GENIE Validation

Introduction & Status

Costas Andreopoulos[1,2]

[1]University of Liverpool, [2]STFC Rutherford Appleton Laboratory

December 13, 2013

Outline

- An introduction to the GENIE validation programs
  - Had very little time and, probably, this is a fairly incomplete talk
- Focus on procedural aspects
  - My main goal is to highlight areas that need work

Delivering Software

Extract from 'Continuous Delivery - Reliable Software Releases Through Build, Test and Deployment Automation, by J.Humble and D.Farley, ISBN-13: 9780321601919':

You have a critical bug in production. It is losing money for your business every day. You know what the fix is: A one-liner in a library that is used in all three layers of your three-tier system, and a corresponding change to one database table. But the last time you released a new version of your software to production it took a weekend of working until 3 A.M., and the person who did the deployment quit in disgust shortly afterward. You know the next release is going to overrun the weekend, which means the application will be down for a period during the business week. If only the business understood our problems.

Critical question: **How long would it take your organization to deploy a change that involves just one single line of code?** (Cycle time)

GENIE Cycle time

Measured in weeks. Release process not repeatable or reliable.
Reasons:

- Process was not automatic.
  Required one to run many different bits of code, examine various outputs for errors, assemble and re-format outputs to be used as inputs in later stages of the validation process,... Work already in progress to address this.
- Some of the validation programs could not be trusted.
  In the preparation of the last major release (v2.8.0), more time was spent worrying about the workings of the validation programs than worrying about the workings of the core GENIE simulation. Many calculations were done by hand to cross-check validation programs that could not be fully trusted (hadronization). Other checks (INTRANUKE) were not run as part of the same effort but, rather, relied on Steve.
- CPU resources: I have plenty! However, in the past 3 years, with my students, I have spent a quarter of million CPU$\times$days (!) running one of the official T2K oscillation analyses (which has priority so, for large periods of time, I have very little CPU r esources for GENIE work).

Definitions: Validation vs Tuning

Validation

Basic question: Is GENIE output sensible?

- Not just the *physics* output, everything (eg file integrity, etc)
- Does not need to always involve "experimental data" (eg sanity checks, conservation principles, sum rules, scaling laws,...)
- Does not need quantitative estimates of data/MC agreement.
  - Sometimes difficult to obtain (various issues with experimental data).
  - Eyeballing sufficient.
  - However, prediction errors due to MC statistics and generator-level uncertainties a necessary ingredient.

Tuning

Anwsers more quantitative questions. Hypothesis testing, point and interval estimation (which is the best model to use, what are the best model parameter values, what is the error on the parameters).
Will discuss more in later talks.

GENIE Validation

- **MC**
  - Can I even run GENIE and get outputs? Can I read the outputs?
  - Do I conserve energy, charge, ...?
  - Do I generate bogus (unphysical, incomplete, ...) events?
  - Are there errors and warnings in log files?
  - Does event generation fail (consistently) for specific valid kinematic points ("output cross-section" $!ne$ input cross-section)?
  - If I generated $\nu_\mu + Fe^{56}$ event with vertex posistion $\overrightarrow{r}$, does my geometry really have a Fe^{56} volume in $\overrightarrow{r}$?
  - Does my reweighting respect unitarity (if it was meant to)?
  - Are we efficient?
  - ...
- **MC vs MC**
  - How do my cross-section predictions compare against the previously released ones?
  - How do my event kinematic?
- **MC vs Data**
  - Does any of the above describe reality?

GENIE Validation: Vision

A fully automated procedure:

https://genie.hepforge.org/trac/browser/trunk/src/scripts/production/batch/run_genie_validation.pl

```
laptop> ssh candreop@linux.pp.rl.ac.uk
ralppd_tier2> cd /software/GENIE/builds/SL5_64bit/
ralppd_tier2> source genie_v?.?.?_release_candidate_setup
ralppd_tier2> cd $GENIE/src/scripts/production/batch/
ralppd_tier2> perl run_genie_validation.pl –version v?.?.?
ralppd_tier2> exit
```

- Submit batch jobs, as needed, typically in many steps
  Output from step N is input in step N+1
- Check (log files, data file integrity, etc) and organize outputs
- Run validation checks
  Produce summary reports and plots (MC vs ref. MC, MC vs data)
- Flag potential issues and produce diagnostic information

GENIE Validation: Goals

- Run the full validation suite continuously...
  - unless no new commits
- Aim at a cycle time of less than 1 week.

The inner workings ...

What happens when you:

```
laptop> ssh candreop@linux.pp.rl.ac.uk
ralppd_tier2> cd /software/GENIE/builds/SL5_64bit/
ralppd_tier2> source genie_v?.?.?_release_candidate_setup
ralppd_tier2> cd $GENIE/src/scripts/production/batch/
ralppd_tier2> perl run_genie_validation.pl –version v?.?.?
```

Start by creating a directory structure for oututs

some_dir/
some_dir/**xsec**/
some_dir/**mctest**/
some_dir/**mctest/ghep**/
some_dir/**mctest/gst**/
some_dir/**xsec_validation**/
some_dir/**xsec_validation/ghep**/
some_dir/**xsec_validation/gst**/
some_dir/**hadronization**/
some_dir/**hadronization/ghep**/
some_dir/**intranuke**/
some_dir/**intranuke/ghep**/
some_dir/**reptest**/
some_dir/**reptest/ghep**/
some_dir/**flux**/
some_dir/**geom**/
some_dir/**reweight**/
some_dir/**reports**/

Calculating neutrino-nucleon cross-section tables

Basic input for nuclear cross-section calculations and event simulation.
Data tables are used for the construction of cubic splines.
For all processes, all (anti)neutrino flavours, and for free proton, neutron targets. Outputs in XML and ROOT format.

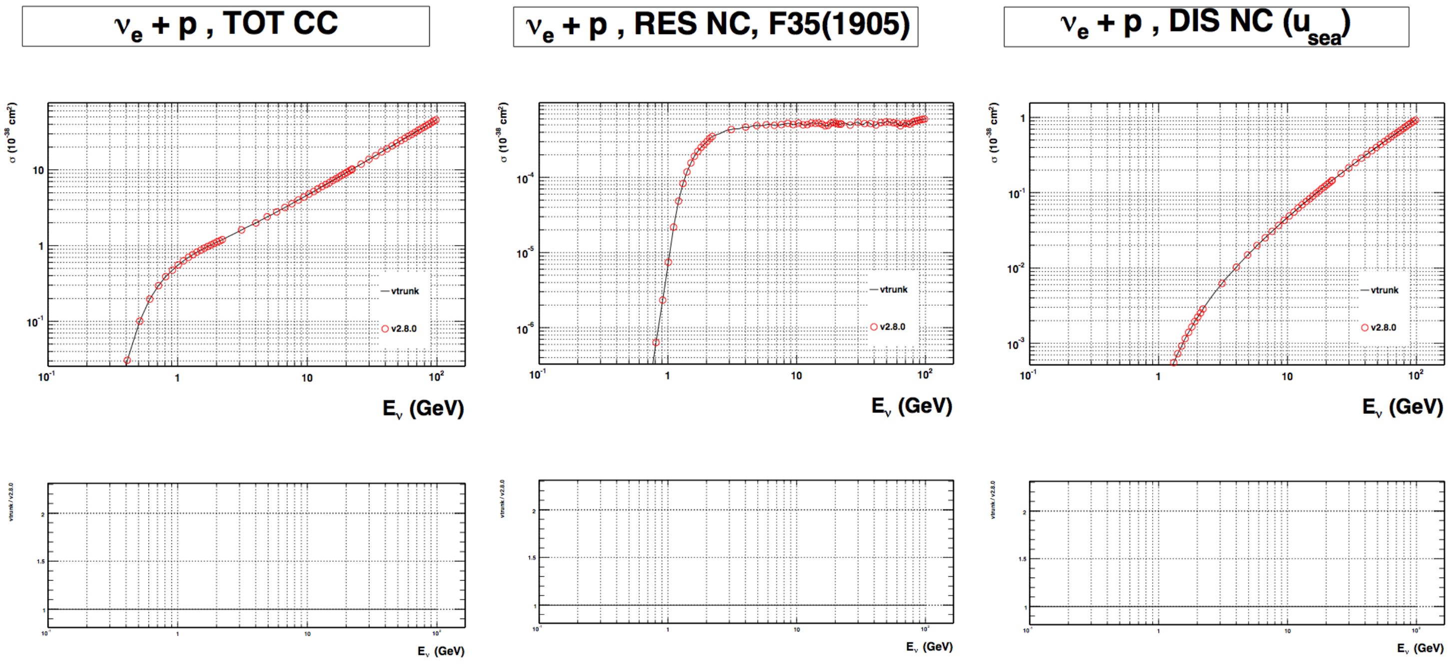
1. Submit all necessary jobs.
   - Using 'perl submit_vN_xsec_calc_jobs.pl –xsplset all ...'
   - Submitted jobs use gmkspl cross-section calculation app.
   - Outputs in XML.
2. Monitor job status till all jobs have finished.
3. Check log files for errors or warnings. Stop if any.
4. Check that all submitted jobs gave outputs (number of input PBS scripts = number of output XML files). Stop if outputs are missing.
5. Merge all XML files (from the various batch jobs) into one.
6. Move files to standard location.
7. Convert cross-section splines to ROOT format too.

Comparing cross-sections with reference calculations

The calculated neutrino-nucleon cross-sections are compared against a reference calculation (eg results from the previous official release).

Feed data to gvld_xsec_comp which dumps out a ∼600 page document.

Examples:



Caveat: Need to add code to check the comparison plots and generate a report.

Calculating neutrino-nucleus cross-section tables

Previous output is now input. Data tables are used for the construction of cubic splines. Neutrino-nucleus cross-section is basic input for simulating events for nuclear targets. Various neutrino-nucleus combinations defined (e.g. the ones relevant for generating events in the T2K 280m off-axis detector, the ones relevant for GENIE validation checks, ...)

Procedure:

1. Submit jobs
   - Using 'perl submit_vA_xsec_calc_jobs.pl –config-file genie_test.list ...'
   - Submitted jobs use gmkspl cross-section calculation app.
   - Outputs in XML.
2. Monitor job status till all jobs have finished.
3. Check log files for errors or warnings. Stop if any.
4. Check that all submitted jobs gave outputs (number of input PBS scripts = number of output XML files). Stop if outputs are missing.
5. Merge all XML files (from the various batch jobs) into one.
6. Move files to standard location.

Running standard neutrino MC jobs

Once all cross-section tables are sucessfully produced, a set of standardized MC samples are generated:

Run num.	Num. of events	Initial state	Energy (GeV)	Processes enabled
1000	100k	ν_μ + n	0.5	all
1001	100k	ν_μ + n	1	all
1002	100k	ν_μ + n	5	all
1003	100k	ν_μ + n	50	all
1101	100k	numubar + p	1	all
1102	100k	numubar + p	5	all
1103	100k	numubar + p	50	all
2001	100k	ν_μ + Fe56	1	all
2002	100k	ν_μ + Fe56	5	all
2003	100k	ν_μ + Fe56	50	all
2101	100k	numubar + Fe56	1	all
2102	100k	numubar + Fe56	5	all
2103	100k	numubar + Fe56	50	all
9001	100k	ν_μ + Fe56	5	DIS charm
9002	100k	ν_μ + Fe56	5	QEL charm
9101	100k	ν_μ + Fe56	2	COH CC+NC
9201	100k	ν_e + Fe56	1	ve elastic
9202	100k	ν_μ + Fe56	1	ve elastic
9203	50k	ν_μ + Fe56	20	IMD
9204	50k	nuebar + Fe56	20	IMD (annihilation)

Running standard neutrino MC jobs

Procedure:

1. Submit jobs
   - Using 'perl submit_standard_neutrino_mc_test_jobs.pl –run all ...'
   - Submitted jobs use gevgen event generation app.
   - Full event trees (GHEP) and summary n-tuples (GST format) obtained using the gntpc file converter app.
2. Monitor job status till all jobs have finished.
3. Check log files for errors or fatal messages. Stop if any.
4. Check that all submitted jobs gave outputs (number of input PBS scripts = number of output XML files). Stop if outputs are missing.
5. Move files to standard location.

Run sanity checks on all event files

Once the standard neutrino MC samples are successfully generated, we run a series of sanity checks using the gvld_sample_scan app.

Running gvld_sample_scan with the following options:

–add-event-printout-in-error-log
–event-record-print-level 2
–max-num-of-errors-shown 10
–check-energy-momentum-conservation
–check-charge-conservation
–check-for-pseudoparticles-in-final-state
–check-for-off-mass-shell-particles-in-final-state
–check-for-num-of-final-state-nucleons-inconsistent-with-target
–check-vertex-distribution
–check-decayer-consistency

Finds problems, writes out example problematic events and summaries in text files. Additional checks can be easily added in gvld_sample_scan.
Caveat: No automated check of summary output files yet.

Compare with reference samples

If no problem was identified running the sanity checks, the MC samples are compared against reference samples (from the last official release).

Events trees are converted to simple flat n-tuple using gntpc. The n-tuples are passed to gvld_sample_comp which dumps out a ~100-150 page document per sample, with various comparisons:

- number of events (per event type)
- distributions of kinematical variables (W,Q^2,x,y) for various event categories
- multiplicity and momentum distributions of various hadrons
  - for primary (vtx) hadronic system & final state hadronic system
  - for all events or for specific event categories

Output in postscript format.

Caveat: Comparisons between samples with the same statistics sometimes difficult to interpret. Would be best to compare samples that correspond to the same exposure. Shape-only comparisons could be added where/if needed.

Caveat: Need to add automated checks for the degree of compatibility of various distributions (current vs reference sample) to help flaging problems.

Repeatability test

1. Submit 3 independent MC jobs
   - $\nu_\mu + Fe^{56}$
   - 1/E flux flux from 100 MeV to 50 GeV
     - $\sim$ flat event rate
2. Run the gvld_repeatability_test app for samples 1 & 2, and 1 & 3. The app loops over the two input samples and, for each event
   - compares all properties with event-wide scope (vtx, cross-section,...)
   - compares *all* initial, intermediate and final-state particles (PDG code, status code, FSI code, mother/daughter indices, 4-momentum, position, polarization...)
3. If any difference is seen, the events are written out

Caveat: Need to add code to check the gvld_repeatability_test output and make a decision whether all was OK.

Cross-section validation

If GENIE passes all previous simple checks, starting checking predictions against data. Cross-sections are checked first. Procedure:

1. Submit jobs
   - Using ‘perl submit_neutrino_xsec_validation_mc_jobs.pl ...‘
   - Submitted jobs use gevgen event generation app.
   - Outputs in GHEP/ROOT and GST/ROOT format.
2. Monitor job status till all jobs have finished.
3. Need to check log files for errors or warnings. Stop if any.
4. Need to Check that all submitted jobs gave outputs. Stop if outputs are missing.
5. Move files to standard location.
6. Generate XML file lists to load event samples to validation program.

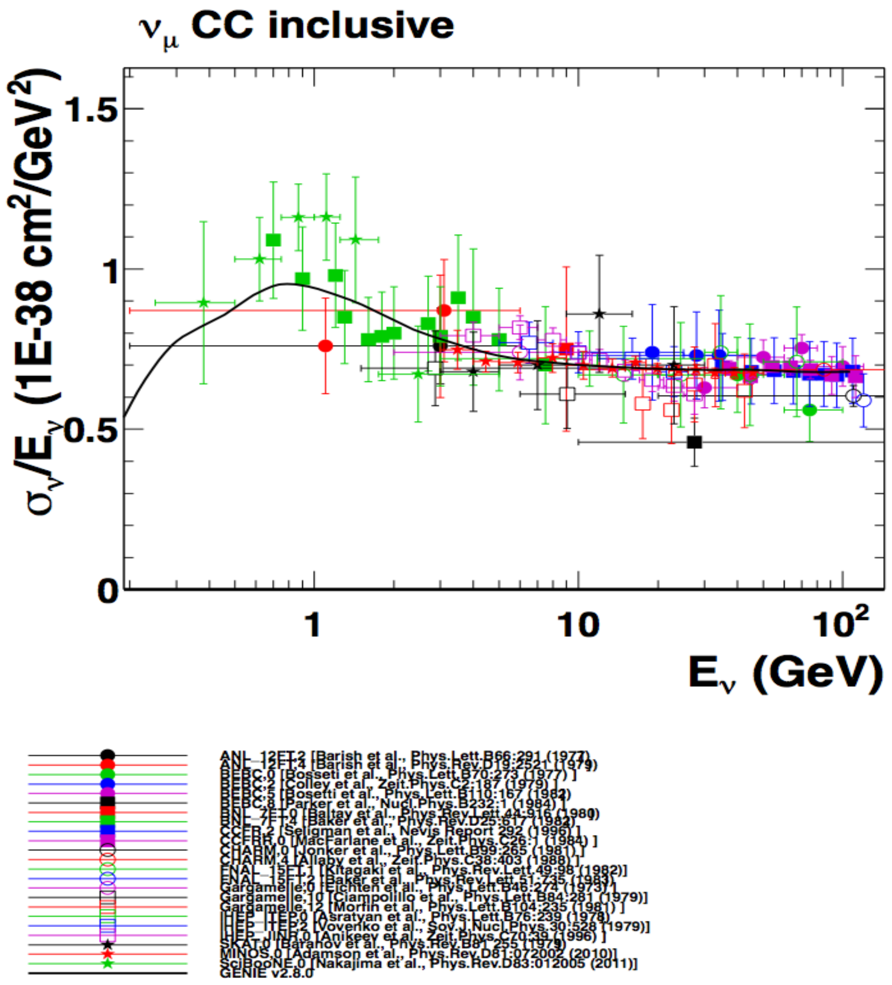
Cross-section validation

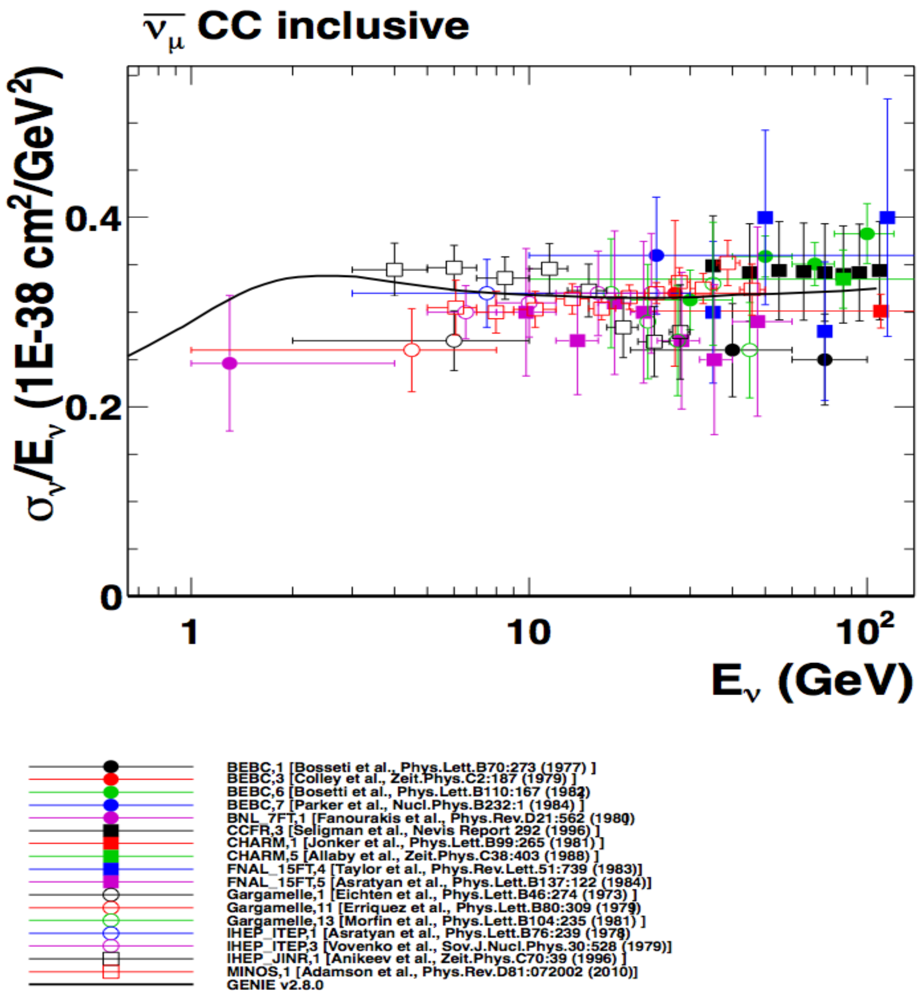
The XML file list for the current version and, potentially, corresponding file lists with reference samples (e.g. from previous official releases) are passed on to the gvld_nu_xsec app. The app compares GENIE with a large fraction of the world data and can also generate error envelopes for many GENIE predictions. Different comparison specified by name:

numuCC_all	numuCCQE_all_12C_nuclear	numuNCcohpi0_Al27
numubarCC_all	numubarCCQE_all	numuNCcohpi0_Si30
numuCC_lowE	numubarCCQE_deuterium	numuCCcohpip_Si30
numubarCC_lowE	numubarCCQE_heavy_target	numubarCCcohpim_Si30
numuCC_highE	numubarCCQE_nomad_nucleon	numuCC_dilepton_ratio_worldavg
numubarCC_highE	numubarCCQE_nomad_nuclear	numubarCC_dilepton_ratio_worldavg
numuCC_minos	numuCCppip	numuCC_charm_ratio_worldavg
numubarCC_minos	numuCCnpip	numuCC_dilepton_cdhs
numuCC_sciboone	numuCCppi0	numuCC_dilepton_nomad
r_minos	numuCCn2pip	numuCC_dilepton_e744_e770
numuCCQE_all	numuCCppippi0	numuCC_dilepton_e744
numuCCQE_deuterium	numuCCppippim	numuCC_dilepton_fnal15ft
numuCCQE_heavy_target	numuCCpi0_numuCCQE_k2k	numuCC_dilepton_gargamelle
numuCCQE_nomad_nucleon	numuNCcohpi0_Ne20	
numuCCQE_nomad_nuclear	numuCCcohpip_Ne20	
numuCCQE_miniboone_nuclear	numubarCCcohpim_Ne20	

Cross-section validation

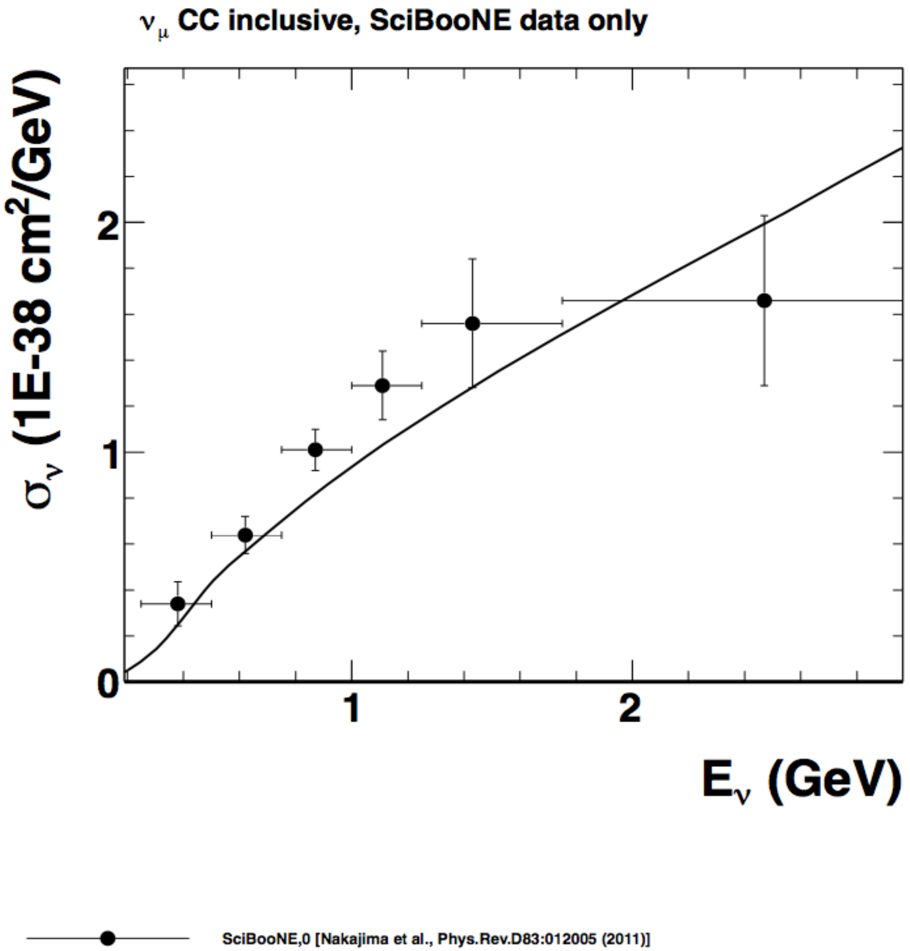
Example plots:

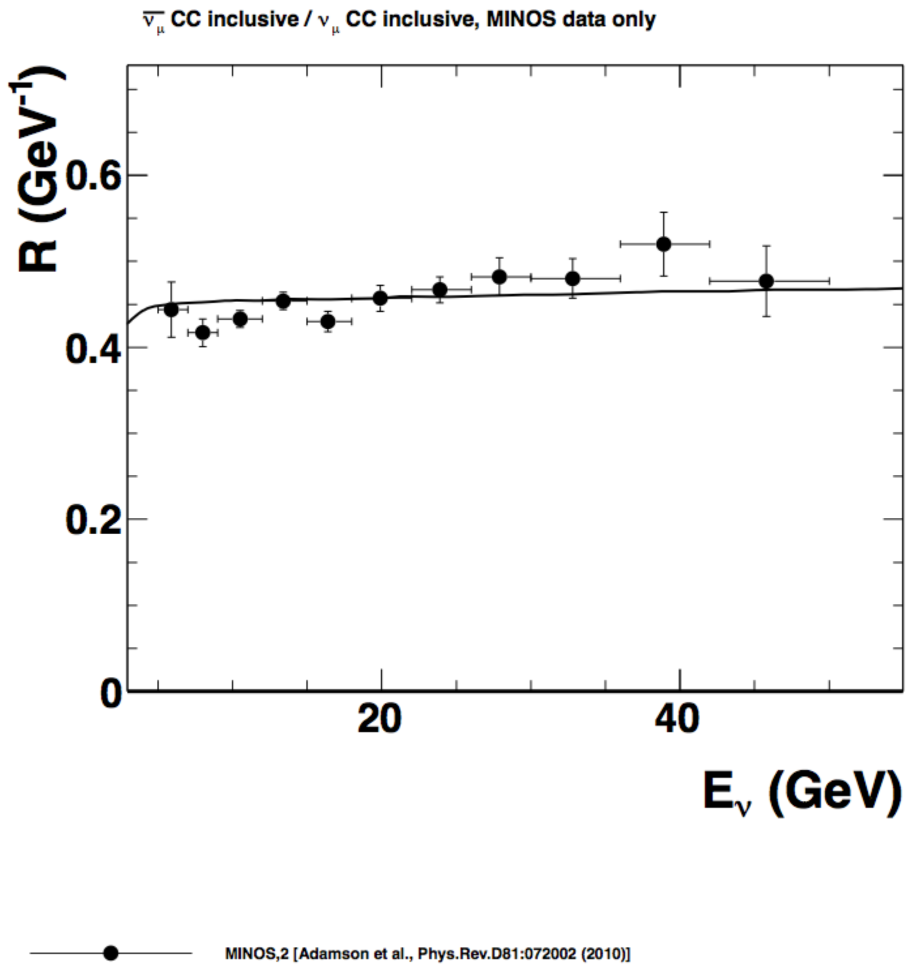




Cross-section validation

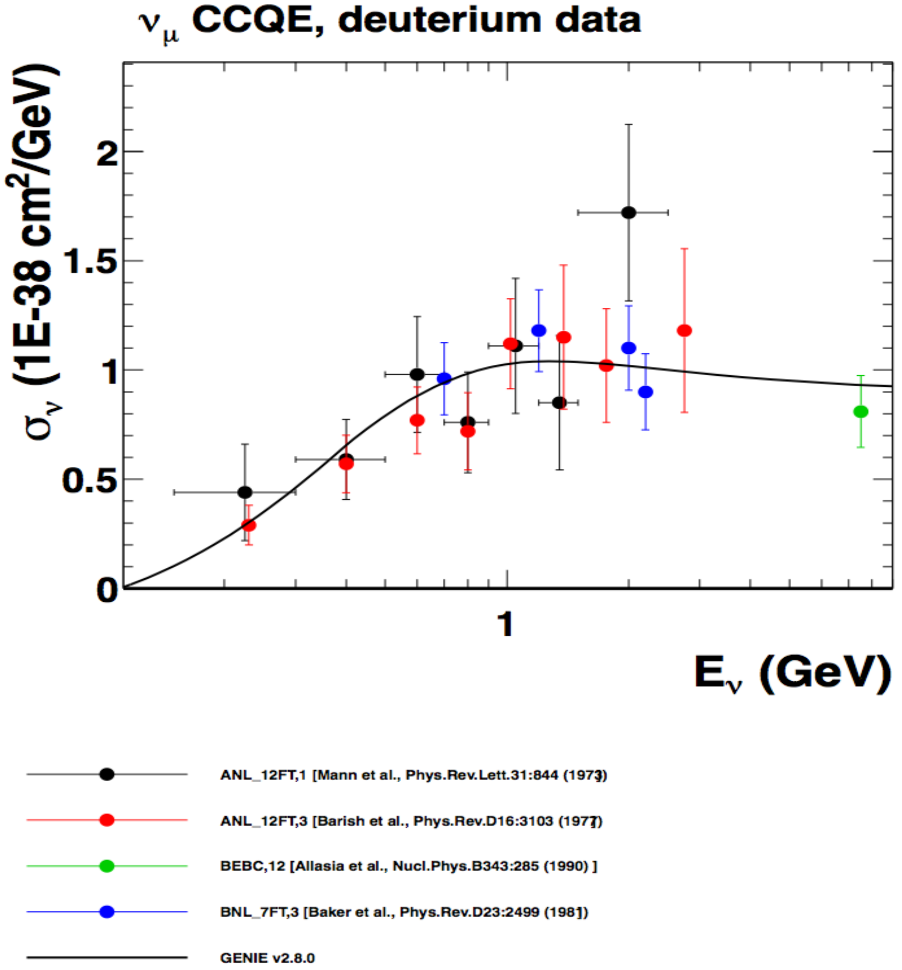
Example plots:

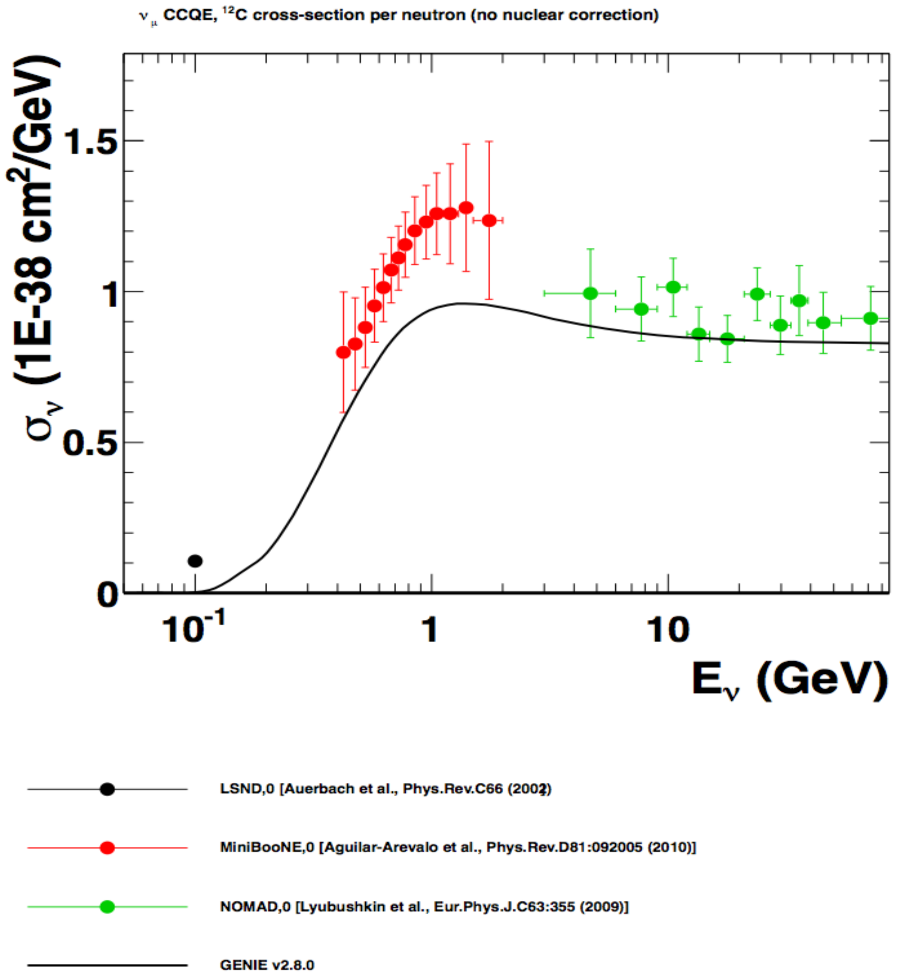




Cross-section validation

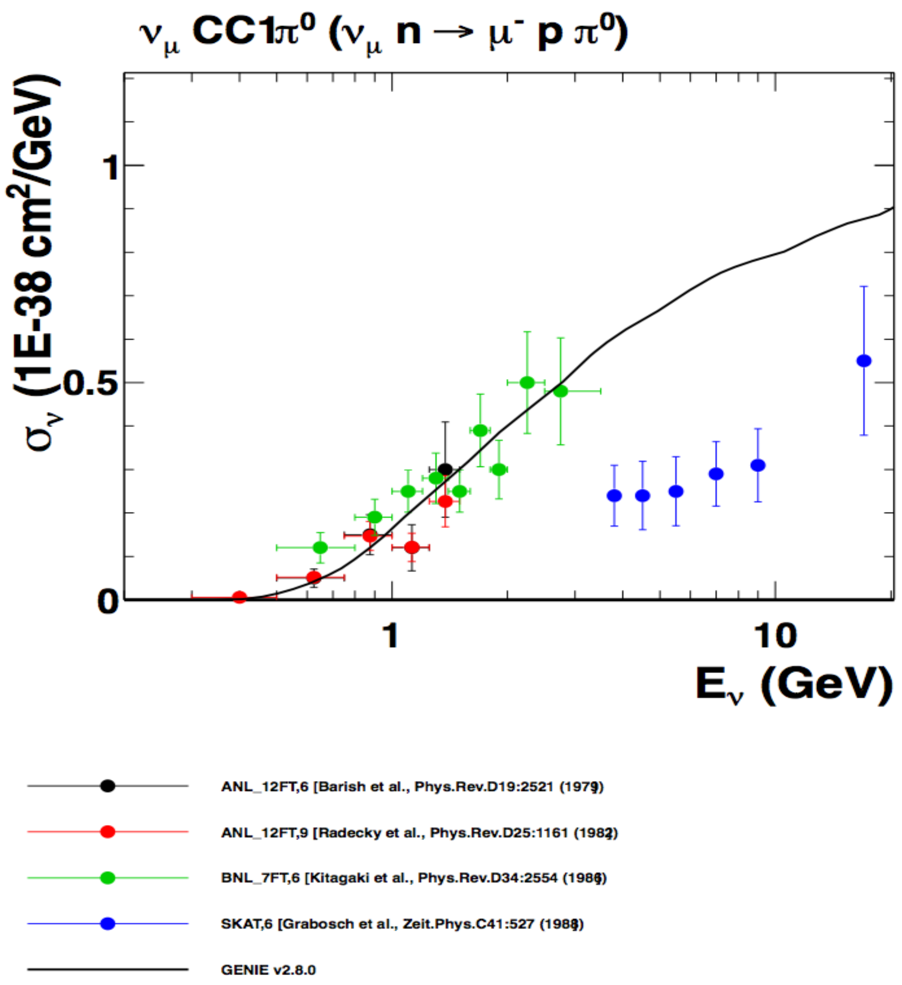
Example plots:

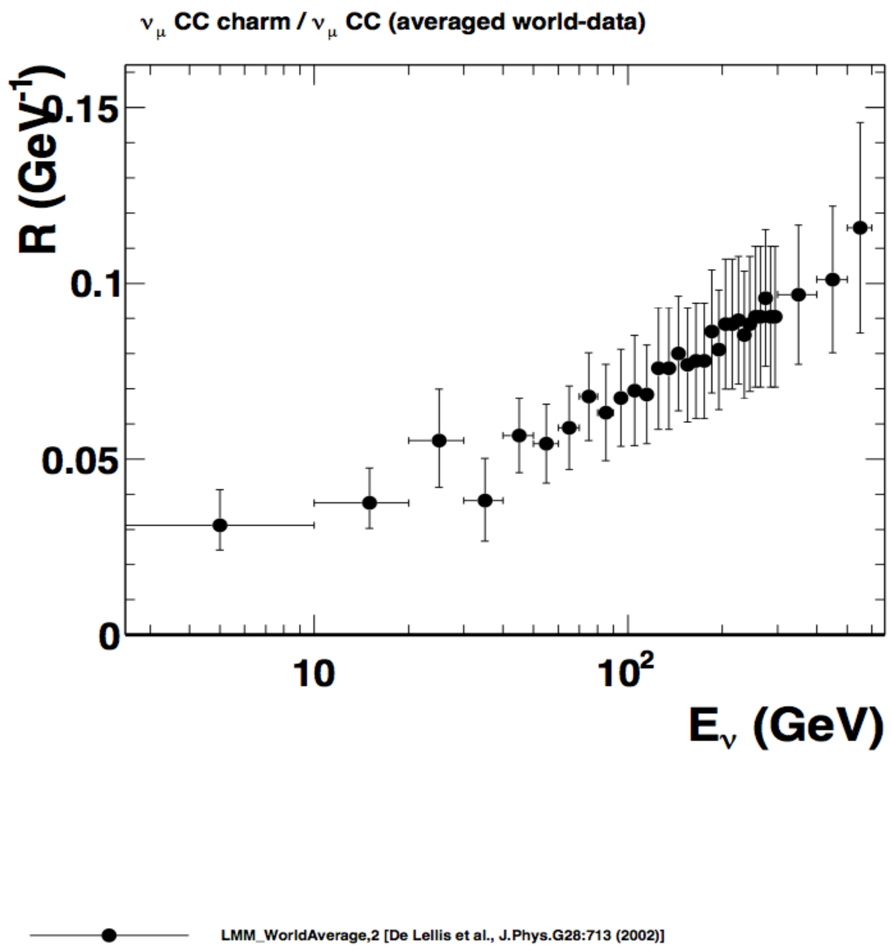




Cross-section validation

Example plots:





Cross-section validation - Status

- Fairly complete and well-written validation program
  - A good template for how I would like all validation programs to be structured
  - Modular - Every single piece of data/MC comparison independent but with some common machinery for all comparisons
  - Reads-in full event trees and calculates error envelopes
    - A small step away from being able to fit all datasets
- All other data/MC comparisons involving integrated cross-sections or cross-section ratios should be added in the existing app.
- Some GENIE predictions and error envelopes not calculated
  - Some work is needed still - not a big deal.

Hadronization validation

Next step is to compare predictions of the GENIE hadronization model with neutrino and anti-neutrino data mainly on Hydrogen/Deuterium (no intranuclear effects). Procedure:

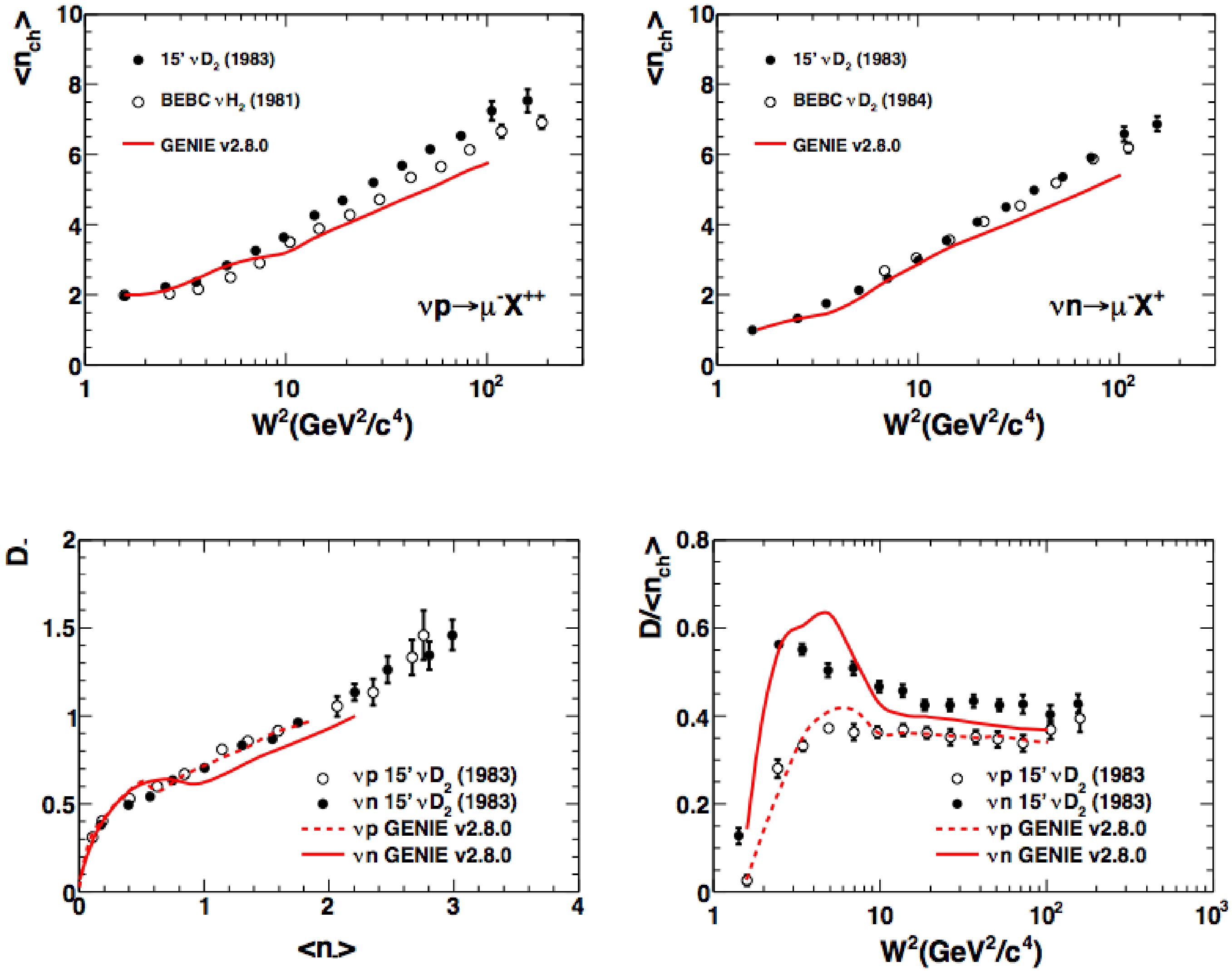
1. Submit jobs
   - Using 'perl submit_hadronization_validation_mc_jobs.pl ...'
   - Submitted jobs use gevgen event generation app.
   - Outputs in GHEP/ROOT format.
2. Monitor job status till all jobs have finished.
3. Need to check log files for errors or warnings. Stop if any.
4. Need to Check that all submitted jobs gave outputs. Stop if outputs are missing.
5. Move files to standard location.
6. Generate XML file lists to load event samples to validation program.

Hadronization validation

The XML file list for the current version and, potentially, corresponding file lists with reference samples (e.g. from previous official releases) are passed on to the gvld_hadronz_test app. The application compares GENIE and data on a number of observables:

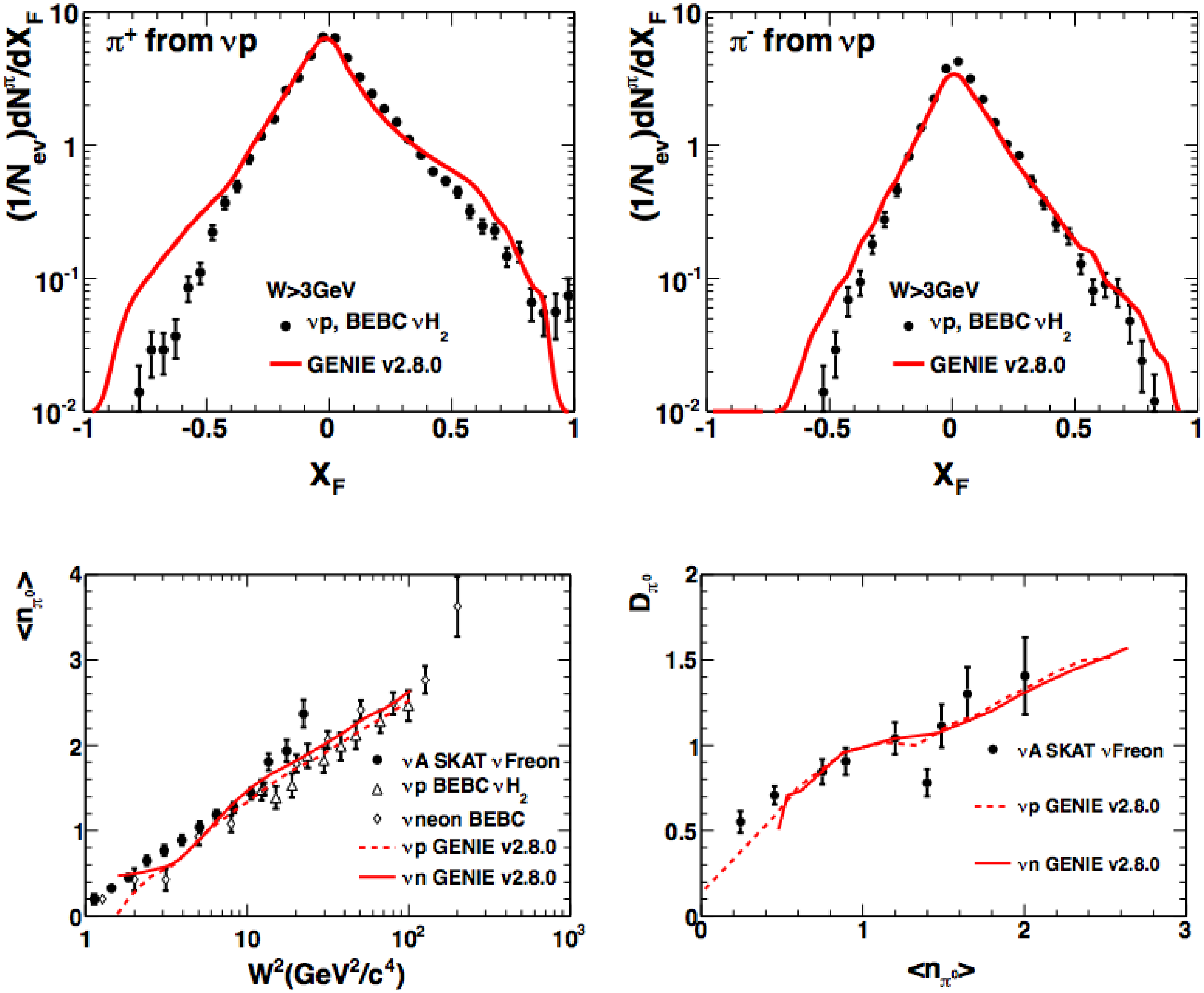
- KNO scaling
- charged hadron multiplicity
- charged hadron disperson
- π^0 multiplicity
- π^0 dispersion
- η multiplicity
- forward ($x_F > 0$) vs backward ($x_F < 0$) multiplicity
- multiplcity correlations (charged vs negative hadrons)
- p_T^2 distribution
- p_T^2 vs W^2 distribution
- p_T^2 vs x_F distribution
- z distribution

Example plots:



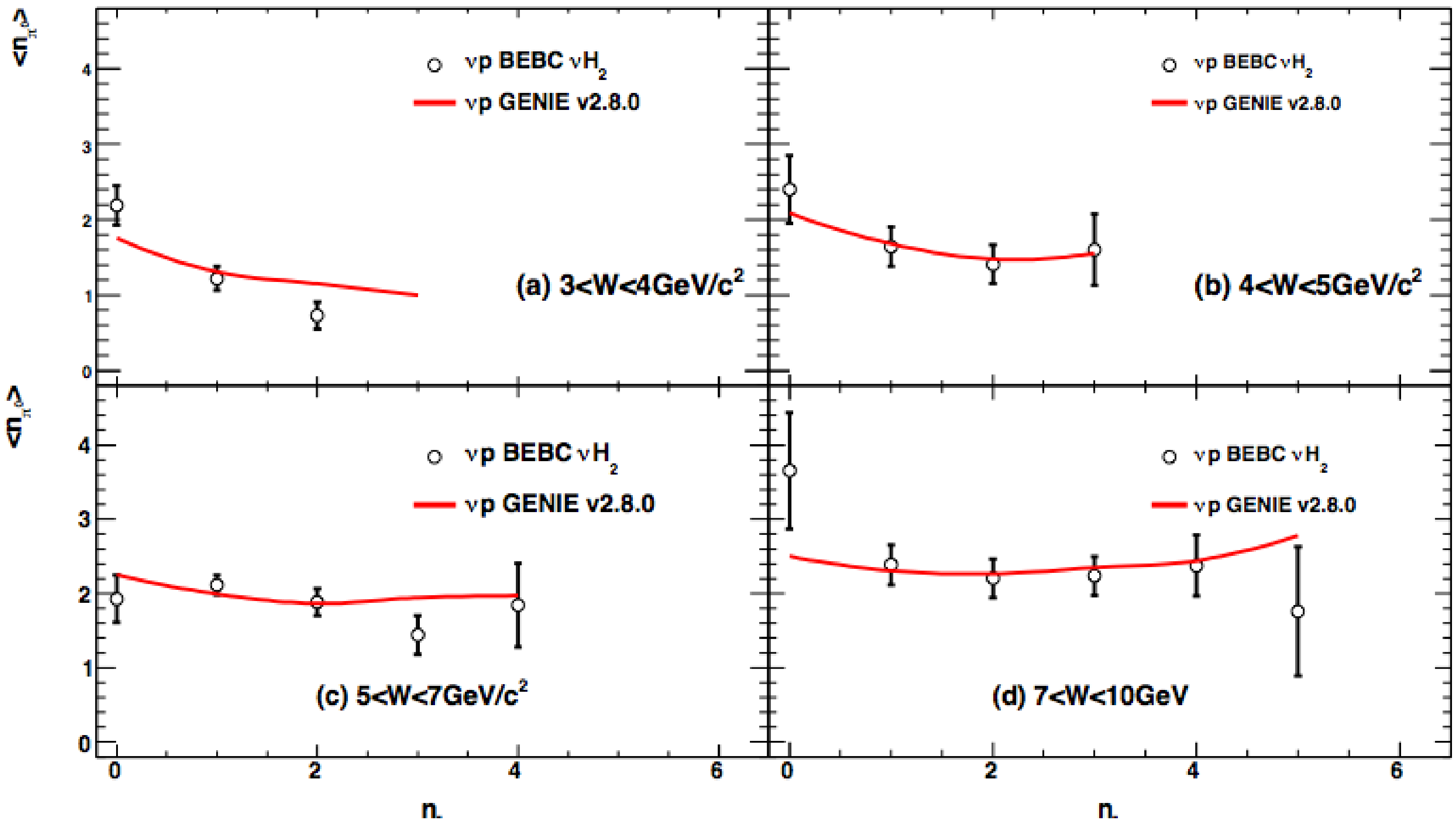
Hadronization validation

Example plots:



Hadronization validation

Example plots:



Hadronization validation - Status

- Physics-wise, a fantastic piece of code developed by T.Yang
- Initially a bunch of macros
  - Not well written - Couldn't extend, large fragments of unused code, scope for confusion and errors
- Code was put in GENIE by the original author
  - But mainly, just a new new facade to the old badly written code
  - Very fragile
- Code needs to be refactorized.
- Redundant code need to be removed.
- Source code very poorly documented - need to improve.
- Code needs to become more robust and implicit assumptions on the order of input files need to be removed.
- New data (e.g. CHORUS) could be added
- Code reads-in GHEP event trees (good!)
- Code doesn't produce error enveloped (MC statistics and systematics) - need to add
- Quite a bit of work is needed

Code currently not run as part of the automated validation

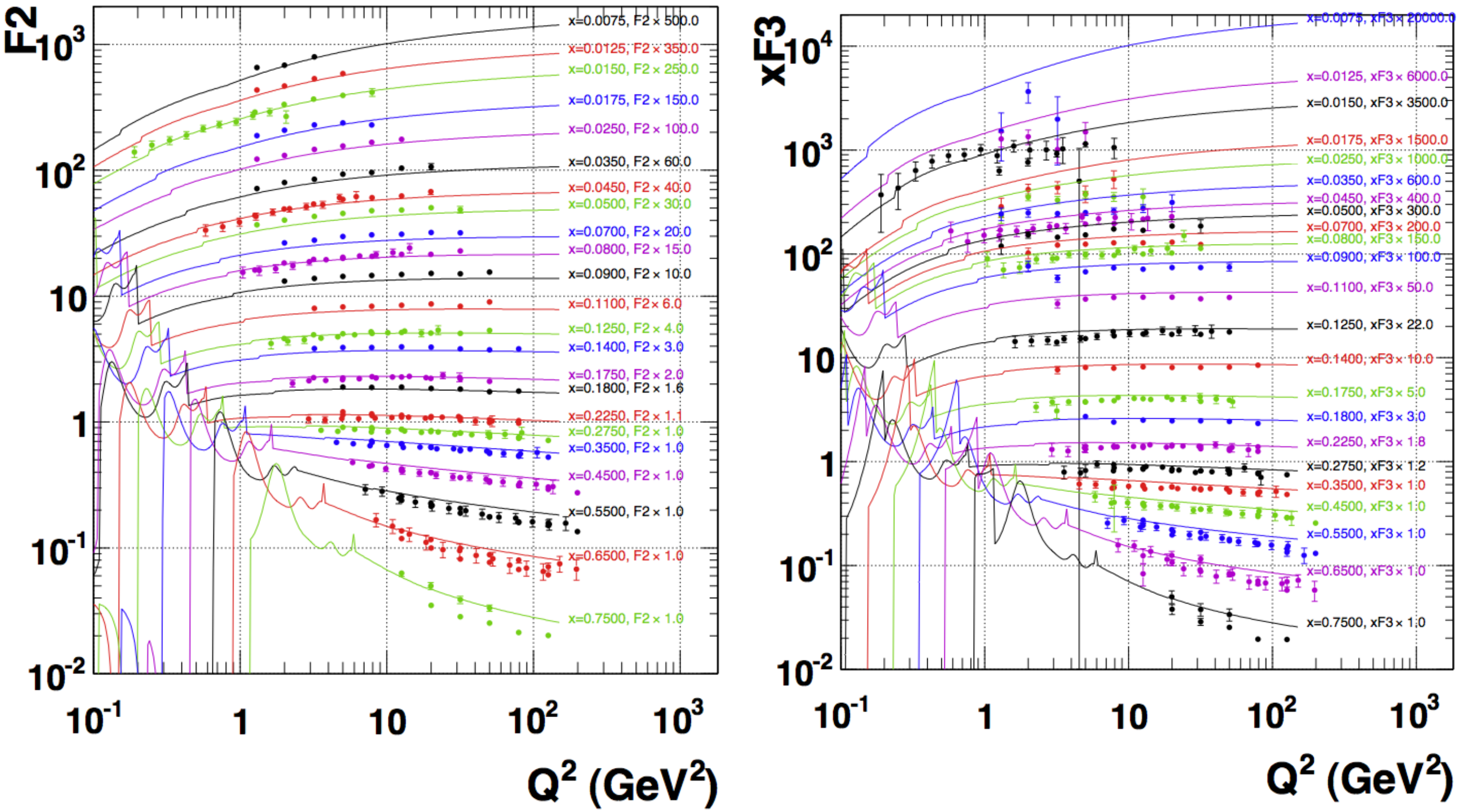
Intranuke rescattering validation - Status

- Some code exists (hadron+nucleus data/MC comparisons)
- Various versions committed by a string of Steve's undergrad students (with mixed programming abilities)
  - Certain amount of disregard for existing templates
  - Somewhat different philosophy (run everything interactively)
- Have never actually run it...
- Work may be needed to bring it up to a good standard
- Code didn't catch INTRANUKE problems that sneaked into v2.8.0
- Code didn't catch major errors present in the v2.8.0 candidate releases
  - Code checks what Steve want to see, not all that needs to be checked
  - I developed code to catch errors seen, certainly not a complete job
- Need to add error estimates in all predictions (reweighting exists)
- Validation programs do not read samples in the GHEP format (reweighting can not be run!)

Differential neutrino cross-sections

- Most recent data in this format (MiniBooNE, T2K,...)
- We all have produced GENIE/MiniBooNE comparisons, but
- no such comparison is part of the GENIE validation suite.
- Need to run these comparisons as part of the validation.
- We will eventually fit these data so, as a first step, I would like to see comparison programs using the GHEP event trees and producing error envelopes
- Quite a bit of work is needed

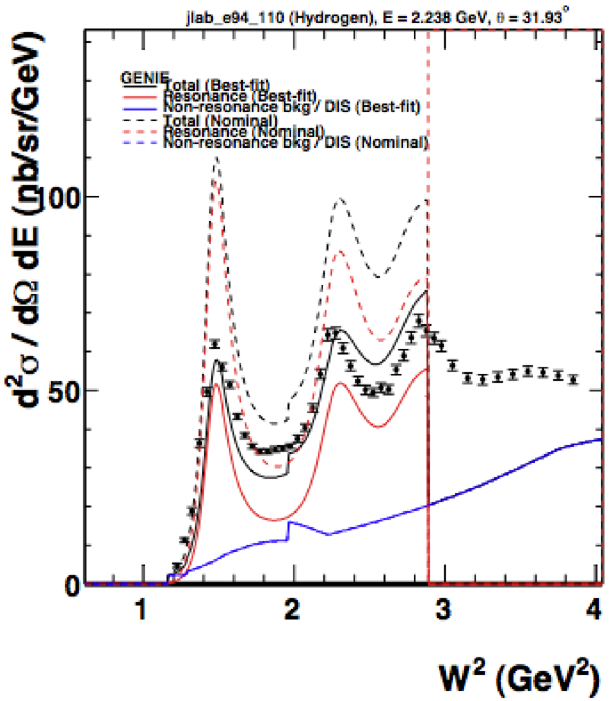
Not run as part of the validation procedure just yet, but the relevant tool already exists (gvld_sf) Example plots:

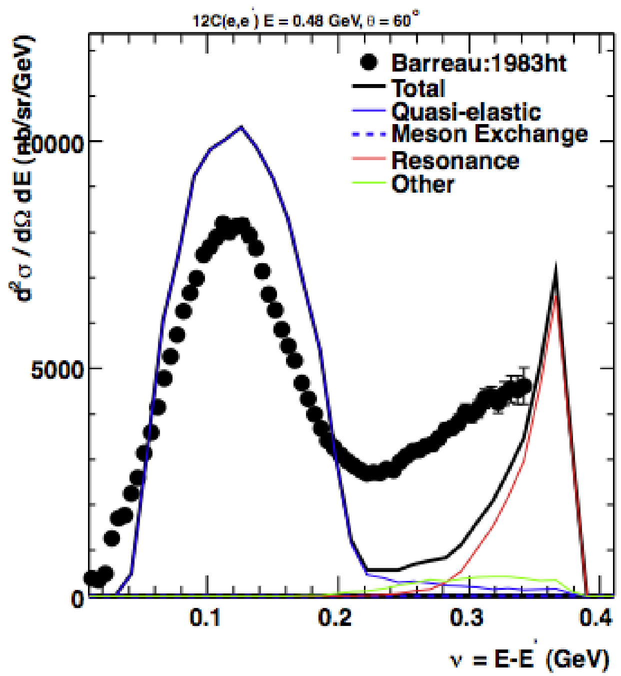


Validation running GENIE in electron mode

Not run as part of the validation procedure just yet. Tools exists for plotting $d^2\sigma/dE'd\Omega$ for Hydrogen/Deuterium (gvld_e_res_xsec, based on GENIE cross-section model) and for nuclear targets (gvld_e_qel_xsec, based on GENIE samples in electron mode).

Example plots:





Well designed.
Enormous amounts of differential cross-section data (O(100k) data points).
Programs also support fitting (see top left plot).
Some bug somewhere affects non-QE component for nuclear targets.
Some awkwardness with hardcoded Q^2 cutoffs requires code recompilation for running the electron mode.
Both issues on my plate, never gets done...,

Event reweighting tests

- A lot of tools were written to check the properties of the event reweighting methods (unitarity, comparisons of emulated and generated non-default samples etc.)
- None of these tools is incuded in GENIE
- Work is needed...

Flux drivers, geometry navigation, expt-specific apps

- Testing of flux drivers not part of the main validation effort.
- Testing of geometry navigators not part of the main validation effort.
- Testing of experiment-specific event generation apps not part of the main validation effort.
- Separate validation of T2K interfaces before all productions
  - Most GENIE revision versions were prepared during preparation for T2K MC productions.
  - GENIE T2K MC productions since 2009 with no problems.
  - ...Other experiments have not been that lucky.
- Partly, not our problem
  - It is a no brainer that experiments should fully validate their MC tools.
  - It is not our job to do the dirty job for all experiments
- This is an issue that needs to be addressed by changes in the GENIE organizational structure (expt. liaisons)
- But keen to have some basic validation
  - Although experiments **should** validate independently.
- Work is needed

Other

- Validation is run almost entirely using high-level GENIE outputs (MC event samples or cross-section calculations). Smaller units of code could be tested separately.
- Sum rule checks (some code exists).
- Hadron multiplicity ratios (data/MC) from charged-lepton scattering (some code exists).
- ?

Source code referenced (1/2)

Note: Paths following
`https://genie.hepforge.org/trac/browser/trunk/` or $GENIE

Scripts:
run_genie_validation.pl:
`src/scripts/production/batch/run_genie_validation.pl`

submit_vN_xsec_calc_jobs.pl:
submit_vA_xsec_calc_jobs.pl:
submit_standard_neutrino_mc_test_jobs.pl:
submit_neutrino_xsec_validation_mc_jobs.pl:
submit_hadronization_validation_mc_jobs.pl:
...
all located in `src/scripts/production/batch/`

Apps:
gmkspl:
`src/stdapp/gMakeSplines.cxx`
gevgen:
`src/stdapp/gEvGen.cxx`

Source code referenced (2/2)

gntpc:
    `src/stdapp/gNtpConv.cxx`
gvld_xsec_comp:
    `src/validation/MCx/validation/MCx/gVldXSecComp.cxx`
gvld_repeatability_test:
    `src/validation/EvScan/gVldRepeatabilityTest.cxx`
gvld_sample_scan:
    `src/validation/EvScan/gVldSampleScan.cxx`
gvld_sample_comp:
    `src/validation/MCx/gVldSampleComp.cxx`
gvld_sf:
    `src/validation/StructFunc/gVldStructFunc.cxx`
gvld_nu_xsec:
    `src/validation/NuXSec/gVldNuXSec.cxx`
gvld_hadronz_test:
    `src/validation/Hadronization/gVldHadronzTest.cxx`
gvld_e_qel_xsec:
    `src/validation/eA/gVldeQELXSec.cxx`
gvld_e_res_xsec:
    `src/validation/eA/gVldeRESXSec.cxx`