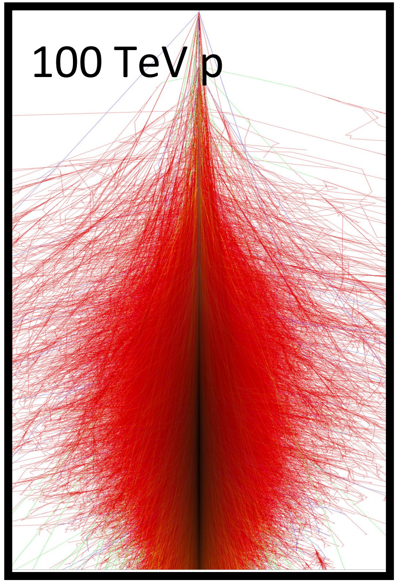
Interfacing CORSIKA air shower simulations with LArSoft

Matt Bass (Oxford)
LArSoft Coordination Meeting
24 November 2015

Overview

- Brief CORSIKA overview
  - CORSIKA vs CRY
  - How CORSIKA is run
- CorsikaGen Module
- CorsikaGen Jobs
- Related issue: LArG4 keep particles by volume feature

CORSIKA



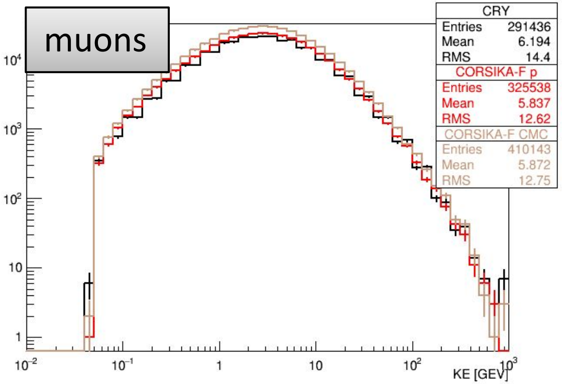
- **CORSIKA** simulates extensive air showers initiated by cosmic ray particles
  - Wide range of energy scales, multiple primary types (p, He, Fe, etc.)
  - Resulting secondary particle flux evaluated at a specified altitude
  - Multiple models available for interactions (GHEISHA, FLUKA, etc.)
- Past CR simulations for uboone have been with CRY
  - Gives pre-generated distributions of particles at sea level, 2100 m, 11300 m based on tables from full MCNPX simulations of proton primaries between 1 GeV and 100 TeV
- CRY has a few shortcomings:
  - FNAL (226 m) elevation not available, roughly 10% effect in muons
    - Larger effect in neutrons, protons, and electrons
  - Wide energy bins lead to binning artifacts in E,θ
  - Only simulates proton primaries
  - Limited to only using MCNPX model for particle propagation

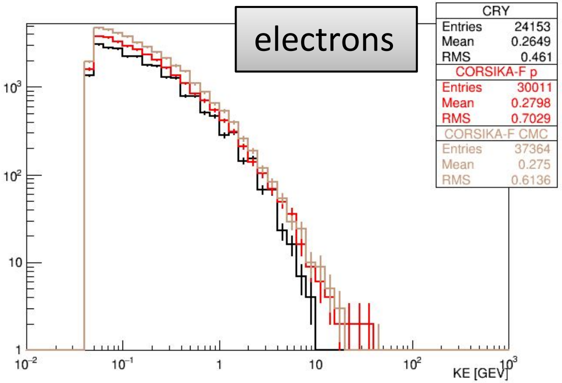
Running CORSIKA

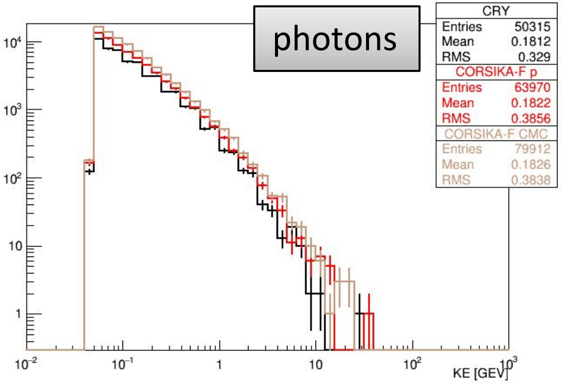
- CORSIKA is available to download from ikp.kit.edu/corsika/
  - Compiles using f77
  - Some options (like FLUKA) have external dependencies
- Binary output file converted to root using corsikaConverter tool
  - Developed by Stefano Tognini, NOvA
- Developed set of scripts to run on the grid
  - Created a **tar** of CORSIKA executables + data + FLUKA
  - **Wrapper** script creates input cards, writes a dag file, and submits a job with the tar
  - **Node** script unpacks tar, executes CORSIKA, executes corsikaConverter, copies outputs back
  - 10 events for uboone ~ 64 ms sim. time ~ 1 million showers ~ 20 GB binary output ~ 4 hours
  - Outputs stored in pnfs scratch space

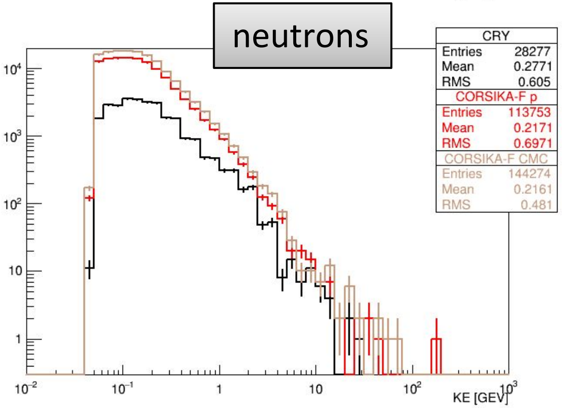
CRY vs CORSIKA-FLUKA-p vs CORSIKA-FLUKA-CMC

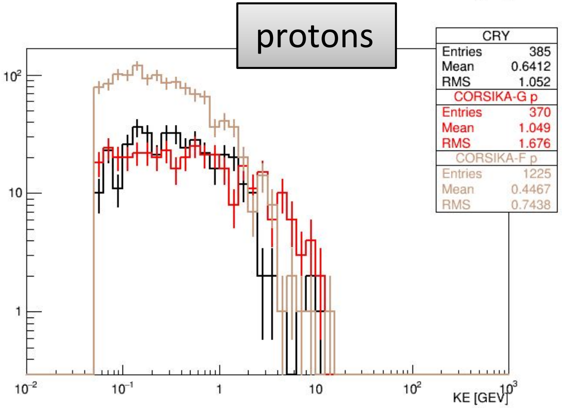
Particles through top of TPC, 96 seconds, Pre-g4

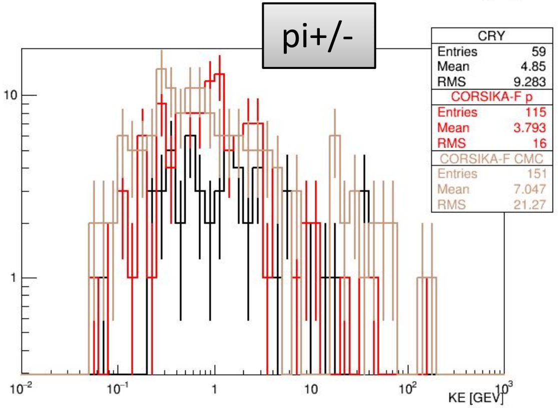












CORSIKA CMC vs CRY rates

µBooNE LArTPC	CORSIKA-FLUKA CMC Rate [$Hz.m^{-2}$]	CRY (proton) Rate [$Hz.m^{-2}$]	Ratio CMC/CRY
muons	161 ± 2.5	114.3 ± 2.1	1.4
photons	31.3 ± 1.1	19.7 ± 0.9	1.6
e	14.7 ± 0.8	9.5 ± 0.6	1.6
p	6.0 ± 0.5	1.5 ± 0.2	4.0
n	56.6 ± 1.5	11.1 ± 0.7	5.1
pion	0.06	0.02	2.6
kaons	0.001	0	-

- Total time≈96 s
- **Pre-g4**
- $KE_{pre\text{-}g4}$>50 MeV

Points of reference	
Measured muon rate at LArTF (uboone docdb 3130)	141 ± 21 $Hz.m^{-2}$
Predicted muon rate at sea level (uboone docdb 430); E>0.2 GeV	172.2 $Hz.m^{-2}$
Predicted neutron rate at **sea level** Ziegler: 5.65e-3 $Hz.cm^2$, KE>10 MeV	56.5 $Hz.m^{-2}$

CorsikaGen Module

- Initial studies used **hepevt** formatted inputs to get events into larsoft
  - Messy and limited unique events to those already generated

- Developed **CorsikaGen module** to
  - Resample from pre-generated CORSIKA samples so there is no need for users to have CORSIKA/FLUKA installations
  - Use sqlite databases to store shower/particle information from CORSIKA
  - At run time, assemble an event by randomly sampling from showers table and arranging the particles in space/time via: $I_N(E) \approx 1.8 \times 10^4 \, (E/1\,\mathrm{GeV})^{-\alpha} \, \frac{\mathrm{nucleons}}{\mathrm{m^2\ s\ sr\ GeV}}$

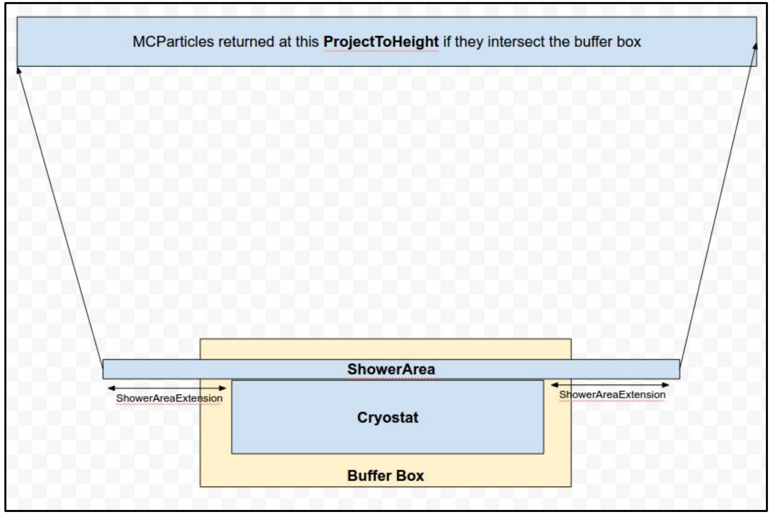
- **Module** lives in larsim/EventGenerator/CORSIKA
  - **Data** files available on per experiment basis (more on this later)
  - Standard fcl should work as a start for all experiments

CorsikaGen Event Generation

1. Random sample of showers, and associated particles, queried from database and distributed at the **ShowerArea** position/time:
   a. Position is shower particle wrapped arrival position at surface; preserves spatial correlations of particles within shower
   b. Time is shower particle's arrival time plus some random offset per shower of a **SpillTime** interval: preserves temporal correlations between particles within shower

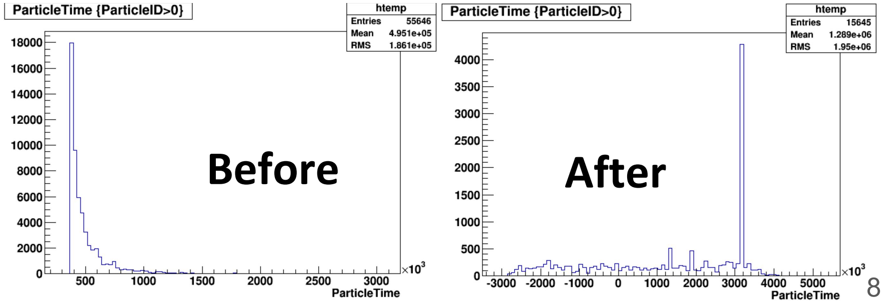
2. Particles kept if they will intersect the **BufferBox**

3. Particles backwards propagated to **ProjectToHeight**



$$I_N(E) \approx 1.8 \times 10^4 \, (E/1 \text{ GeV})^{-\alpha} \, \frac{\text{nucleons}}{\text{m}^2 \text{ s sr GeV}}$$

Timing Distributions



CorsikaGen Parameters

```
standard_CORSIKAGen_protons:
{
 module_type:          "CORSIKAGen"
 SampleTime:            6.4e-3                #integration time in seconds
 TimeOffset:            0.0                   #time in seconds before a spill to begin the interactions
 ProjectToHeight:       1800                  #height to which particles are projected [cm]
 ShowerInputFiles:      [ "p_showers.db" ]    #list of sqlite dbs with corsika showers
 ShowerFluxConstants:   [ 1.8e4 ]             #list of flux constants per shower file
 BufferBox:             [ -250.0,250.0,-250.0,250.0,-250.0,250.0 ] #list of buffer box extensions to cryo volume in each dimension/dir (-x,+x,-y,+y,-z,+z)
 ShowerAreaExtension:   1000                  #amount to extend the shower area beyond the cryo dimensions
 RandomXZShift:         1000                  #amount to randomly shift shower start point in x & z [cm]
}
```

- Parameters similar to CRY interface (CosmicsGen_module): **SampleTime**, **TimeOffset**, **BufferBox**
- **ProjectToHeight**, **ShowerAreaExtension** described on previous slide
- **RandomXZShift**: randomize start position of shower within area (uniform distribution)
  - Keeps shower particles from ending up in the same place every time a shower is used
- **ShowerInputFiles**: list of flux databases to use
- **ShowerFluxConstants**: flux normalization (K) to use in: $I_N \approx K(E/1\mathrm{GeV})^{-\alpha} \frac{\mathrm{nucleons}}{\mathrm{m}^2 \cdot \mathrm{s} \cdot \mathrm{sr} \cdot \mathrm{GeV}}$

CorsikaGen Parameters - Flux Mixing

```
standard_CORSIKAGen_CMC:
{
 module_type:          "CORSIKAGen"
 SampleTime:            6.4e-3               #integration time in seconds
 TimeOffset:            0.0                  #time in seconds before a spill to begin the interactions
 ProjectToHeight:       1800                 #height to which particles are projected [cm]
 ShowerInputFiles:      [ "p_showers.db", "He_showers.db", "N_showers.db", "Mg_showers.db", "Fe_showers.db"] #list of sqlite dbs with corsika showers
 ShowerFluxConstants:   [ 1.72e4, 9.2e3, 6.2e3, 9.2e3, 6.2e3] #list of flux constants per shower file
 BufferBox:             [ -250.0,250.0,-250.0,250.0,-250.0,250.0 ] #list of buffer box extensions to cryo volume in each dimension/dir (-x,+x,-y,+y,-z,+z)
 ShowerAreaExtension:   1000                 #amount to extend the shower area beyond the cryo dimensions
 RandomXZShift:         1000                 #amount to randomly shift shower start point in x & z [cm]
}
```

- **ShowerInputFiles**: is a list to allow mixing of multiple fluxes, with mixing constants controlled via **ShowerFluxConstants** list
- **Constant Mass Composition (CMC) model** implemented above
  - Forti, C. et al. Phys.Rev. D42 (1990) 3668-3689
- **Tuneable** parameters could be adjusted according to measurements

Running CorsikaGen Jobs

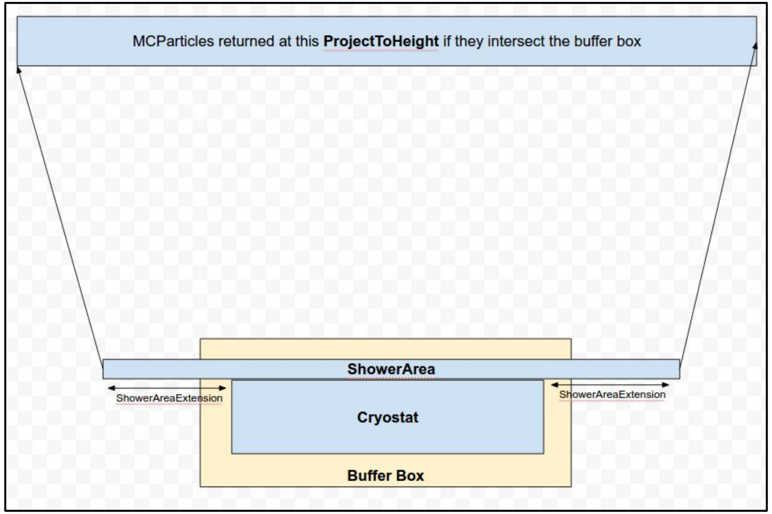
- As presented so far, CorsikaGen just expects the db files to be in the specified location specified in the **ShowerInputFiles** fcl parameter
- There is also an experiment specific bash script (PickFluxAndCopy.sh) to
    - Randomly choose a set of db flux files from a defined list
    - ifdh cp (~30 MB) to the local directory from pnfs
- MicroBooNE flux dbs stored in: `/pnfs/uboone/persistent/users/mibass/corsika/sqShowers/`
    - Not optimal, since the same flux files could be used for multiple experiments
    - Open to suggestions;
- Total flux db set (compressed) is 1.5 GB; 50 sets of files
    - To increase by a factor of 10 soon to increase statistical independence of resampling
- This plays well with project.py:

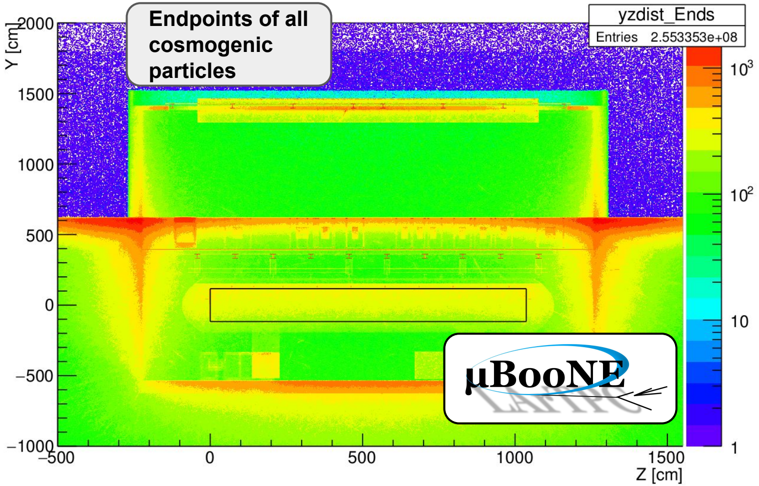
```
<stage name="CorsikaGen_CMC">
  <fcl>prodcosmics_CORSIKA_CMC.fcl</fcl>
  <initscript>PickFluxAndCopy.sh</initscript>
  ...
</stage>
```

Large Particle Fluxes and LArG4

- Keeping every MCParticle produced in cosmics events is expensive
- **BufferBox** changes, introduced last Summer, made events even larger
  - In MicroBooneE, peak memory usage increased from 4 GB to 8GB!
  - But increased acceptance, pre-g4, is good!

    Means we don't miss particles that scatter into TPC.
- But we are keeping MCParticles, post-g4, that don't do anything interesting->





LArG4 KeepParticlesInVolumes

- Added new fcl parameters to LArG4

```
physics.producers.largeant.KeepParticlesInVolumes: ["volTPCActive"] #keep MCParticles contained in these volumes
physics.producers.largeant.VolumesOffsetsX: [-128.175]             #World to local x translation
physics.producers.largeant.VolumesOffsetsY: [0.]                   #World to local y translation
physics.producers.largeant.VolumesOffsetsZ: [-518.4]               #World to local z translation
```

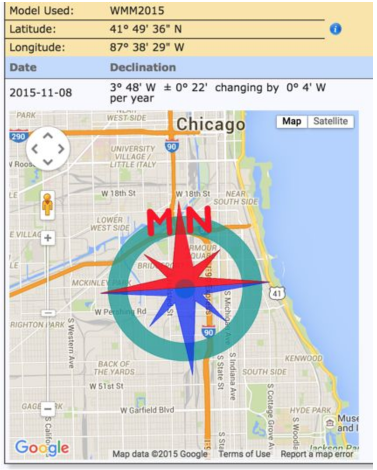
- Using Contains(...) method of geometry volumes to check every trajectory point of every MCParticle
  - `geom->ROOTGeoManager()->GetVolume(fKeepParticlesInVolumes[vol].c_str())->Contains(localpos)`
  - Keep if it intersects any of the kept volumes
  - Still will keep all MCParticles if KeepParticlesInVolumes is empty
  - Expensive, but worth it:

100 corsika events - LArG4	Peak memory	File size: Event Stream	<Running Time/event>
Modified	4.161 GB	6.7 GB	133.91 s
Standard	8.073 GB	20 GB	143.07 s

Summary and next steps

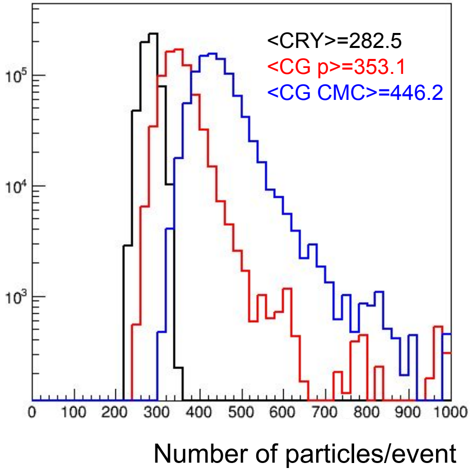
- Added **CorsikaGen module** to larsim
    - Resamples pre-generated CORSIKA flux databases to generate unique cosmic events
    - Flux files available; Looking for ways to make publicly available
- **Modified LArG4 module** to only keep MCParticles of interest
    - Reduces memory/disk/cpu usage for large cosmics events
    - Will push to develop soon...

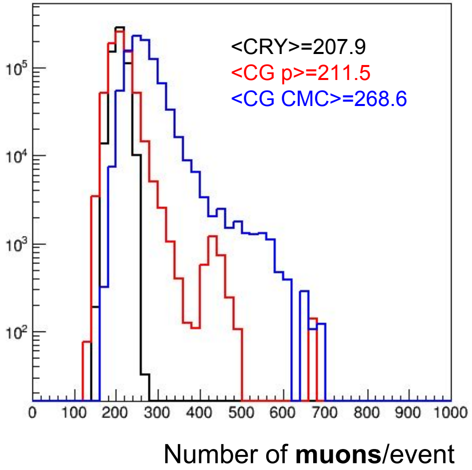
- Next:
    - Finish/push **KeepParticlesInVolumes** feature in LArG4_module.cc
    - Add **rotation** in CorsikaGen, to account for difference between z-axis and **magnetic** north->
    - Experiment specific **fcl** files and PickFluxAndCopy.sh scripts

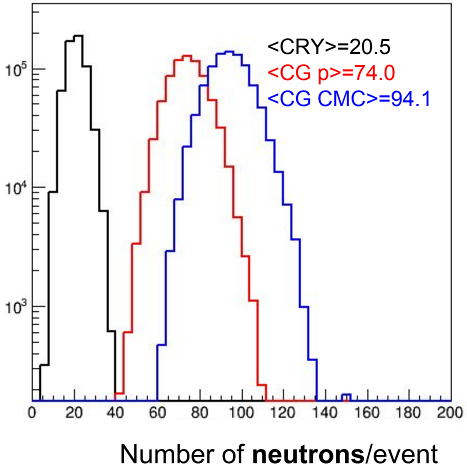


EXTRA SLIDES

CorsikaGen Testing







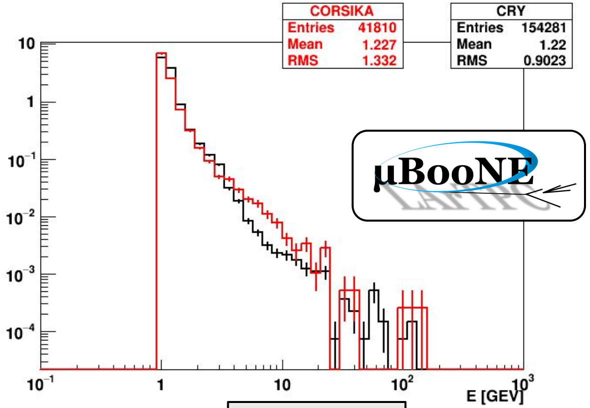
Tests so far show rates as expected.

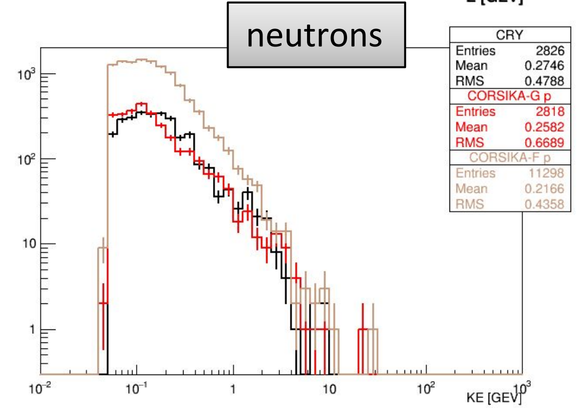
CORSIKA-FLUKA

- When we initially ran with GHEISHA we didn't see the increase in **neutron rate** that we expected
  - Expectation was based on Ziegler parameterization of the neutron flux
  - Pretty much agreed with CRY

- Previous experiments have noticed significant differences between **FLUKA** and **GHEISHA** hadronic interaction models

- CORSIKA can pass off to FLUKA **below 80 GEV** for hadronic interactions
  - "Just" had to compile/install FLUKA and put everything in its right place
  - And recompile CORSIKA
  - And reconfigure job submission scripts





Constant Mass Composition Model

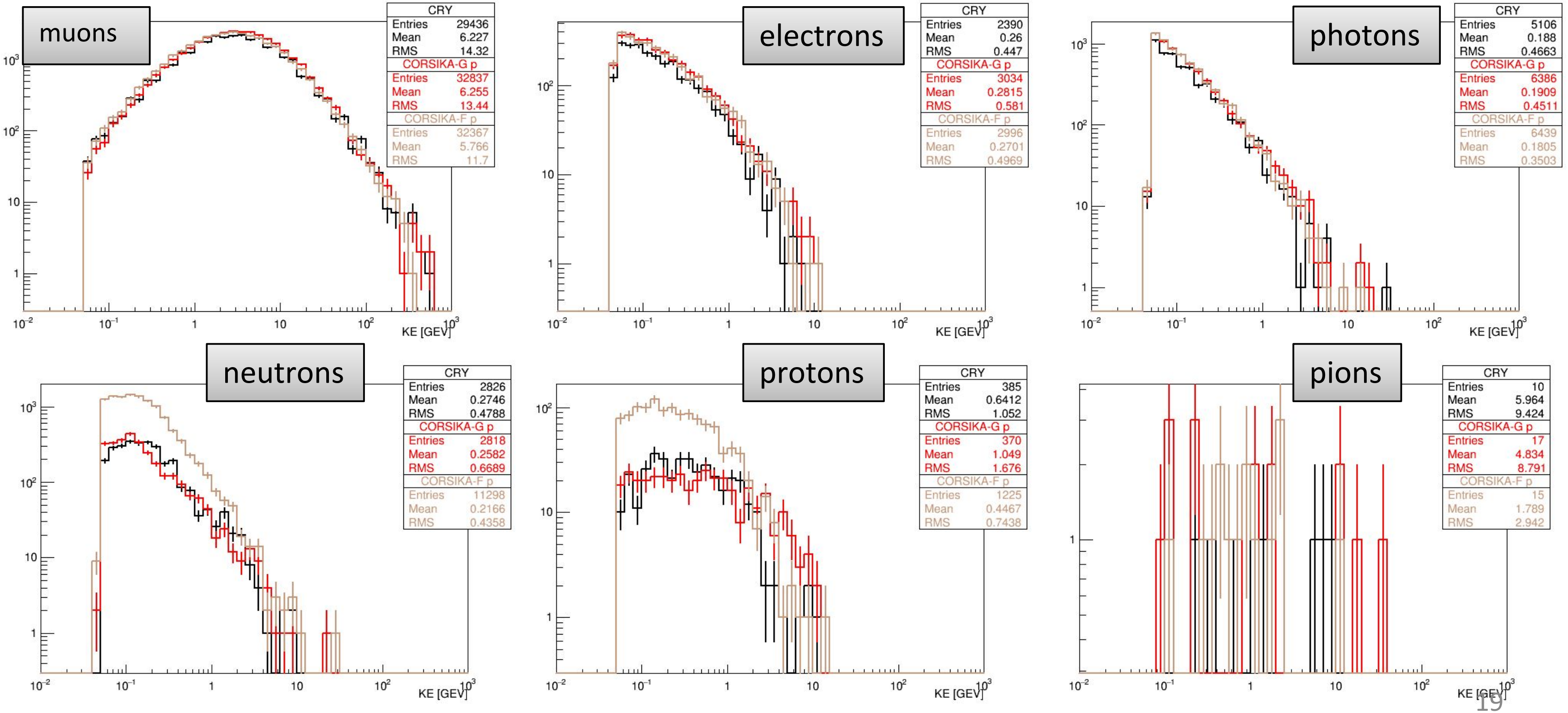
- A 5-component primary cosmic ray flux model called the **constant mass composition** (CMC) model was added in addition to the proton only runs
  - Treats the cosmic ray nuclei as five groups of mass:
    - protons, alpha, N, Mg, Fe
    - Based on treatment in: Simulation of atmospheric cascades and deep-underground muons, Forti et al, 1990
- Simulated in CORSIKA by running each **group** separately
- Flux of each nuclei type:
  - $\Phi_A(E)=K(E/1\ \text{GeV})^{-\gamma}$

Group	K	γ
p	1.72×10^4	2.71
α	9.20×10^3	2.71
CNO	6.20×10^3	2.71
Mg	9.20×10^3	2.71
Fe	6.20×10^3	2.71

CMC variables from: Simulation of atmospheric cascades and deep-underground muons, Forti et al, 1990

Results: CRY vs CORSIKA-GHEISHA vs CORSIKA-FLUKA

Particles through top of TPC, 9.6 seconds, Pre-g4

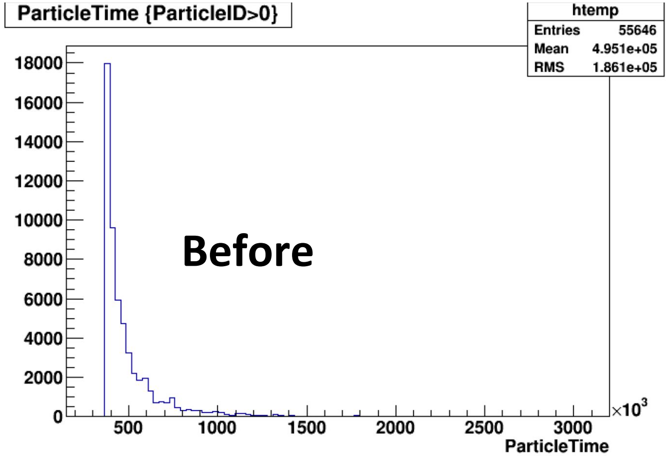


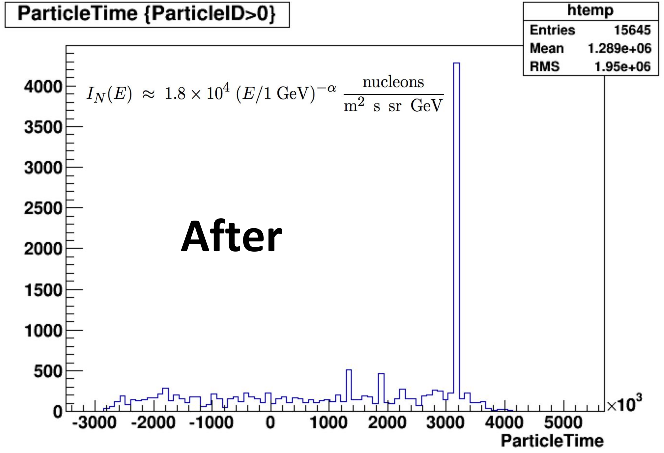
Template CORSIKA Input Card

```
RUNNR   _RUNNR_                     run number
EVTNR   1                           number of first shower event
NSHOW   _NSHOW_                     number of showers to generate
PRMPAR  _PRMPAR_                    particle type of prim. particle  (14=p)
ESLOPE  -2.7                        slope of primary energy spectrum
ERANGE  _ERANGELOW_ 100000          energy range of primary (GeV)
THETAP  0.  90.                     range of zenith angle (degree)
PHIP    -180.  180.                 range of azimuth angle (degree)
SEED    _SEED1_   0  0              seed for 1. random number sequence
SEED    _SEED2_   0  0              seed for 2. random number sequence
QGSJET  T   0                       QGSJET for high energy & debug level
QGSSIG  T                           QGSJET cross-sections enable
OBSLEV  228E2                       observation level (in cm)
MAGNET  19.066  50.628              Earth's mag. field at detector- Bx & Bz (22/09/2014)
HADFLG  0  0  0  0  0  2            flags hadr.interact.&fragmentation
ECUTS   0.05 0.05 0.05 0.05         energy cuts for particles
MUADDI  T                           additional info for muons
MUMULT  T                           muon multiple scattering angle
ELMFLG  T   T                       em. interaction flags (NKG,EGS)
STEPFC  1.0                         mult. scattering step length fact.
RADNKG  200.E2                      outer radius for NKG lat.dens.distr.
ARRANG  0                           rotation of array to north
ATMOD   1                           U.S. standard atmosphere (1-Linsley; 22-Keilhauer)
LONGI   T  20.   T  F               longit.distr. & step size & fit & out
ECTMAP  1.E2                        cut on gamma factor for printout
MAXPRT  100                         max. number of printed events
DIRECT  ./                          output directory
DATBAS  F                           write .dbase file
USER    mibass                      user
DEBUG   F  6  F  1000000            debug flag and log.unit for out
EXIT                                terminates input
```

corsikaConverter

- **corsikaConverter**: tool developed to process the output of CORSIKA into ROOT
  - Written by **Stefano Tognini**, a NOvA collab. Member
- Arranges showers into spills and arranges into a surface area
- **Problem**: CORSIKA does not distribute air shower primary particles in space or time
  - The clock and coordinate system are reset with each air shower
  - Showers are uncorrelated in space/time (particles within a shower ARE correlated of course)
  - Shower particles are distributed relative to shower start time and start position
- **Solution**: Distribute the primary showers in space/time based on measurements of the intensity of primary nucleons
  - This is what the corsikaConverter tool is doing
  - Added: buffer box to increase acceptance of multiple scattering muons
  - Added: arrangement of boxes in space by wrapping particle positions with respect to surface box





CORSIKA: Experimental comparison

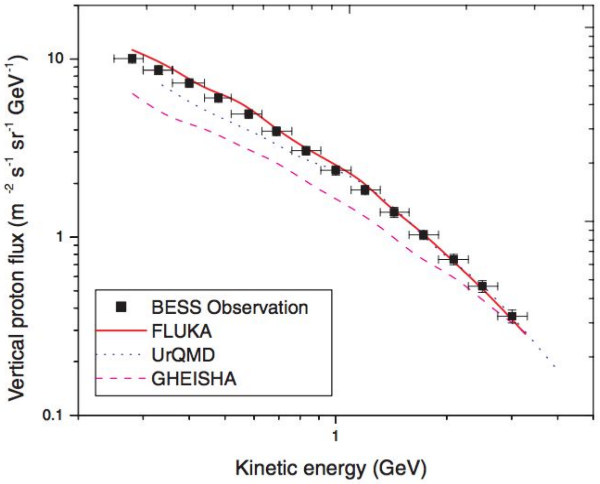


FIG. 5 (color online). The differential spectra of vertical proton at Mt. Norikura obtained by using three low energy models as is described in the text. The results of the BESS observation is also given for comparison.

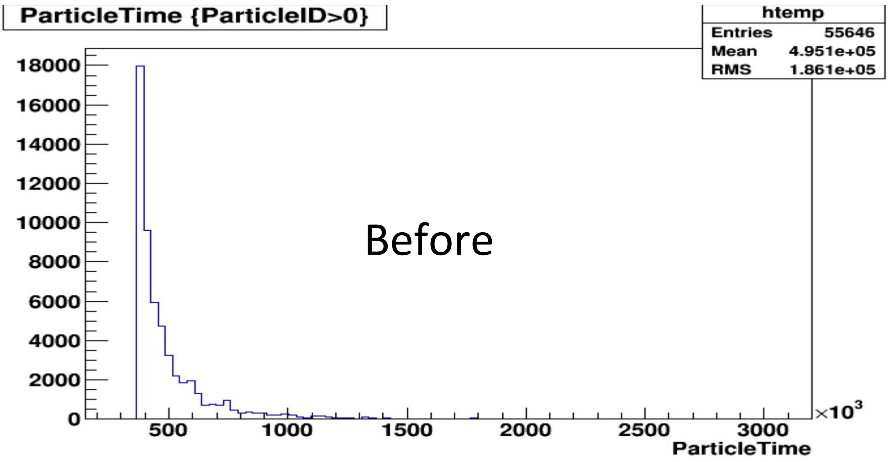
"Study of low energy hadronic interaction models based on BESS observed cosmic ray proton and antiproton spectra at medium high altitude", PHYSICAL REVIEW D 79, 114027 (2009)

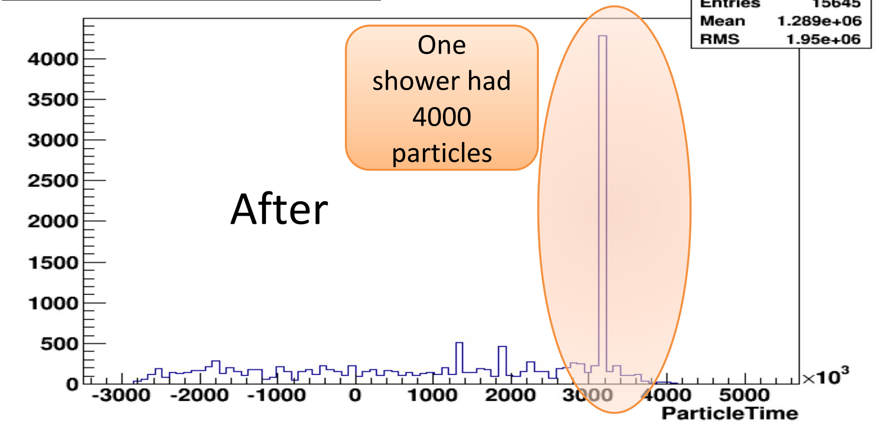
(2) The present analysis suggests that, among the three low energy models admissible in CORSIKA, only the FLUKA model produces a secondary proton spectrum in consistence with the BESS observation over the whole observed energy range. The model UrQMD works well at relatively higher energies, whereas the spectrum obtained with GHEISHA exhibits significant deviation from the measured spectrum; i.e., it generates too few protons in general.

What’s happening in time?

- Particle times come out of corsika relative to shower start
  - Clock resets for each air shower
- Tool arranges primary shower particles in time:
  - From PDG: “The intensity of primary nucleons in the energy range from several GeV to somewhat beyond 100 TeV is given approximately by”:

$$I_N(E) \approx 1.8 \times 10^4 \ (E/1 \ \mathrm{GeV})^{-\alpha} \ \frac{\mathrm{nucleons}}{\mathrm{m^2 \ s \ sr \ GeV}}$$





CorsikaGen Sampling

How to randomly sample from a table, based on a seed:

```
const TString kStatement("select shower,pdg,px,py,pz,x,z,t,e from particles where shower \
                          in (select id from showers ORDER BY substr(id*%f,length(id)+2)  \
                          limit %d) ORDER BY substr(shower*%f,length(shower)+2)");
```

%f’s are the random seed (0 to 1)

%d is the number of showers to pull