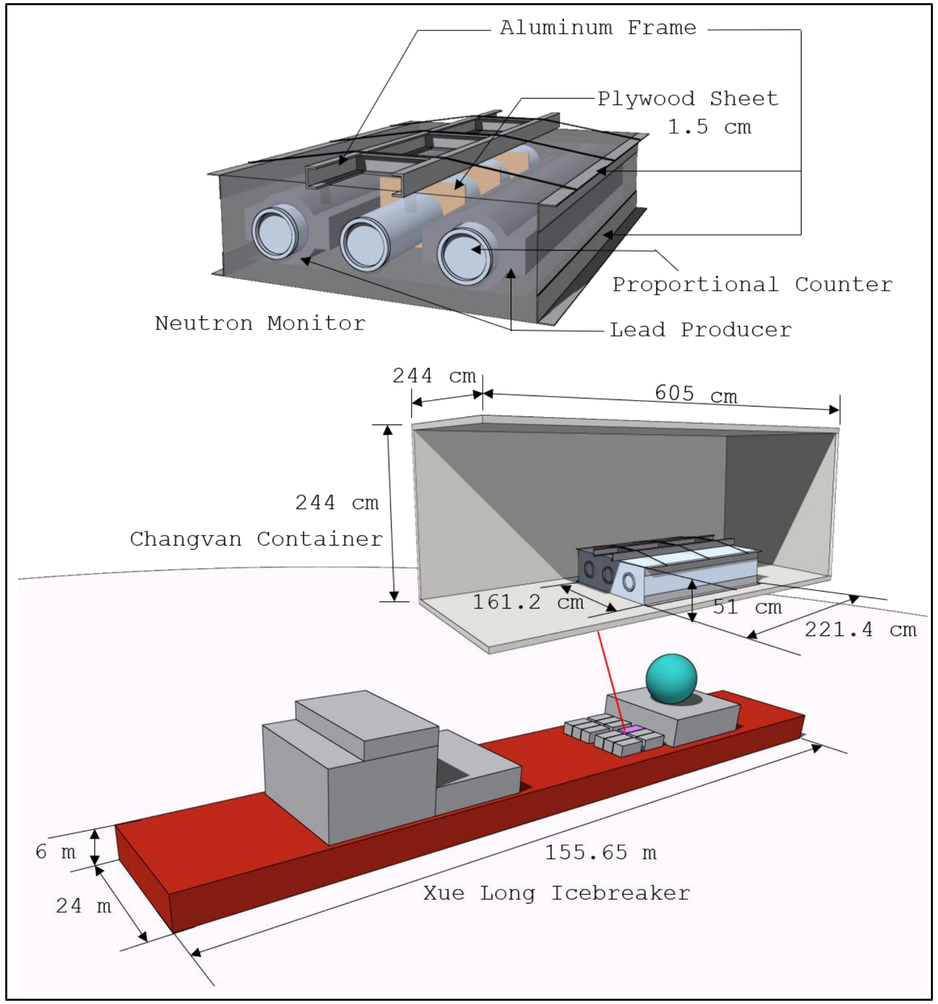


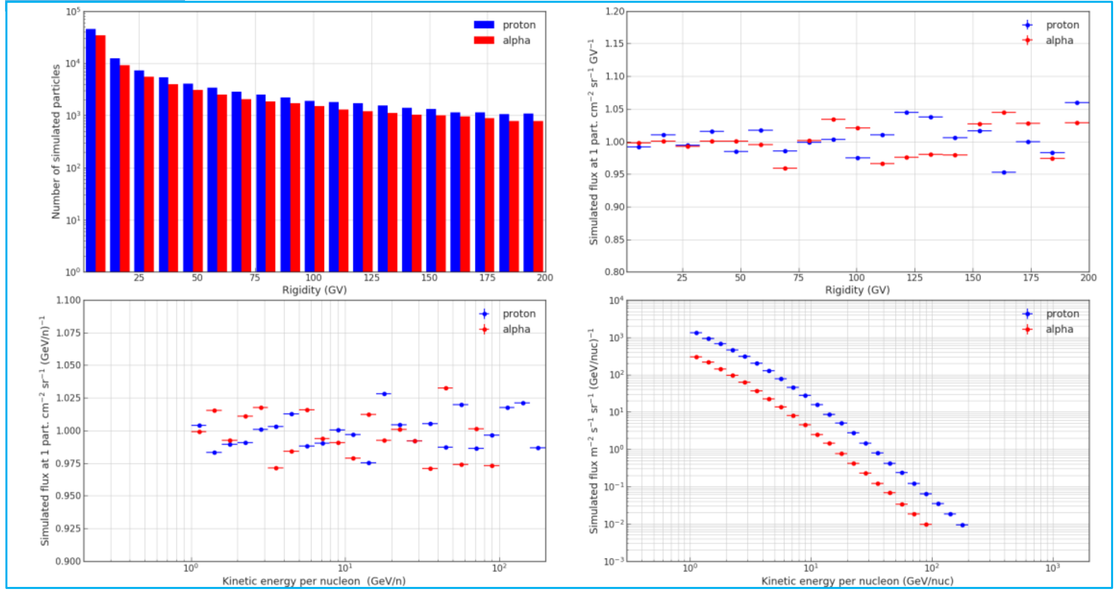
Image credit: http://scifun.ed.ac.uk/card/images/left/cosmic-rays.jpg

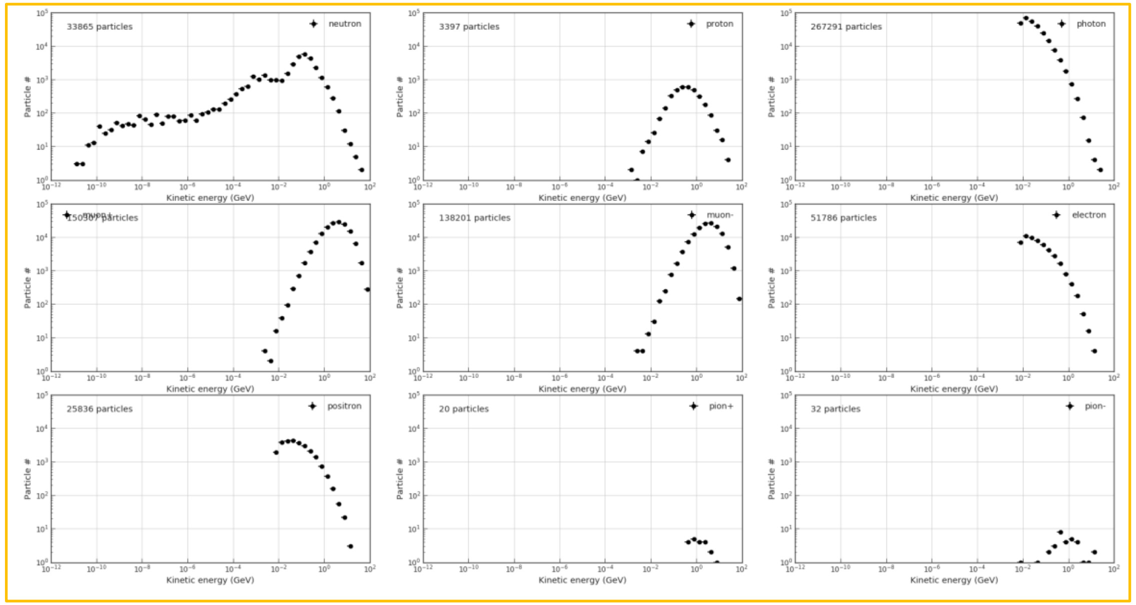
DETECTOR SIMULATION



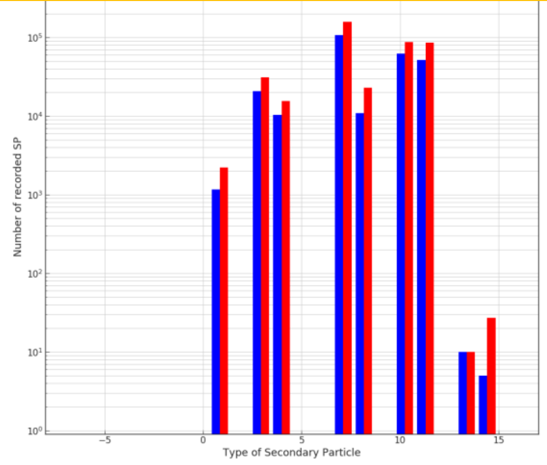
ATMOSPHERIC SIMULATION RESULTS

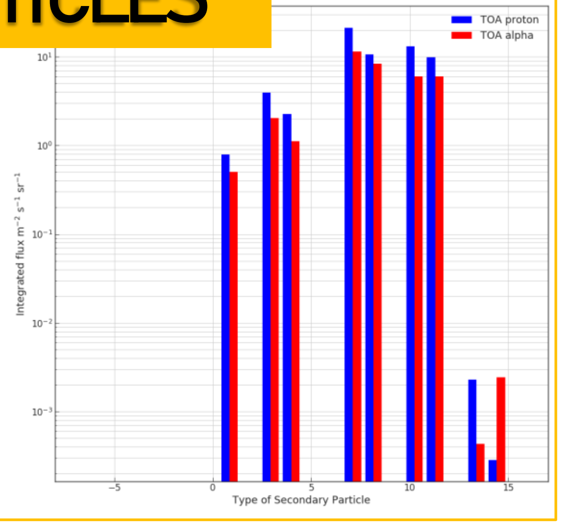
TOA

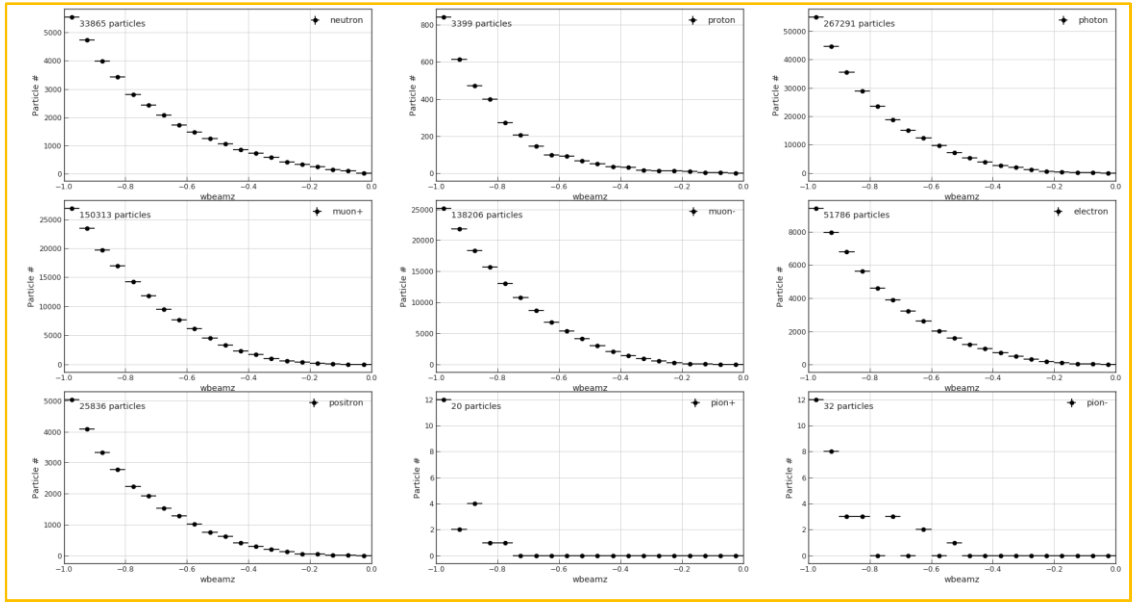




SECONDARY PARTICLES

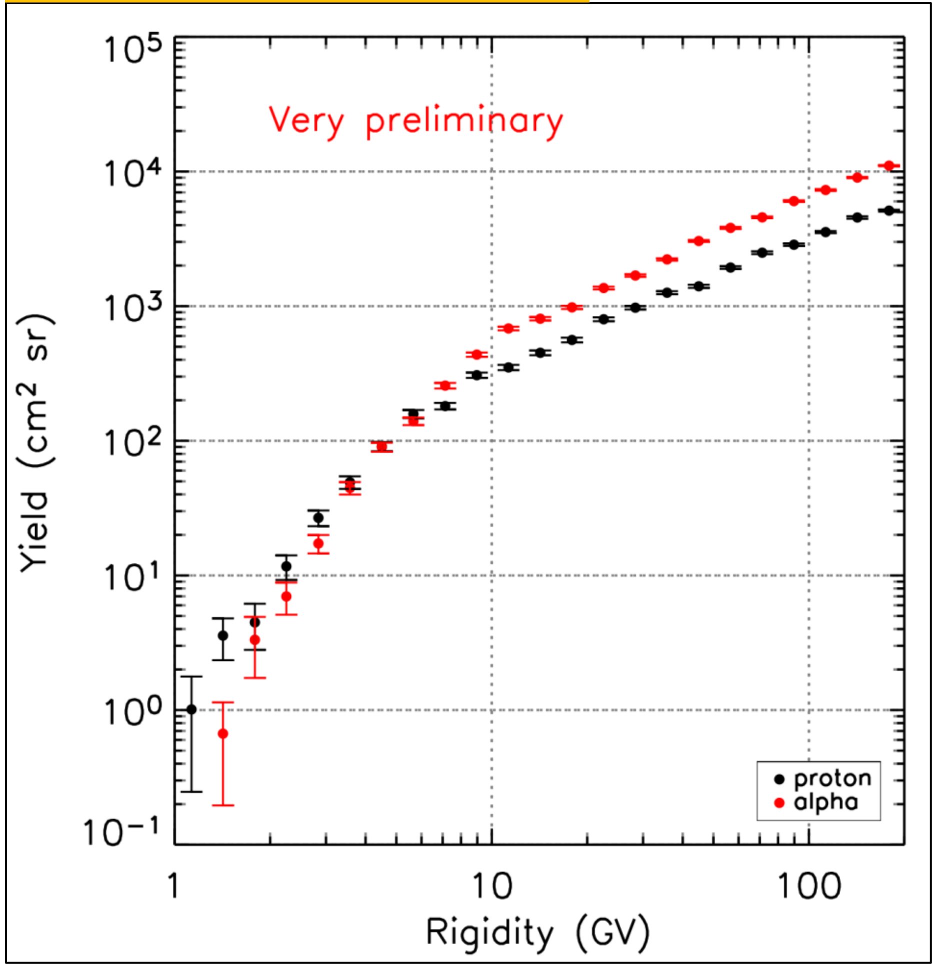




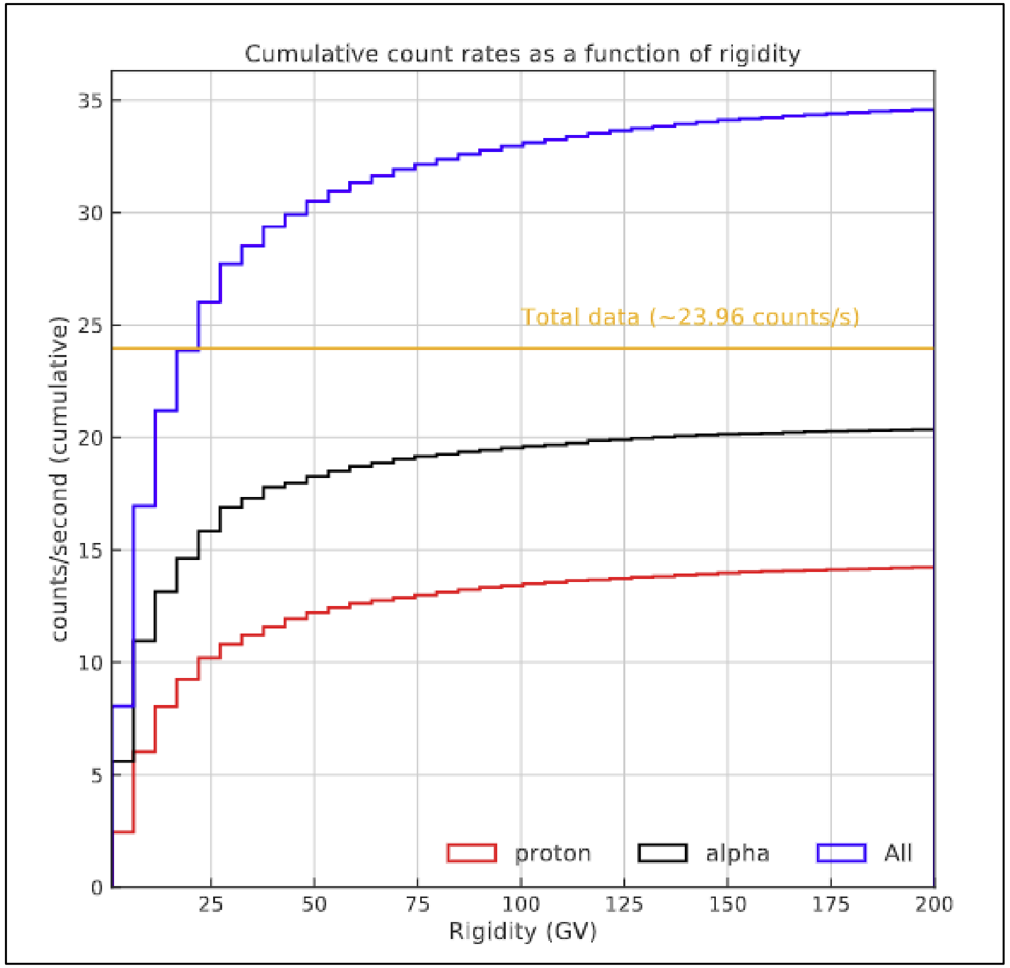


DETECTOR SIMULATION RESULTS

YIELD FUNCTION



COUNT RATES



ATMOSPHERIC SIMULATION

INPUT

- Detector's location (LAT, LON, ALT)
- Primary particles
- Rigidity range
- Spectral index

FILES NEED FOR RUN

- .inp
- .f
- .cards
- .geo
- .sh (for compile)

STEPS FOR RUN

```
source CompileAndLink.sh
```

```
nohup rfluka -N0 -M100 -e DIAtmo  DIproton.inp > info-prot &
nohup rfluka -N0 -M100 -e DIAtmo  DIalpha.inp > info-alpha &
```

Library of secondary particles (sp) - NEED TO RECREATE EVERY TIME YOU RUN MORE CYCLE

```
cat DI*spneut > Library_spneut
cat DI*spprot > Library_spprot
cat DI*spmuon > Library_spmuon
```

Find number of particles in Library of secondary particles (sp)

```
wc -l Library*
```

DETECTOR SIMULATION

INPUT

- Detector's location (LAT, LON, ALT)
- Primary particles
- Rigidity range
- Spectral index

FILES NEED FOR RUN

- Libraries
- .inp (correspond to library)
- .cards (correspond to your detector)
- source.f -> use beam position of your detector / directory of library
- mgdraw.f
- .sh (for compile)

STEPS FOR RUN

```
source CompileAndLink.sh
```

NEED TO RUN from N0 every time you update Library

```
nohup rfluka -N0 -M10 -e PSNMF4SP PSNMF4_SPneutron.inp > SPneutron &
nohup rfluka -N0 -M50 -e PSNMF4SP PSNMF4_SPproton.inp > SPproton &
nohup rfluka -N0 -M100 -e PSNMF4SP PSNMF4_SPmuon.inp > SPmuon &
```

ANALYSIS PACKAGE (PYTHON3)

- 7 python files
- Physics_var.py: Definition of physic variables
- GCRFlux.py: Definition of 2 different models of GCR flux at the top of the atmosphere
- Classes.py: Definition of 4 classes for events, counts, primary particles and secondary particles
- Check_FluxTOAatDI.py and Check_FluxSPatDI.py: Check the simulation outputs
- MCyield_config.py: Configuration files that contains the information related to the MC simulations (number of tubes, pathnames, number of particles, number of cycles, GCR spectrum, etc) We will go over it together
- AnalyseCounts.py: Main analysis code, read files, create events, apply deadtimes, calculate yield functions, make figures etc

MCyield_config.py

```
########################
### Detector configuration
########################

#FOR PSNM (You may have to add/remove variables depending on  your detector)
N=18        # Number of tubes
DeltaT=50   # distance between tubes
shiftT=10   # tube shift (used to determine easily the tube where the count is)

#Dead time in tubes
#Same index as tube number from Thanin
#dt=[20.6e-6,21.2e-6,19.6e-6,20.0e-6,19.6e-6,20.0e-6,28.8e-6,20.4e-6,27.7e-6,28.8e-6,26.8e-6,20.4e-6,19.6e-6,20.4e
-6,19.6e-6,20.0e-6,26.4e-6,28.0e-6]
#Same index as tube number from Thanin after switch 1<->18
dt=[28.0e-6,21.2e-6,19.6e-6,20.0e-6,19.6e-6,20.0e-6,28.8e-6,20.4e-6,27.7e-6,28.8e-6,26.8e-6,20.4e-6,19.6e-6,20.4e-
6,19.6e-6,20.0e-6,26.4e-6,20.6e-6]
dt=numpy.array(dt)
#NEED TO FLIP THE ARRAY TO GET THE SAME INDEX ORDERING THAN IN FLUKA
dt=numpy.flip(dt)
```

MCyield_config.py

```
########################
### Atmospheric simulation information
########################
#First index is for proton simulation
#Second index is for alpha simulation
#Number of simulated particle per cycle
Npc=numpy.array([5000,5000])
#Number of cycle
Nc=numpy.array([10,10])
#Number of simulated particles (is automatically a numpy array as a product of 2 numpy arrays)
Npp=Nc*Npc
#Range of simulated rigidity
MinRig=numpy.array([10.,10])
MaxRig=numpy.array([200.,200.])
```

	proton	alpha
N	5000	5000
Nc	10	10

MCyield_config.py

```
########################
### Flux of primary particles at the top of the atmosphere (for exammple to get count rates and Dorman functions)
########################
#Force field Solar modulation in MV. The value should correspond to the right flux model of GCR
PHI=300
#SET SOLAR MODULATION PHI in GV
PHI=PHI/1000.

#Two choices of flux at TOA
#Usoskin LIS model (US17): 'US17'
TOAmod = 'US17'
#Use values of Ghelfi et al
#TOAmod= 'GH17'
#GhLIS_index is between 0 and 5: use 0 or 5 (See the 6 column definition in their paper)
GhLIS_index=0
```

MCyield_config.py

```
########################
### Detector simulation information
########################

#Beam limit for vertical secondary particles in cm
#Values from source.f of your detector simulation
baxmin=-1310
baxmax=2090
baymin=-1350
baymax=1650
BeamArea=(baxmax-baxmin)*(baymax-baymin)

#Number of particles in SP library
#neutrons, protons  (Can add more )
Npartlib=numpy.array([59811,7480])

#Number of cycles of SP
Nspcyc=numpy.array([5,5])
#Number of particle per cycle
Npartsp=numpy.array([400000,400000])
```

59811 Library_spneut
7480 Library_spprot
1465924 Library_spmuon

	neutron	proton	muon
N	400,000	400,000	500,000
Nc	5	5	100

MCyield_config.py

```
########################
### Parameters for the script
########################

#Input path directory
Inppath='/home/psm/Documents/Fluka4/Detector/PSNM/'
#Output path directory
Outpath='.'

#List of string for glob (look for counts files to read)
#The glob module finds all the pathnames matching a specified pattern
#according to the rules used by the Unix shell, although results are returned
#in arbitrary order. No tilde expansion is done, but *, ?, and character
#ranges expressed with [] will be correctly matched.
listglob=['PSNMF4_SPneutron00*_counts','PSNMF4_SPproton00*_counts']

#Flag Createnpz:
# if True then all the counts files are read and the npz file  defined by the
# variable npzfile is created with all the events
# if False: the file defined by the variable npzfile is loaded with all the events.
# The option False should be used once all the events were previously created with the flag at True.
# It will make the code runs much faster especially with large statistics
Createnpz=True
```

AnalyseCounts.py

```python
########################################
########################################
########################################
# Weigthing of SECONDARY PARTICLE FLUX
########################################
########################################
########################################

#weighted counts for yield per tube
YieldTube=numpy.ones((len(Ncts),N))
YieldTube[tSP==8]= NCtsTube[tSP==8]*wPP[tSP==8,None]*wSP[tSP==8,None]/Ncallave[0]  #For neutrons
YieldTube[tSP==1]= NCtsTube[tSP==1]*wPP[tSP==1,None]*wSP[tSP==1,None]/Ncallave[1]  #For protons
YieldTube[tSP==10]= NCtsTube[tSP==10]*wPP[tSP==10,None]*wSP[tSP==10,None]/Ncallave[2]  #For muon+
YieldTube[tSP==11]= NCtsTube[tSP==11]*wPP[tSP==11,None]*wSP[tSP==11,None]/Ncallave[2]  #For muon-
#Can add a line here for muons with different mask

#weighted counts for total yield
Yield=numpy.ones(len(Ncts))
Yield[tSP==8]= Ncts[tSP==8]*wPP[tSP==8]*wSP[tSP==8]/Ncallave[0]     #For neutrons
Yield[tSP==1]= Ncts[tSP==1]*wPP[tSP==1]*wSP[tSP==1]/Ncallave[1]     #For protons
Yield[tSP==10]= Ncts[tSP==10]*wPP[tSP==10]*wSP[tSP==10]/Ncallave[2]     #For muon+
Yield[tSP==11]= Ncts[tSP==11]*wPP[tSP==11]*wSP[tSP==11]/Ncallave[2]     #For muon-
#Can add a line here for muons with different mask

#weighted counts for the chosen flux at TOA (total detector)
Weightedcounts=numpy.ones(len(Ncts))
Weightedcounts[tSP==8]= Ncts[tSP==8]*wGCR[tSP==8]*wSP[tSP==8]/Ncallave[0]   #For neutrons
Weightedcounts[tSP==1]= Ncts[tSP==1]*wGCR[tSP==1]*wSP[tSP==1]/Ncallave[1]   #For protons
Weightedcounts[tSP==10]= Ncts[tSP==10]*wGCR[tSP==10]*wSP[tSP==10]/Ncallave[2]   #For muon+
Weightedcounts[tSP==11]= Ncts[tSP==11]*wGCR[tSP==11]*wSP[tSP==11]/Ncallave[2]   #For muon-
#Can add a line here for muons with different mask
```