ProcessLevel testing

A. Dotti ; SLAC – SD/EPP/Computing

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Introduction

SLAC

Physics Validation at the process/model level is performed by developers with ad-hoc applications

General workflow:

1. initialize minimal set of G4 kernel classes
2. initialize model/process
3. book histograms
4. perform pseudo event-loop calling model/process main API
5. analysis saving histograms

- Only part of the code is common between validation test suites
- **Adding new histogram/analysis usually involves writing new code**
- **Only owner/developer is able to run test** (time consuming)
- Difficult to re-use G4 features: g4analsysis, MT, MPI, …

ProcessLevel testing

SLAC

A full Geant4 application:

- geometry: a very small single box of the target material
- physics: a physics list, ensure all processes are correctly setup
- kernel: build an application with MT and MPI support (for large statistics tests)

EventAction: force a single interaction with a given primary/energy on the target material

- implement direct call to physics processes in the EventAction, **replicating what is done in the SteppingManager**

Collect secondaries and analyse them in a general way

Analysis of secondaries

SLAC

For each interaction:

- get secondaries from process,
- order them by kinetic energy
- call analysis algorithm: a function performing some manipulation of secondary 4-momentum

Return value of analysis algorithm is used to fill histogram or ntuple column

- analysis algorithm defined in library, several already provided
- possible to extend with new algorithms

Configure everything via UI commnads. **No coding is required!**

Example: setting up interaction

SLAC

- Interested in study Bertini interactions at 1 GeV of π^- on Lead.
- Plot spectra of most energetic neutron.

Start application using FTFP_BERT physics list and use the following macro file:

```
###Energy of the primary
/processTest/primaryEnergy 1 GeV
##Target material from NIST database
/processTest/targetMaterial G4_Pb
##Primary name in standard G4 naming scheme
/processTest/primaryName pi-
#Process type. See G4ProcessType
## 2->Electromagnetic, 3->Optical, 4->Hadronic, 5->Decay
/processTest/processType 4
##Process sub-type, depends on process type (121 == Inelastic)
/processTest/processSubType 121

/run/initialize
```

Example Setting up histograms

SLAC

We can now define histograms/nutples to fill with quantities extracted from secondaries

```
##Specify libraries to be load containing analysis algorithms
/processTest/analysis/library libAnalysisDefault.so
## Book an histogram
# Id = 0
/analysis/h1/create nSpectraLeading EnergyNeutronLeading 100 0.001 1 GeV
/analysis/h1/setXaxis 0 "Ekin [GeV]”
/analysis/h1/setYaxis 0 "dXS/dE”
## fill histogram with id 0  with the kinetic
##               energy of the leading neutron and normalize histogram
/processTest/analysis/histo1D 0 Ekin 0 neutron true

/run/beamOn 1000
```

Specify name of library containing analysis algorithms

Example Setting up histograms

SLAC

We can now define histograms/nutples to fill with quantities extracted from secondaries

```
##Specify libraries to be load containing analysis algorithms
/processTest/analysis/library libAnalysisDefault.so
## Book an histogram
# Id = 0
/analysis/h1/create nSpectraLeading EnergyNeutronLeading 100 0.001 1 GeV
/analysis/h1/setXaxis 0 "Ekin [GeV]”
/analysis/h1/setYaxis 0 "dXS/dE”
## fill histogram with id 0  with the kinetic
##               energy of the leading neutron and normalize histogram
/processTest/analysis/histo1D 0 Ekin 0 neutron true

/run/beamOn 1000
```

Use g4analysis to create and setup histogram

Example Setting up histograms

SLAC

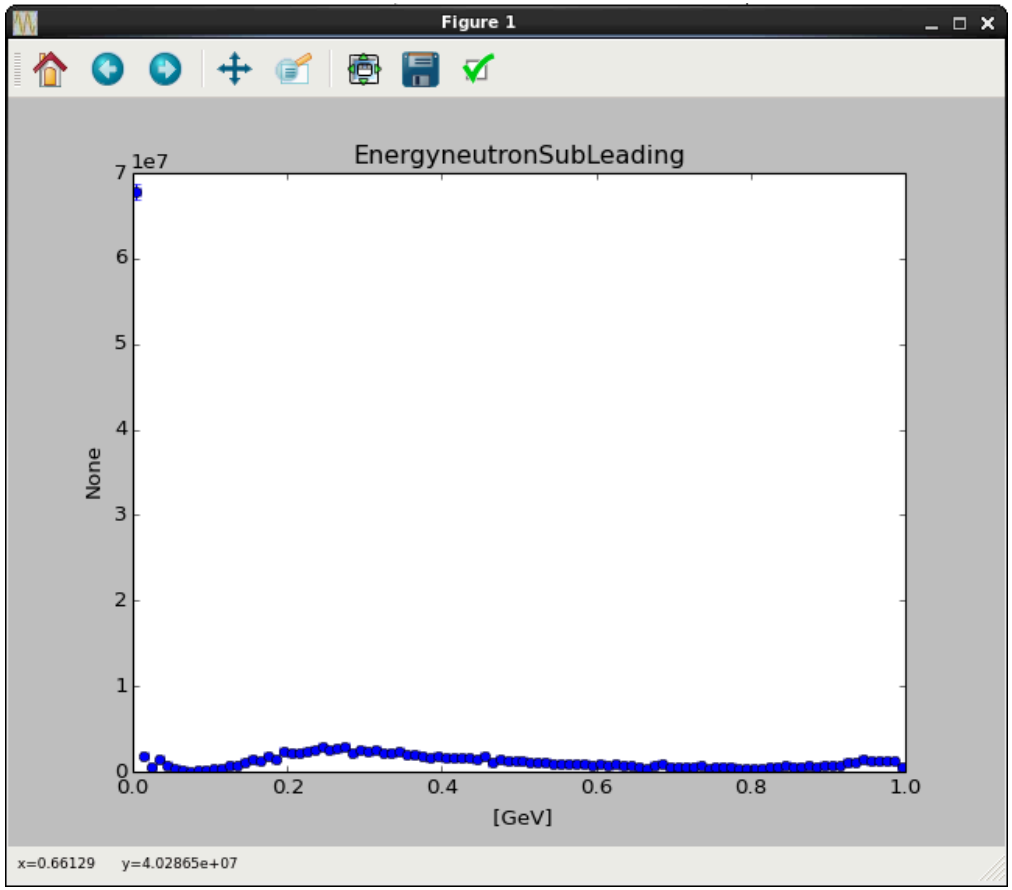
We can now define histograms/nutples to fill with quantities extracted from secondaries

```
##Specify libraries to be load containing analysis algorithms
/processTest/analysis/library libAnalysisDefault.so
## Book an histogram
# Id = 0
/analysis/h1/create nSpectraLeading EnergyNeutronLeading 100 0.001 1 GeV
/analysis/h1/setXaxis 0 "Ekin [GeV]”
/analysis/h1/setYaxis 0 "dXS/dE”
## fill histogram with id 0  with the kinetic
##              energy of the leading neutron and normalize histogram
/processTest/analysis/histo1D 0 Ekin 0 neutron true

/run/beamOn 1000
```

This reads: fill histogram id=0 with output of algorithm “Ekin”, for the most energetic neutron. Histogram should be normalized by cross-section

Example of output (CVS, ROOT, AIDA)



Bin at zero: no neutrons
Plot from CVS, plotted with matplotlib (python) plot utility provided for CVS (or use ROOT)

Why CSV format: simple to import into FNAL validation DB

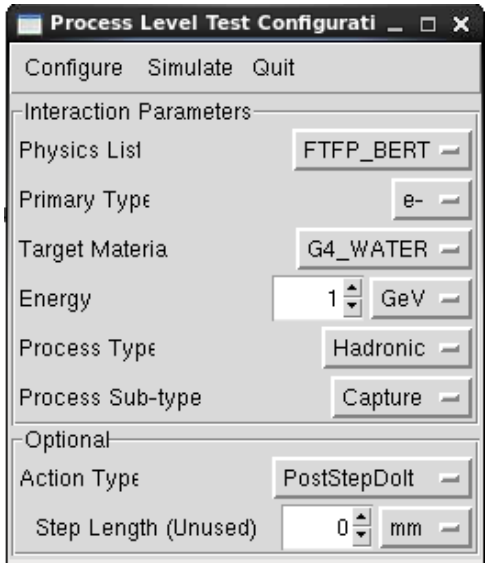
Simple plots like this will be used for regression testing

GUI

SLAC

Creating macro files with many histograms **can be tedious and error prone**

A simple GUI (still under development) can be used to **automatically generate macro files**

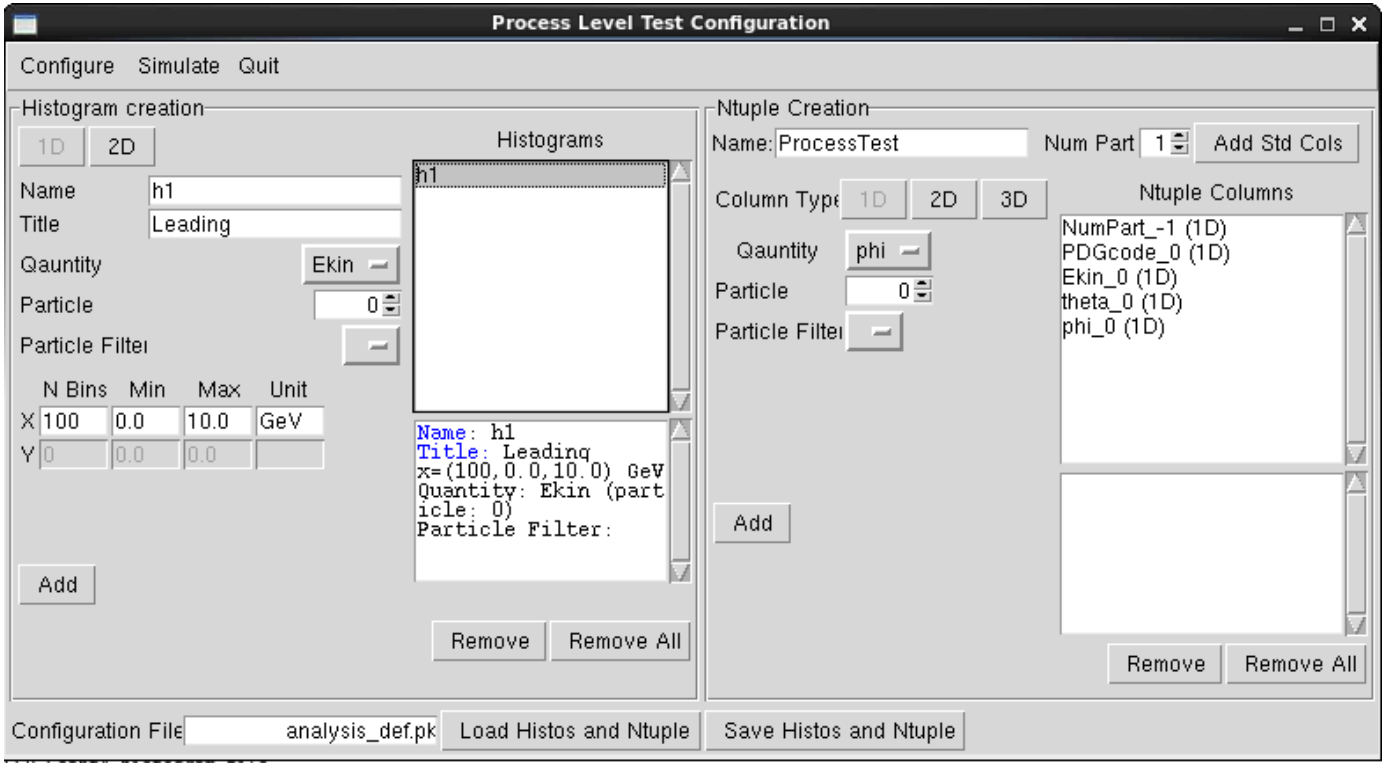


Configure interaction parameters

GUI

Creating macro files with many histograms **can be tedious and error prone**

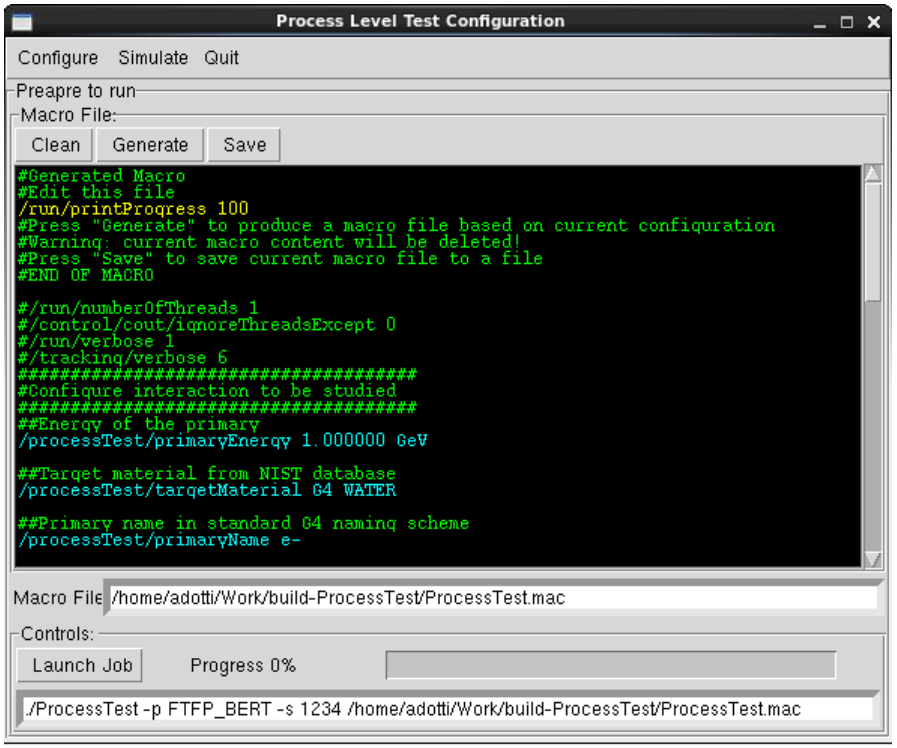
A simple GUI (still under development) can be used to **automatically generate macro files**



GUI

Creating macro files with many histograms **can be tedious and error prone**

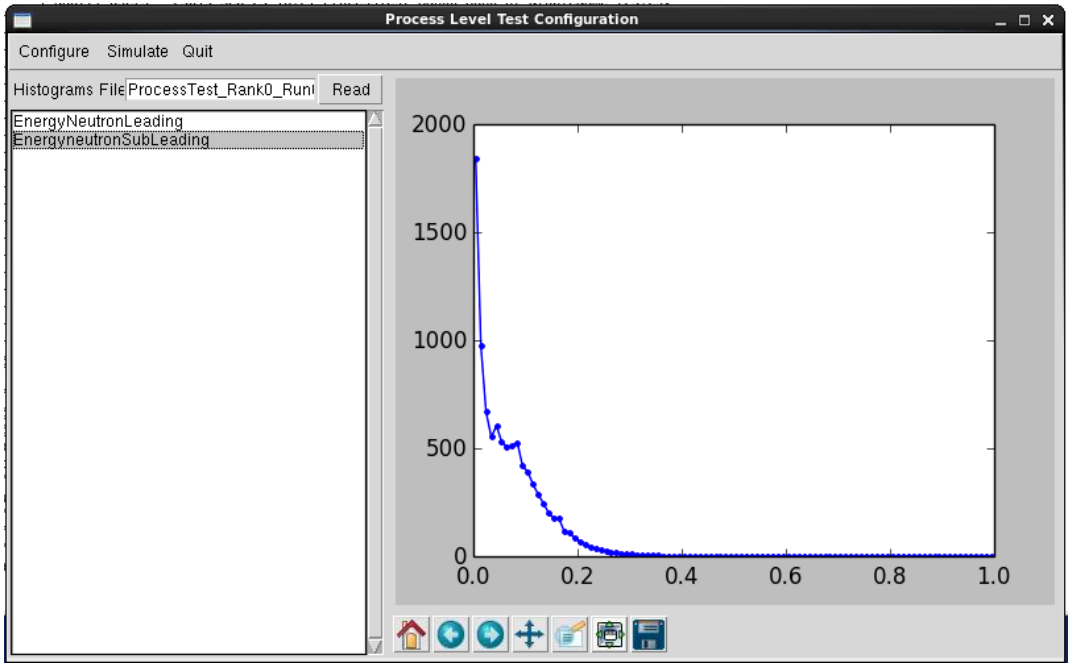
A simple GUI (still under development) can be used to **automatically generate macro files**



GUI

Creating macro files with many histograms **can be tedious and error prone**

A simple GUI (still under development) can be used to **automatically generate macro files**



Advanced examples

SLAC

Provided example for advanced analysis:

- Bertini validation data (as from its internal validation suite)
- Omega Zero experiment: Z. Phys. C - Particles and Fields 52, 397-405 (1991)
- HARP experiment: CERN-PH-EP-2008-010

Selected these because we already did in the past: verify output, different complexities, including lorentz-invariant histograms and configuration of analysis

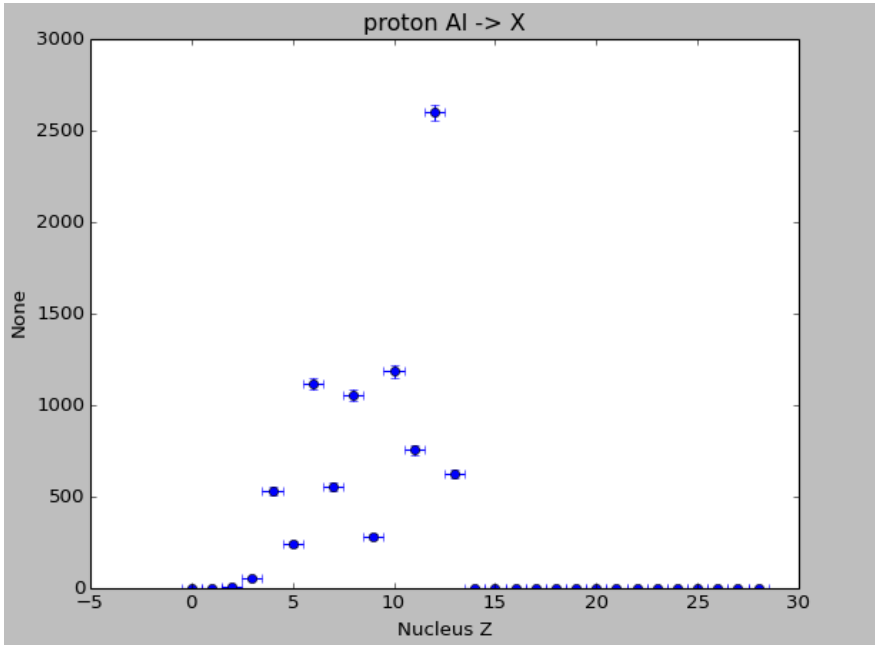
The following plots are examples created from ProcessTest application with the provided analysis plugins

- general enough for regression testing
- can be used as a basis for detailed experimental measurements

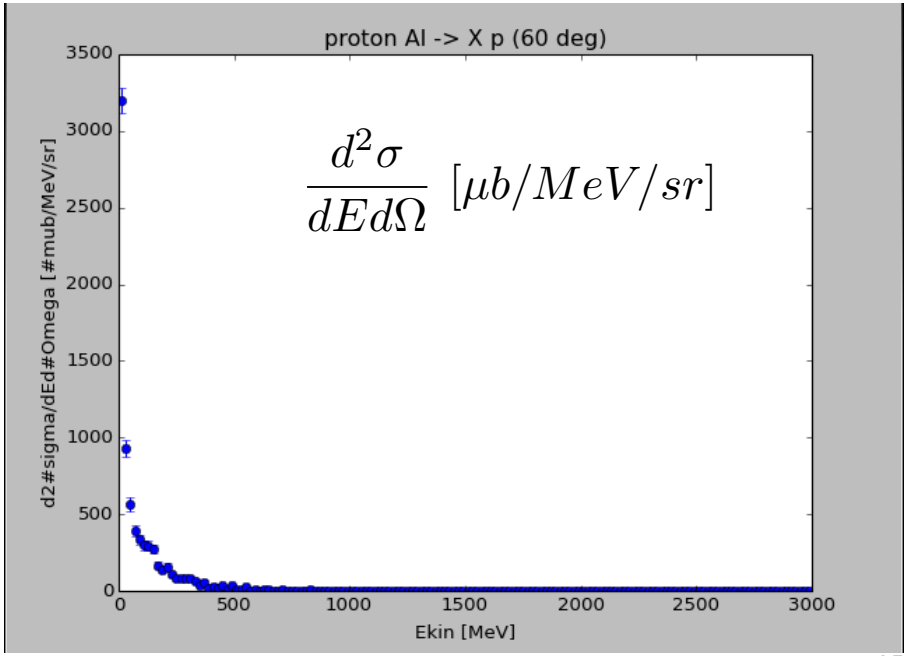
Bertini Validation: pAl→X @ p=4GeV/c

42 histograms, two examples shown here

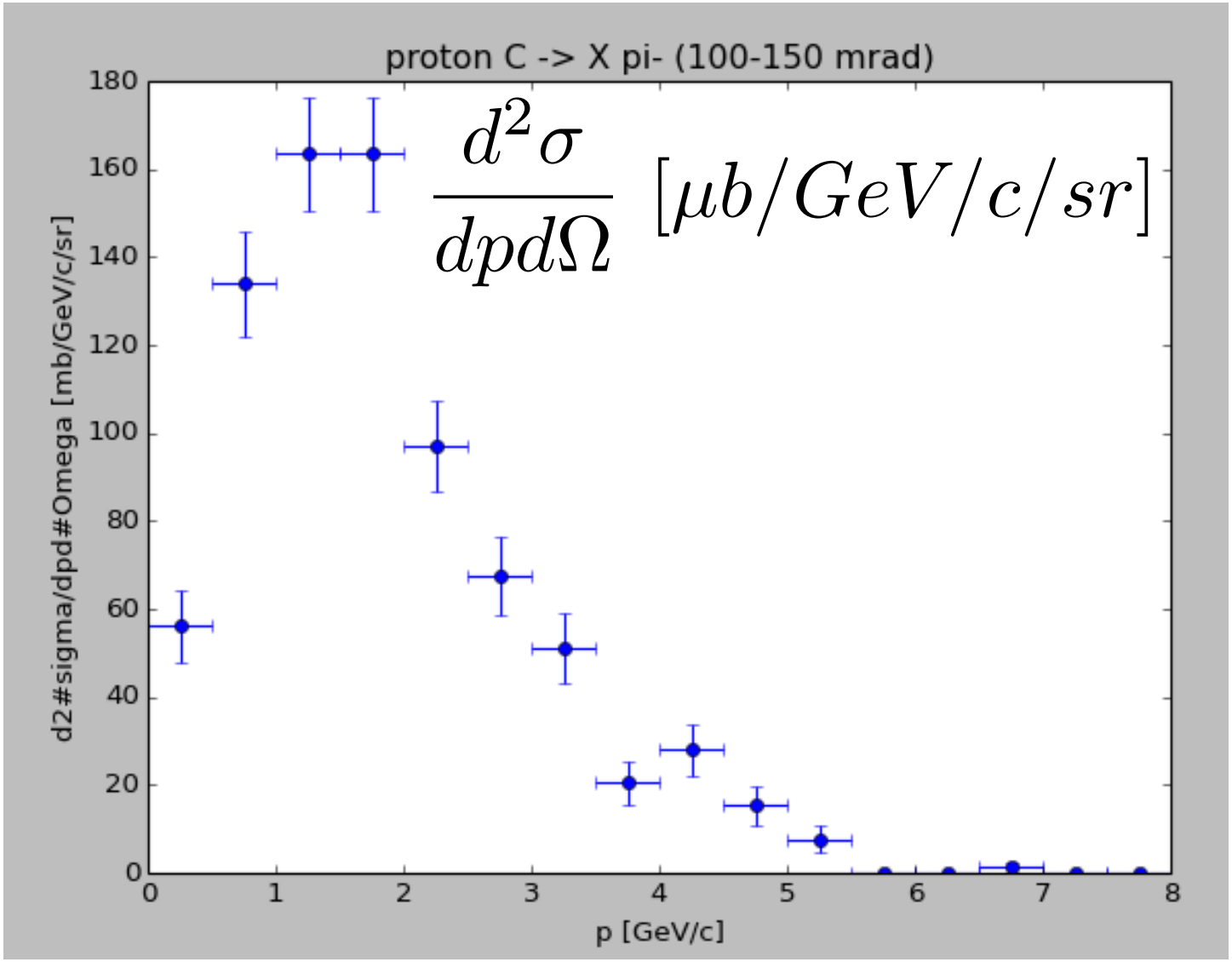
Number of residual nuclei
as a function of Z



Double differential cross-section proton
at 60 deg
Weighted histogram ($\Delta\ \theta$)



HARP Validation: forward validation pC->π^- @ p=12GeV/c



Extending Analysis

SLAC

- A new analysis algorithm can be added via a library loaded at run-time independently of analysis algorithm
- Example from Bertini validation plots. Calculate cos(θ) of secondary

```
#include "Analysis.hh”
#include "Plugin.hh"
using namespace Analysis;
struct cosTheta : public VAnalysis {
   G4double Analyse(const Particle& p )
            { return p.fourMom.cosTheta(); }
   DECLARENAME("cosTheta"); //Name used in macro files
   DECLAREDOC("cos(theta)");//Help documentation
};
//Library manifest
PLUGIN(cosTheta);
BEGINLIB()
  ADDPLUGIN(cosTheta);
ENDLIB()
```

Magic macros to automatically generate
code for plugin loading and content identification

Extending Analysis

SLAC

Different types of analysis can be created:

Input	Ouput
Single Secondary	Single value
Single Secondary	pair or triplet of values (e.g. both η,φ)
Vector of Secondaries	Single Value (e.g. total E_{kin})
Vector of Secondaries	pair of triplet of values. Also Vector of homogenous quantities (e.g. E_{kin} of all secondaries matching criteria)

Analysis algorithms can also return a weight associated with value to be used in filling histograms (e.g. bin size, for lorentz-invariant coefficients)

What is next

SLAC

- Use application to **automatically perform regression testing** of selected hadronic models (on Tachyon2 super-computer)
- Extend usage to "along-step" and "at rest" (code ready, need to test on realistic use-cases)
- Produce histogram of cross-sections associated to process
- **Integration with FNAL results DB**
- Application is feature complete, cleaning up code would be beneficial (manpower)

Contribution (poster) at IEEE/NSS 2015, paper will follow

Backup

A complete macro: interaction

```
#####################################
##Configure interaction to be studied
#####################################
##Energy of the primary
/processTest/primaryEnergy 1 GeV
##Target material from NIST database
/processTest/targetMaterial G4_Pb
##Primary name in standard G4 naming scheme
/processTest/primaryName pi-
##Process type. See G4ProcessType
## 2->Electromagnetic, 3->Optical, 4->Hadronic, 5->Decay
/processTest/processType 4
##Process sub-type, depends on process type.
##For hadronics is:
##  111->Elastic, 121->Inelastic, 131->Capture
##  141->Fission, 151->HadronAtRest, 152->LeptonAtRest
##  161->ChargeExchange, 210->Radioactive Decay
## For EM is:
##  1->Coulomb Scatt, 2->Ionisation
##  3->Bremss, 4->PairProd, 5->Annihilation
##  6->Annhil2MuMu, 7->Annhil2hh,
##  8->Nuclear Stopping, 10->MSC
##  11->Rayleigh, 12->PhotoElectric
##  13->Compton Scattering, 14->Gamma Conversion
##  15->GammaConv2MuMu, 21->Cherenkov,
##  22->Scintillation, 23->Synch rad, 24->Trans. Radiation
## Use -1 to disable. In such case first process to match
## Type will be used
/processTest/processSubType 121
##Optional.
##Select here which G4VProcessInterface one has to select
## AtRestDoIt, AlongStepDoIt or PostStepDoIt (default)
## Note that some processes may need a specific API to be
 ## be called. Set it up correctly
## If AlongStepDoIt is selected, by default step length is
## 1, use /process/stepLength command to change it
#/processTest/actionType AtRestDoIt
#/processTest/stepLength 1 mm
/run/initialize
```

A complete macro: analysis

```
########################################
## Analysis
########################################
## Set verbosity of analysis of interaction
#/processTest/analysis/verbose 4
##Specify libraries to be load containing analysis algorithms
/processTest/analysis/library libAnalysisDefault.so
##It is possible to pass to loaded libraries some configuration parameters
##with the following commands of the format: KEY VALUE
##Note that the set of all configurations pair will be passed to ALL libraries
##and that KEY must be unique
#/processTest/analysis/libraryConfiguration A A1
#/processTest/analysis/libraryConfiguration B B1
##Create histograms. Note that the ID (used later) will be
##strictly the order in which histograms are created
##Tip: if there are no particles to fill a histogram, it will be filled with
##zeros, it can be useful to set the histo limit to a small number to avoid that
# Id = 0
/analysis/h1/create nSpectraLeading EnergyNeutronLeading 100 0.001 1 GeV
/analysis/h1/setXaxis 0 "Ekin [GeV]”
/analysis/h1/setYaxis 0 "dXS/dE”
# Id = 1
/analysis/h1/create nSpectraSubLeading EnergyneutronSubLeading 100 0 1 GeV
# Id = 2
/analysis/h1/create SpectraLeading EnergyneutronSubLeading 100 0 1 GeV
```

A complete macro: analysis

```
## Fill an histogram with the output of an anlysis. Format:
## histo_id algorithm_Name particle_number particle_name norm
## Where:
## particle_number  if -1 consider all particles that match "particle_name" is specified
##                  if !=-1 consider the set of particles (eventually matching particle_name) ordered
##                  in descending momentum magnitude. Get that particle.
## particle_name    consider only particles matching name. Use "all" (default) for no selection
## norm             If true (default: false) normalize histograms for
##                  cross-section (in mub) and bin width
##
## ***** histo1D command fills a 1-D histogram with the single value returned by the "analysis"
## ***** histo2D command fills a 2-D histogram with the pair of values returned by the "analysis"
## ***** histo3D command fills a 3-D histogram with the triplet of values returned by the "analysis"
## Analysis algorithm can return also a vector of omogeneus {1,2,3}-D observables. In such a case, all values of the vector
## are used to fill the histogram. Note that this make sense only if particle_number==-1
### For a list of possible algorithm_Name see output of application with -h: it is printed at the beginning
## of the run.
##
## For a list of possible algorithm_Name see output of application with -h: it is printed at the beginning
## of the run.
#### For example: fill histogram with id 0  with the kinetic
##              energy of the leading neutron and normalize histogram
##              fill histogram with id 1 with kinetic energy of the sub-leading neutron
##              Fill histogram with id 2 with kinetic energy of leading particle and normalize histogram
/processTest/analysis/histo1D 0 Ekin 0 neutron true
/processTest/analysis/histo1D 1 Ekin 1 neutron
/processTest/analysis/histo1D 2 Ekin 0 all true
```

A complete macro: analysis (ntuple)

```
## Create Ntuple, parameters of command:
## algorithm_Name particle_number particle_name
## See previous section for meaning of particle_number and particle_name.
## ***** ntuple1D create a column (of type Double) for an algorithm returning a single value
## ***** ntuple2D create two columns (of type Double)  for an algorithm returning a pair of values
## ***** ntuple3D create three columns (of type Double)  for an algorithm returning a triplet of values
## Ntuple column names are created as a combination of the parameters
## (in case of pairs of triplets add a prefix _X,_Y,_Z).
## Ntuple should be opened (with an optional name) and closed: see first and last commands in this sequence
## For example: fill first columns with PDG code of leading, sub- and sub-sub- leading secondaries
##              fourth and fifth columns with kinetic energies of leading and sub-leading
##              sixth column with energy deposited along step (warning here particle_number has no meaning
##              simply skip it)
##              seventh and eight column with leading and sub-leading neutrons kin energy
##              last column with number of neutrons (note that here -1 is required for particle number)
/processTest/analysis/open NtupleName
/processTest/analysis/ntuple1D PDGcode 0
/processTest/analysis/ntuple1D PDGcode 1
/processTest/analysis/ntuple1D PDGcode 2
/processTest/analysis/ntuple1D Ekin 0
/processTest/analysis/ntuple1D Ekin 1
/processTest/analysis/ntuple1D edep
/processTest/analysis/ntuple1D Ekin 0 neutron
/processTest/analysis/ntuple1D Ekin 1 neutron
/processTest/analysis/ntuple1D NumPart -1 neutron
/processTest/analysis/close
#########################################
## Perform Analysis
#########################################
##Limitation: one job must contain one file for ntuples to work correctly
/run/beamOn 10000
```