The GENIE Neutrino Monte Carlo Generator

PHYSICS & USER MANUAL

(version 2.10.0)

Costas Andreopoulos[2, 5], Christopher Barry[2], Steve Dytman[3], Hugh Gallagher[6], Tomasz Golan[1, 4], Robert Hatcher[1], Gabriel Perdue[1], and Julia Yarba[1]

[1] Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA

[2] University of Liverpool, Dept. of Physics, Liverpool L69 7ZE, UK

[3] University of Pittsburgh, Dept. of Physics and Astronomy, Pittsburgh PA 15260, USA

[4] University of Rochester, Dept. of Physics and Astronomy, Rochester NY 14627, USA

[5] STFC Rutherford Appleton Laboratory, Particle Physics Dept., Oxfordshire OX11 0QX, UK

[6] Tufts University, Dept. of Physics and Astronomy, Medford MA, 02155, USA

October 20, 2015

Preface

GENIE [1] is a suite of products for the experimental neutrino physics community. This suite includes i) a modern software framework for implementing neutrino event generators, a state-of-the-art comprehensive physics model and tools to support neutrino interaction simulation for realistic experimental setups (aka the "Generator"), ii) extensive archives of neutrino, charged-lepton and hadron scattering data and software to produce a comprehensive set of data/MC comparisons (aka the "Comparisons"), and iii) a generator tuning framework and fitting applications (aka the "Tuning"). These products come with different licenses. The Generator is an open-source product, whereas the Comparisons and the Tuning are distributed only under special agreement.

This book provides the definite guide for the GENIE Generator: It presents the software architecture and a detailed description of its physics model and official tunes. In addition, it provides a rich set of data/MC comparisons that characterise the physics performance of GENIE. Detailed step-by-step instructions on how to install and configure the Generator, run its applications and analyze its outputs are also included.

Key

- Class names: *Italic* (eg. *TRandom3*, *GHepRecord*, ...)

- GENIE application / library names: *Italic* (eg. *gT2Kevgen*, *gevdump*, ...)

- External packages: Normal (eg. ROOT, PYTHIA6, ...)

- Object names: 'Normal' (eg. 'flux', ...)

- Typed-in commands: `Courier+Small+Bold` (eg. `$ gevdump -f /data/file.ghep.root`)

- Fragments of typed-in commands: '`Courier+Small+Bold`' (eg. '`-n 100`')

- Environmnental variables: '`Courier+Small+Bold`' (eg. `$GENIE`)

- Filenames and paths: '*Italic*' (eg. '*/data/flux/atmospheric/bglrs/numu.root*')

- URLs: *Italic* (eg. *http://www.genie-mc.org*)

Notes:

- A leading \$, & or % in typed-in commands represent your command shell prompt. Don't type that in.

Contents

Chapter 1

Introduction

1.1 GENIE project overview

Over the last few years, throughout the field of high energy physics (HEP), we have witnessed an enormous effort committed to migrating many popular procedural Monte Carlo Generators into their C++ equivalents designed using object-oriented methodologies. Well-known examples are the GEANT [2], HERWIG [3] and PYTHIA [4] Monte Carlo Generators. This reflects a radical change in our approach to scientific computing. Along with the eternal requirement that the modeled physics be correct and extensively validated with external data, the evolving nature of computing in HEP has introduced new requirements. These requirements relate to the way large HEP software systems are developed and maintained, by wide geographically-spread collaborations over a typical time-span of $\sim$25 years during which they will undergo many (initially unforeseen) extensions and modifications to accommodate new needs. This puts a stress on software qualities such re-usability, maintainability, robustness, modularity and extensibility. Software engineering provides many well proven techniques to address these requirements and thereby improve the quality and lifetime of HEP software. In neutrino physics, the requirements for a new physics generator are more challenging for three reasons: the lack of a 'canonical' procedural generator, theoretical and phenomenological challenges in modeling few-GeV neutrino interactions, and the rapidly evolving experimental and theoretical landscape.

The long-term goal of this project is for GENIE to become a 'canonical' neutrino event generator whose validity will extend to all nuclear targets and neutrino flavors over a wide spectrum of energies ranging from $\sim$1 MeV to $\sim$1 PeV. Currently, emphasis is given to the few-GeV energy range, the challenging boundary between the non-perturbative and perturbative regimes which is relevant for the current and near future long-baseline precision neutrino experiments using accelerator-made beams. The present version provides comprehensive neutrino interaction modelling in the energy from, approximately, $\sim$100 MeV to a few hundred GeV.

GENIE[1] is a ROOT-based [5] Neutrino MC Generator. It was designed using object-oriented methodologies and developed entirely in C++ over a period of more than three years, from 2004 to 2007. Its first official physics release (v2.0.0) was made available on August 2007. GENIE has already being adopted by the majority of neutrino experiments, including those as the JPARC and NuMI neutrino beamlines, and will be an important physics tool for the exploitation of the world accelerator neutrino program.

The project is supported by a group of physicists from all major neutrino experiments operating in this energy range, establishing GENIE as a major HEP event generator collaboration. Many members

---

[1] **GENIE** stands for **G**enerates **E**vents for **N**eutrino **I**nteraction **E**xperiments

of the GENIE collaboration have extensive experience in developing and maintaining the legacy Monte Carlo Generators that GENIE strives to replace, which guarantees knowledge exchange and continuation. The default set of physics models in GENIE have adiabatically evolved from those in the NEUGEN [6] package, which was used as the event generator by numerous experiments over the past decade.

1.2   Neutrino Interaction Simulation: Challenges and Significance

Neutrinos have played an important role in particle physics since their discovery half a century ago. They have been used to elucidate the structure of the electroweak symmetry groups, to illuminate the quark nature of hadrons, and to confirm our models of astrophysical phenomena. With the discovery of neutrino oscillations using atmospheric, solar, accelerator, and reactor neutrinos, these elusive particles now take center stage as the objects of study themselves. Precision measurements of the lepton mixing matrix, the search for lepton charge-parity (CP) violation, and the determination of the neutrino masses and hierarchy will be major efforts in HEP for several decades. The cost of this next generation of experiments will be significant, typically tens to hundreds of millions of dollars. A comprehensive, thoroughly tested neutrino event generator package plays an important role in the design and execution of these experiments, since this tool is used to evaluate the feasibility of proposed projects and estimate their physics impact, make decisions about detector design and optimization, analyze the collected data samples, and evaluate systematic errors. With the advent of high-intensity neutrino beams from proton colliders, we have entered a new era of high-statistics, precision neutrino experiments which will require a new level of accuracy in our knowledge, and simulation, of neutrino interaction physics.

While object-oriented physics generators in other fields of high energy physics were evolved from well established legacy systems, in neutrino physics no such 'canonical' MC exists. Until quite recently, most neutrino experiments developed their own neutrino event generators. This was due partly to the wide variety of energies, nuclear targets, detectors, and physics topics being simulated. Without doubt these generators, the most commonly used of which have been GENEVE [7], NEUT [8], NeuGEN [6], NUANCE [9] and NUX [10], played an important role in the design and exploration of the previous and current generation of accelerator neutrino experiments. Tuning on the basis of unpublished data from each group's own experiment has not been unusual making it virtually impossible to perform a global, independent evaluation for the state-of-the-art in neutrino interaction physics simulations. Moreover, limited manpower and the fragility of the overextended software architectures meant that many of these legacy physics generators were not keeping up with the latest theoretical ideas and experimental measurements. A more recent development in the field has been the direct involvement of theory groups in the development of neutrino event generators, such as the NuWRO [11] and GiBUU [12] packages, and the inclusion of neutrino scattering in the long-established FLUKA hadron scattering package [13].

Simulating neutrino interactions in the energy range of interest to current and near-future experiments poses significant challenges. This broad energy range bridges the perturbative and non-perturbative pictures of the nucleon and a variety of scattering mechanisms are important. In many areas, including elementary cross sections, hadronization models, and nuclear physics, one is required to piece together models with different ranges of validity in order to generate events over all of the available phase space. This inevitably introduces challenges in merging and tuning models, making sure that double counting and discontinuities are avoided. In addition there are kinematic regimes which are outside the stated range of validity of all available models, in which case we are left with the challenge of developing our own models or deciding which model best extrapolates into this region. An additional fundamental problem in this energy range is a lack of data. Most simulations have been tuned to bubble chamber data taken in the 70's and 80's. Because of the limited size of the data samples (important exclusive channels might only contain a handful of events), and the limited coverage in the space of $(\nu/\overline{\nu}, E_\nu, \text{A})$, substantial uncertainties exist in numerous aspects of the simulations.

The use cases for GENIE are also informed by the experiences of the developers and users of the previous generation of procedural codes. Dealing with these substantial model uncertainties has been an important analysis challenge for many recent experiments. The impact of these uncertainties on physics analyses have been evaluated in detailed systematics studies and in some cases the models have been fit directly to experimental data to reduce systematics. These 'downstream' simulation-related studies can often be among the most challenging and time-consuming in an analysis.

To see the difficulties facing the current generation of neutrino experiments, one can look no further than the K2K and MiniBooNE experiments. Both of these experiments have measured a substantially different Q^2 distribution for their quasielastic-like events when compared with their simulations, which involve a standard Fermi Gas model nuclear model [14, 15]. The disagreement between nominal Monte Carlo and data is quite large - in the lowest Q^2 bin of MiniBooNE the deficit in the data is around 30% [15]. It seems likely that the discrepancies seen by both experiments have a common origin. However the two experiments have been able to obtain internal consistency with very different model changes - the K2K experiment does this by eliminating the charged current (CC) coherent contribution in the Monte Carlo [16] and the MiniBooNE experiment does this by modifying certain parameters in their Fermi Gas model [15]. Another example of the rapidly evolving nature of this field is the recently reported excess of low energy electron-like events by the MiniBooNE collaboration [17]. These discrepancies have generated significant new theoretical work on these topics over the past several years [18, 19, 20, **?**, 21, 22, 23, 24]. The situation is bound to become even more interesting, and complicated, in the coming decade, as new high-statistics experiments begin taking data in this energy range. Designing a software system that can be responsive to this rapidly evolving experimental and theoretical landscape is a major challenge.

One of the aims of this manual is to describe the ways in which the GENIE neutrino event generator addresses these challenges. These solutions rely heavily on the power of modern software engineering, particularly the extensibility, modularity, and flexibility of object oriented design, as well as the combined expertise and experience of the collaboration with previous procedural codes.

Chapter 2

Neutrino Interaction Physics Modeling

2.1 Introduction

The set of physics models used in GENIE incorporates the dominant scattering mechanisms from several MeV to several hundred GeV and are appropriate for any neutrino flavor and target type. Over this energy range, many different physical processes are important. These physics models can be broadly categorized into nuclear physics models, cross section models, and hadronization models.

Substantial uncertainties exist in modeling neutrino-nucleus interactions, particularly in the few-GeV regime which bridges the transition region between perturbative and non-perturbative descriptions of the scattering process. For the purposes of developing an event generator this theoretical difficulty is compounded by the empirical limitation that previous experiments often did not publish results in these difficult kinematic regions since a theoretical interpretation was unavailable.

In physics model development for GENIE we have been forced to pay particular attention to this 'transition region', as for few-GeV experiments it dominates the event rate. In particular the boundaries between regions of validity of different models need to be treated with care in order to avoid theoretical inconsistencies, discontinuities in generated distributions, and double-counting. In this brief section we will describe the models available in GENIE and the ways in which we combine models to cover regions of phase space where clear theoretical or empirical guidance is lacking.

2.2 Nuclear Physics Model

The importance of the nuclear model depends strongly on the kinematics of the reaction. Nuclear physics plays a large role in every aspect of neutrino scattering simulations at few-GeV energies and introduces coupling between several aspects of the simulation. The relativistic Fermi gas (RFG) nuclear model is used for all processes. GENIE uses the version of Bodek and Ritchie which has been modified to incorporate short range nucleon-nucleon correlations [25]. This is simple, yet applicable across a broad range of target atoms and neutrino energies. The best tests of the RFG model come from electron scattering experiments [26]. At high energies, the nuclear model requires broad features due to shadowing and similar effects. At the lower end of the GENIE energy range, the impulse approximation works very well and the RFG is often successful. The nuclear medium gives the struck nucleon a momentum and average binding energy which have been determined in electron scattering experiments. Mass densities are taken from review articles [27]. For $A < 20$, the modified Gaussian density parameterization is used. For heavier nuclei, the 2-parameter Woods-Saxon density function is used. Thus, the model can be used for *all* nuclei. Presently, fit parameters for selected nuclei are used with interpolations for other nuclei. All isotopes of a particular

nucleus are assumed to have the same density.

It is well known that scattering kinematics for nucleons in a nuclear environment are different from those obtained in scattering from free nucleons. For quasi-elastic and elastic scattering, Pauli blocking is applied as described in Sec. 2.3. For nuclear targets a nuclear modification factor is included to account for observed differences between nuclear and free nucleon structure functions which include shadowing, anti-shadowing, and the EMC effect [28].

Nuclear reinteractions of produced hadrons is simulated using a cascade Monte Carlo which will be described in more detail in a following section. The struck nucleus is undoubtedly left in a highly excited state and will typically de-excite by emitting nuclear fission fragments, nucleons, and photons. At present de-excitation photon emission is simulated only for oxygen [29, 30] due to the significance of these 3-10 MeV photons in energy reconstruction at water Cherenkov detectors. Future versions of the generator will handle de-excitation photon emission from additional nuclear targets.

2.3   Cross section model

The cross section model provides the calculation of the differential and total cross sections. During event generation the total cross section is used together with the flux to determine the energies of interacting neutrinos. The cross sections for specific processes are then used to determine which interaction type occurs, and the differential distributions for that interaction model are used to determine the event kinematics. While the differential distributions must be calculated event-by-event, the total cross sections can be pre-calculated and stored for use by many jobs sharing the same physics models. Over this energy range neutrinos can scatter off a variety of different 'targets' including the nucleus (via coherent scattering), individual nucleons, quarks within the nucleons, and atomic electrons.

Quasi-Elastic Scattering: Quasi-elastic scattering (e.g. $\nu_\mu + n \to \mu^- + p$) is modeled using an implementation of the Llewellyn-Smith model [31]. In this model the hadronic weak current is expressed in terms of the most general Lorentz-invariant form factors. Two are set to zero as they violate G-parity. Two vector form factors can be related via CVC to electromagnetic form factors which are measured over a broad range of kinematics in electron elastic scattering experiments. Several different parametrizations of these electromagnetic form factors including Sachs [32], BBA2003 [33] and BBBA2005 [34] models are available with BBBA2005 being the default. Two form factors - the pseudo-scalar and axial vector, remain. The pseudo-scalar form factor is assumed to have the form suggested by the partially conserved axial current (PCAC) hypothesis [31], which leaves the axial form factor $F_A(Q^2)$ as the sole remaining unknown quantity. $F_A(0)$ is well known from measurements of neutron beta decay and the Q^2 dependence of this form factor can only be determined in neutrino experiments and has been the focus of a large amount of experimental work over several decades. In GENIE a dipole form is assumed, with the axial vector mass m_A remaining as the sole free parameter with a default value of 0.99 GeV/c^2.

For nuclear targets, the struck a suppression factor is included from an analytic calculation of the rejection factor in the Fermi Gas model, based on the simple requirement that the momentum of the outgoing nucleon exceed the fermi momentum k_F for the nucleus in question. Typical values of k_F are 0.221 GeV/c for nucleons in ^{12}C, 0.251 GeV/c for protons in ^{56}Fe, and 0.256 GeV/c for neutrons in ^{56}Fe.

Elastic Neutral Current Scattering: Elastic neutral current processes are computed according to the model described by Ahrens et al. [35], where the axial form factor is given by:

$$G_A(Q^2) = \frac{1}{2} \frac{G_A(0)}{(1 + Q^2/M_A^2)^2}(1 + \eta). \tag{2.1}$$

The adjustable parameter η includes possible isoscalar contributions to the axial current, and the GENIE default value is $\eta = 0.12$. For nuclear targets the same reduction factor described above is used.

Baryon Resonance Production: The production of baryon resonances in neutral and charged current channels is included with the Rein-Sehgal model [36]. This model employs the Feynman-Kislinger-Ravndal [37] model of baryon resonances, which gives wavefunctions for the resonances as excited states of a 3-quark system in a relativistic harmonic oscillator potential with spin-flavor symmetry. In the Rein-Sehgal paper the helicity amplitudes for the FKR model are computed and used to construct the cross sections for neutrino-production of the baryon resonances. From the 18 resonances of the original paper we include the 16 that are listed as unambiguous at the latest PDG baryon tables and all resonance parameters have been updated. In our implementation of the Rein-Sehgal model interference between neighboring resonances has been ignored. Lepton mass terms are not included in the calculation of the differential cross section, but the effect of the lepton mass on the phase space boundaries is taken into account. For tau neutrino charged current interactions an overall correction factor to the total cross section is applied to account for neglected elements (pseudoscalar form factors and lepton mass terms) in the original model. The default value for the resonance axial vector mass m_A is 1.12 GeV/c^2, as determined in the global fits carried out in Reference [38].

Coherent Neutrino-Nucleus Scattering: Coherent scattering results in the production of forward going pions in both charged current ($\nu_\mu + A \rightarrow \mu^- + \pi^+ + A$) and neutral current ($\nu_\mu + A \rightarrow \nu_\mu + \pi^0 + A$) channels. Coherent neutrino-nucleus interactions are modeled according to the Rein-Sehgal model [39]. Since the coherence condition requires a small momentum transfer to the target nucleus, it is a low-Q^2 process which is related via PCAC to the pion field. The Rein-Sehgal model begins from the PCAC form at Q^2=0, assumes a dipole dependence for non-zero Q^2, with $m_A = 1.00$ GeV/c^2, and then calculates the relevant pion-nucleus cross section from measured data on total and inelastic pion scattering from protons and deuterium [40]. The GENIE implementation is using the modified PCAC formula described in a recent revision of the Rein-Sehgal model [41] that includes lepton mass terms.

Non-Resonance Inelastic Scattering: Deep (and not-so-deep) inelastic scattering (DIS) is calculated in an effective leading order model using the modifications suggested by Bodek and Yang [28] to describe scattering at low Q^2. In this model higher twist and target mass corrections are accounted for through the use of a new scaling variable and modifications to the low Q^2 parton distributions. The cross sections are computed at a fully partonic level (the $\nu q \rightarrow l q'$ cross sections are computed for all relevant sea and valence quarks). The longitudinal structure function is taken into account using the Whitlow R ($R = F_L/2xF_1$) parameterization [42]. The default parameter values are those given in [28], which are determined based on the GRV98 LO parton distributions [43]. An overall scale factor of 1.032 is applied to the predictions of the Bodek-Yang model to achieve agreement with the measured value of the neutrino cross section at high energy (100 GeV). This factor is necessary since the Bodek-Yang model treats axial and vector form modifications identically and would therefore not be expected to reproduce neutrino data perfectly. This overall DIS scale factor needs to be recalculated when elements of the cross section model are changed.

The same model can be extended to low energies; it is the model used for the nonresonant processes that compete with resonances in the few-GeV region.

Quasi-Elastic Charm Production: QEL charm production is modeled according to the Kovalenko local duality inspired model [44] tuned by the GENIE authors to recent NOMAD data [45].

Deep-Inelastic Charm Production: DIS charm production is modeled according to the Aivazis, Olness and Tung model [46]. Charm-production fractions for neutrino interactions are taken from [47], and utilize both Peterson [48] and Collins-Spiller [49] fragmentation functions, with Peterson fragmentation functions being the default. The charm mass is adjustable and is set to 1.43 GeV/c^2 by default.

Inclusive Inverse Muon Decay: Inclusive Inverse Muon Decay cross section is computed using an implementation of the Bardin and Dokuchaeva model [50] taking into account all 1-loop radiative corrections.

Neutrino-Electron Elastic Scattering: The cross sections for all $\nu e-$ scattering channels other than Inverse Muon Decay is computed according to [51]. Inverse Tau decay is neglected.

Modeling the transition region

As discussed, for example, by Kuzmin, Lyubushkin and Naumov [52] one typically considers the total νN CC scattering cross section as

$$\sigma^{tot} = \sigma^{QEL} \oplus \sigma^{1\pi} \oplus \sigma^{2\pi} \oplus ... \oplus \sigma^{1K} \oplus ... \oplus \sigma^{DIS}$$

In the absence of a model for exclusive inelastic multi-particle neutrinoproduction, the above is usually being approximated as

$$\sigma^{tot} = \sigma^{QEL} \oplus \sigma^{RES} \oplus \sigma^{DIS}$$

assuming that all exclusive low multiplicity inelastic reactions proceed primarily through resonance neutrinoproduction. For the sake of simplicity, small contributions to the total cross section in the few GeV energy range, such as coherent and elastic νe^- scattering, were omitted from the expression above. In this picture, one should be careful in avoiding double counting the low multiplicity inelastic reaction cross sections.

In GENIE release the total cross sections is constructed along the same lines, adopting the procedure developed in NeuGEN [6] to avoid double counting. The total inelastic differential cross section is computed as

$$\frac{d^2\sigma^{inel}}{dQ^2 dW} = \frac{d^2\sigma^{RES}}{dQ^2 dW} + \frac{d^2\sigma^{DIS}}{dQ^2 dW}$$

The term $d^2\sigma^{RES}/dQ^2 dW$ represents the contribution from all low multiplicity inelastic channels proceeding via resonance production. This term is computed as

$$\frac{d^2\sigma^{RES}}{dQ^2 dW} = \sum_k \left(\frac{d^2\sigma^{R/S}}{dQ^2 dW}\right)_k \cdot \Theta(W cut - W)$$

where the index k runs over all baryon resonances taken into account, W_{cut} is a configurable parameter and $(d^2\sigma_{\nu N}^{RS}/dQ^2 dW)_k$ is the Rein-Seghal model prediction for the kth resonance cross section.

The DIS term of the inelastic differential cross section is expressed in terms of the differential cross section predicted by the Bodek-Yang model appropriately modulated in the "resonance-dominance" region $W < W_{cut}$ so that the RES/DIS mixture in this region agrees with inclusive cross section data [53, 54, 55, 56, 57, 58, 59, 60, 61, 62] and exclusive 1-pion [63, 64, 65, 66, 67, 68, 69, 70, 71, 60, 72] and 2-pion [73, 67] cross section data:

$$\begin{aligned}
\frac{d^2\sigma^{DIS}}{dQ^2 dW} &= \frac{d^2\sigma^{DIS,BY}}{dQ^2 dW} \cdot \Theta(W - W cut) + \\
&+ \frac{d^2\sigma^{DIS,BY}}{dQ^2 dW} \cdot \Theta(W cut - W) \cdot \sum_m f_m
\end{aligned}$$

In the above expression, m refers to the multiplicity of the hadronic system and, therefore, the factor f_m relates the total calculated DIS cross section to the DIS contribution to this particular multiplicity channel. These factors are computed as $f_m = R_m \cdot P_m^{had}$ where R_m is a tunable parameter and P_m^{had} is the probability, taken from the hadronization model, that the DIS final state hadronic system multiplicity would be equal to m. The approach described above couples the GENIE cross section and hadronic multiparticle production model [74].

Cross section model tuning

As mentioned previously, the quasi-elastic, resonance production, and DIS models employ form factors, axial vector masses, and other parameters which have been determined by others in their global fits [34, 38]. In order to check the overall consistency of our model, and to verify that we have correctly implemented the DIS model, predictions are compared to electron scattering inclusive data [75, 76] and neutrino structure function data [77]. The current default values for transition region parameters are W_{cut}=1.7 GeV/c^2, $R_2(\nu p) = R_2(\overline{\nu}n)$=0.1, $R_2(\nu n) = R_2(\overline{\nu}p)$=0.3, and R_m=1.0 for all $m > 2$ reactions. These are determined from fits to inclusive and exclusive (one and two-pion) production neutrino interaction channels. For these comparisons we rely heavily on online compilations of neutrino data [78] and related fitting tools [79] that allow one to include some correlated systematic errors (such as arising from flux uncertainties). The GENIE default cross section for $\nu\mu$ charged current scattering from an isoscalar target, together with the estimated uncertainty on the total cross section, as evaluated in [80] are shown in Fig. 2.1.

2.4 Neutrino-induced Hadron Production

Introduction

In neutrino interaction simulations the hadronization model (or fragmentation model) determines the final state particles and 4-momenta given the nature of a neutrino-nucleon interaction (CC/NC, $\nu/\bar{\nu}$, target neutron/proton) and the event kinematics (W^2, Q^2, x, y). The modeling of neutrino-produced hadronic showers is important for a number of analyses in the current and coming generation of neutrino oscillation experiments:

Calorimetry: Neutrino oscillation experiments like MINOS which use calorimetry to reconstruct the shower energy, and hence the neutrino energy, are sensitive to the modelling of hadronic showers. These detectors are typically calibrated using single particle test beams, which introduces a model dependence in determining the conversion between detector activity and the energy of neutrino-produced hadronic systems [80].

NC/CC Identification: Analyses which classify events as charged current (CC) or neutral current (NC) based on topological features such as track length in the few-GeV region rely on accurate simulation of hadronic particle distributions to determine NC contamination in CC samples.

Topological Classification: Analyses which rely on topological classifications, for instance selecting quasi-elastic-like events based on track or ring counting depend on the simulation of hadronic systems to determine feeddown of multi-particle states into selected samples. Because of the wide-band nature of most current neutrino beams, this feeddown is non-neglible even for experiments operating in beams with mean energy as low as 1 GeV [15, 81].

ν_e *Appearance Backgrounds:* A new generation of $\nu_\mu \to \nu_e$ appearance experiments are being developed around the world, which hope to find evidence of charge-parity (CP) violation in the lepton sector. In these experiments background is dominated by neutral pions generated in NC interaction. The evaluation of NC backgrounds in these analysis can be quite sensitive to the details of the NC shower simulation and specifically the π^0 shower content and transverse momentum distributions of hadrons [82].

In order to improve Monte Carlo simulations for the MINOS experiment, a new hadronization model, referred to here as the 'AGKY model', was developed. We use the PYTHIA/JETSET [84] model to simulate the hadronic showers at high hadronic invariant masses. We also developed a phenomenological description of the low invariant mass hadronization since the applicability of the PYTHIA/JETSET model, for neutrino-induced showers, is known to deteriorate as one approaches the pion production threshold. We present here a description of the AGKY hadronization model and the tuning and validation of this model using bubble chamber experimental data.

The AGKY Model

The AGKY model, which is now the default hadronization model in the neutrino Monte Carlo generators NEUGEN [6] and GENIE-2.0.0 [85], includes a phenomenological description of the low invariant mass region based on Koba-Nielsen-Olesen (KNO) scaling [86], while at higher masses it gradually switches over to the PYTHIA/JETSET model. The transition from the KNO-based model to the PYTHIA/JETSET model takes place gradually, at an intermediate invariant mass region, ensuring the continuity of all simulated observables as a function of the invariant mass. This is accomplished by using a transition window $[W_{min}^{tr}, W_{max}^{tr}]$ over which we linearly increase the fraction of neutrino events for which the hadronization is performed by the PYTHIA/JETSET model from 0% at W_{min}^{tr} to 100% at W_{max}^{tr}. The default values used in the AGKY model are:

$$W_{min}^{tr} = 2.3 \text{ GeV/c}^2, W_{max}^{tr} = 3.0 \text{ GeV/c}^2. \tag{2.2}$$

The kinematic region probed by any particular experiment depends on the neutrino flux, and for the 1-10 GeV range of importance to oscillation experiments, the KNO-based phenomenological description plays a particularly crucial role. The higher invariant mass region where PYTHIA/JETSET is used is not accessed until a neutrino energy of approximately 3 GeV is reached, at which point 44.6% of charged current interactions are non-resonant inelastic and are hadronized using the KNO-based part of the model. For 1 GeV neutrinos this component is 8.3%, indicating that this model plays a significant role even at relatively low neutrino energies. At 9 GeV, the contributions from the KNO-based and PYTHIA/JETSET components of the model are approximately equal, with each handling around 40% of generated CC interactions. The main thrust of this work was to improve the modeling of hadronic showers in this low invariant mass / energy regime which is of importance to oscillation experiments.

The description of AGKY's KNO model, used at low invariant masses, can be split into two independent parts:

- Generation of the hadron shower particle content

- Generation of hadron 4-momenta

These two will be described in detail in the following sections.

The neutrino interactions are often described by the following kinematic variables:

$$
\begin{aligned}
Q^2 &= 2E_\nu(E_\mu - p_\mu^L) - m^2 \\
\nu &= E_\nu - E_\mu \\
W^2 &= M^2 + 2M\nu - Q^2 \\
x &= Q^2/2M\nu \\
y &= \nu/E_\nu
\end{aligned}
\tag{2.3}
$$

where Q^2 is the invariant 4-momentum transfer squared, ν is the neutrino energy transfer, W is the effective mass of all secondary hadrons (invariant hadronic mass), x is the Bjorken scaling variable, y is the relative energy transfer, E_ν is the incident neutrino energy, E_μ and p_μ^L are the energy and longitudinal momentum of the muon, M is the nucleon mass and m is the muon mass.

For each hadron in the hadronic system, we define the variables $z = E_h/\nu$, $x_F = 2p_L^*/W$ and p_T where E_h is the energy in the laboratory frame, p_L^* is the longitudinal momentum in the hadronic c.m.s., and p_T is the transverse momentum.

Low-W model: Particle content   At low invariant masses the AGKY model generates hadronic systems that typically consist of exactly one baryon (p or n) and any number of π and K mesons that are kinematically possible and consistent with charge conservation.

Table 2.1: Default average hadron multiplicity and dispersion parameters used in the AGKY model.

	νp	νn	$\bar{\nu} p$	$\bar{\nu} n$
a_{ch}	0.40 [83]	-0.20 [83]	0.02 [88]	0.80 [88]
b_{ch}	1.42 [83]	1.42 [83]	1.28 [88]	0.95 [88]
c_{ch}	7.93 [83]	5.22 [83]	5.22	7.93
$a_{hyperon}$	0.022	0.022	0.022	0.022
$b_{hyperon}$	0.042	0.042	0.042	0.042

For a fixed hadronic invariant mass and initial state (neutrino and struck nucleon), the method for generating the hadron shower particles generally proceeds in four steps:

Determine $\langle n_{ch} \rangle$: Compute the average charged hadron multiplicity using the empirical expression:

$$\langle n_{ch} \rangle = a_{ch} + b_{ch} \ \ln W^2 \qquad (2.4)$$

The coefficients a_{ch}, b_{ch}, which depend on the initial state, have been determined by bubble chamber experiments.

Determine $\langle n \rangle$: Compute the average hadron multiplicity as $\langle n_{tot} \rangle = 1.5 \langle n_{ch} \rangle$ [87].

Deterimine n: Generate the actual hadron multiplicity taking into account that the multiplicity dispersion is described by the KNO scaling law [86]:

$$\langle n \rangle \times P(n) = f(n/\langle n \rangle) \qquad (2.5)$$

where $P(n)$ is the probability of generating n hadrons and f is the universal scaling function which can be parametrized by the Levy function [1] $(z = n/\langle n \rangle)$ with an input parameter c that depends on the initial state. Fig.2.2 shows the KNO scaling distributions for νp (left) and νn (right) CC interactions. We fit the data points to the Levy function and the best fit parameters are $c_{ch} = 7.93 \pm 0.34$ for the νp interactions and $c_{ch} = 5.22 \pm 0.15$ for the νn interactions.

Select particle types: Select hadrons up to the generated hadron multiplicity taking into account charge conservation and kinematic constraints. The hadronic system contains any number of mesons and exactly one baryon which is generated based on simple quark model arguments. Protons and neutrons are produced in the ratio 2:1 for νp interactions, 1:1 for νn and $\bar{\nu} p$, and 1:2 for $\bar{\nu} n$ interactions. Charged mesons are then created in order to balance charge, and the remaining mesons are generated in neutral pairs. The probablilities for each are 31.33% (π^0, π^0), 62.66% (π^+, π^-), and 6% strange meson pairs. The probability of producing a strange baryon via associated production is determined from a fit to Λ production data:

$$P_{hyperon} = a_{hyperon} + b_{hyperon} \ \ln W^2 \qquad (2.6)$$

TABLE 2.1 shows the default average hadron multiplicity and dispersion parameters used in the AGKY model.

Low-W model: Hadron system decay Once an acceptable particle content has been generated, the available invariant mass needs to be partitioned amongst the generated hadrons. The most pronounced kinematic features in the low-W region result from the fact that the produced baryon is much heavier than the mesons and exhibits a strong directional anticorrelation with the current direction.

Our strategy is to first attempt to reproduce the experimentally measured final state nucleon momentum distributions. We then perform a phase space decay on the remnant system employing, in addition, a p_T-based rejection scheme designed to reproduce the expected meson transverse momentum distribution. The hadronization model performs its calculation in the hadronic c.m.s., where the z-axis is in

---

[1]The Levy function: $Levy(z; c) = 2e^{-c}c^{cz+1}/\Gamma(cz + 1)$

the direction of the momentum transfer. Once the hadronization is completed, the hadronic system will be boosted and rotated to the LAB frame. The boost and rotation maintains the p_T generated in the hadronic c.m.s.

In more detail, the algorithm for decaying a system of N hadrons is the following:

Generate baryon: Generate the baryon 4-momentum $P_N^* = (E_N^*, \mathbf{p}_N^*)$ using the nucleon p_T^2 and x_F PDFs which are parametrized based on experimental data [89, 90]. The x_F distribution used is shown in Fig.2.3. We do not take into account the correlation between p_T and x_F in our selection.

Remnant System: Once an accepted P_N^* has been generated, calculate the 4-momentum of the remaining N-1 hadrons, (the "remnant" hadronic system) as $P_R^* = P_X^* - P_N^*$ where $P_X^* = (W, 0)$ is the initial hadron shower 4-momentum in the hadronic c.m.s.

Decay Remnant System: Generate an unweighted phase space decay of the remnant hadronic system. The decay takes place at the remnant system c.m.s. after which the particles are boosted back to the hadronic c.m.s. The phase space decay employs a rejection method suggested in [91], with a rejection factor e^{-A*p_T} for each meson. This causes the transverse momentum distribution of the generated mesons to fall exponentially with increasing p_T^2. Here p_T is the momentum component perpendicular to the current direction.

Two-body hadronic systems are treated as a special case. Their decay is performed isotropically in the hadronic c.m.s. and no p_T-based suppression factor is applied.

High-W model: PYTHIA The high invariant mass hadronization is performed by the PYTHIA model [84]. The PYTHIA program is a standard tool for the generation of high-energy collisions, comprising a coherent set of physics models for the evolution from a few-body hard process to a complex multihadronic final state. It contains a library of hard processes and models for initial- and final-state parton showers, multiple parton-parton interactions, beam remnants, string fragmentation and particle decays. The hadronization model in PYTHIA is based on the Lund string fragmentation framework [92]. In the AGKY model, all but four of the PYTHIA configuration parameters are set to be the default values. Those four parameters take the non-default values tuned by NUX [10], a high energy neutrino MC generator used by the NOMAD experiment:

- $P_{s\bar{s}}$ controlling the $s\bar{s}$ production suppression:
  (PARJ(2))=0.21.

- $P_{\langle p_T^2 \rangle}$ determining the average hadron $\langle p_T^2 \rangle$:
  (PARJ(21))=0.44.

- P_{ngt} parameterizing the non-gaussian p_T tails:
  (PARJ(23))=0.01.

- P_{Ec} an energy cutoff for the fragmentation process:
  (PARJ(33))=0.20.

Data/MC Comparisons

The characteristics of neutrino-produced hadronic systems have been extensively studied by several bubble chamber experiments. The bubble chamber technique is well suited for studying details of charged hadron production in neutrino interactions since the detector can provide precise information for each track. However, the bubble chamber has disadvantages for measurements of hadronic system characteristics as well. The detection of neutral particles, in particular of photons from π^0 decay, was difficult for the low density hydrogen and deuterium experiments. Experiments that measured neutral pions typically used heavily liquids such as neon-hydrogen mixtures and Freon. While these exposures had the advantage of higher statistics and improved neutral particle identification, they had the disadvantage of introducing

intranuclear rescattering which complicates the extraction of information related to the hadronization process itself.

We tried to distill the vast literature and focus on the following aspects of $\nu/\bar{\nu}$ measurements made in three bubble chambers - the Big European Bubble Chamber (BEBC) at CERN, the 15-foot bubble chamber at Fermilab, and the SKAT bubble chamber in Russia. Measurements from the experiments of particular interest for tuning purposes can be broadly categorized as multiplicity measurements and hadronic system measurements. Multiplicity measurements include averaged charged and neutral particle (π^0) multiplicities, forward and backward hemisphere average multiplicities and correlations, topological cross sections of charged particles, and neutral - charged pion multiplicity correlations. Hadronic system measurements include fragmentation functions (z distributions), x_F distributions, p_T^2 (transverse momentum squared) distributions, and $x_F - \langle p_T^2 \rangle$ correlations ("seagull" plots).

The systematic errors in many of these measurements are substantial and various corrections had to be made to correct for muon selection efficiency, neutrino energy smearing, *etc.* The direction of the incident $\nu/\bar{\nu}$ is well known from the geometry of the beam and the position of the interaction point. Its energy is unknown and is usually estimated using a method based on transverse momentum imbalance. The muon is usually identified through the kinematic information or by using an external muon identifier (EMI). The resolution in neutrino energy is typically 10% in the bubble chamber experiments and the invariant hadronic mass W is less well determined.

The differential cross section for semi-inclusive pion production in neutrino interactions

$$\nu + N \rightarrow \mu^- + \pi + X \tag{2.7}$$

may in general be written as:

$$\frac{d\sigma(x, Q^2, z)}{dx dQ^2 dz} = \frac{d\sigma(x, Q^2)}{dx dQ^2} D^\pi(x, Q^2, z), \tag{2.8}$$

where $D^\pi(x, Q^2, z)$ is the pion fragmentation function. Experimentally D^π is determined as:

$$D^\pi(x, Q^2, z) = [N_{ev}(x, Q^2)]^{-1} dN/dz. \tag{2.9}$$

In the framework of the Quark Parton Model (QPM) the dominant mechanism for reactions (2.7) is the interaction of the exchanged W boson with a d-quark to give a u-quark which fragments into hadrons in neutrino interactions, leaving a di-quark spectator system which produces target fragments. In this picture the fragmentation function is independent of x and the scaling hypothesis excludes a Q^2 dependence; therefore the fragmentation function should depend only on z. There is no reliable way to separate the current fragmentation region from the target fragmentation region if the effective mass of the hadronic system (W) is not sufficiently high. Most experiments required $W > W_0$ where W_0 is between 3 GeV/c^2 and 4 GeV/c^2 when studying the fragmentation characteristics. The caused difficulties in the tuning of our model because we are mostly interested in the interactions at low hadronic invariant masses.

We determined the parameters in our model by fitting experimental data with simulated CC neutrino free nucleon interactions uniformly distributed in the energy range from 1 to 61 GeV. The events were analyzed to determine the hadronic system characteristics and compared with published experimental data from the BEBC, Fermilab 15-foot, and SKAT bubble chamber experiments. We reweight our MC to the energy spectrum measured by the experiment if that information is available. This step is not strictly necessary for the following two reasons: many observables (mean multiplicity, dispersion, *etc.*) are measured as a function of the hadronic invariant mass W, in which case the energy dependency is removed; secondly the scaling variables (x_F, z, *etc.*) are rather independent of energy according to the scaling hypothesis.

Some experiments required $Q^2 > 1\text{GeV}^2$ to reduce the quasi-elastic contribution, $y < 0.9$ to reduce the neutral currents, and $x > 0.1$ to reduce the sea-quark contribution. They often applied a cut on the muon momentum to select clean CC events. We apply the same kinematic cuts as explicitly stated in the papers to our simulated events. The hadronization model described here is used only for continuum production of hadrons, resonance-mediated production is described as part of the resonance model [36]. Combining resonance and non-resonant inelastic contributions to the inclusive cross section requires care to avoid double-counting [94], and the underlying model used here includes a resonant contribution which dominates the cross section at threshold, but whose contribution gradually diminishes up to a cutoff value of W=1.7 GeV/c^2, above which only non-resonant processes contribute. All of the comparisons shown in this paper between models and data include the resonant contribution to the models unless it is explicitly excluded by experimental cuts.

Fig.2.4 shows the average charged hadron multiplicity $\langle n_{ch} \rangle$ (the number of charged hadrons in the final state, $i.e.$ excluding the muon) as a function of W^2. $\langle n_{ch} \rangle$ rises linearly with $\ln(W^2)$ for $W > 2\text{GeV}/c^2$. At the lowest W values the dominant interaction channels are single pion production from baryon resonances:

$$\nu + p \quad \rightarrow \quad \mu^- + p + \pi^+ \tag{2.10}$$
$$\nu + n \quad \rightarrow \quad \mu^- + p + \pi^0 \tag{2.11}$$
$$\nu + n \quad \rightarrow \quad \mu^- + n + \pi^+ \tag{2.12}$$

Therefore $\langle n_{ch} \rangle$ becomes 2 (1) for νp (νn) interactions as W approaches the pion production threshold. For νp interactions there is a disagreement between the two measurements especially at high invariant masses, which is probably due to differences in scattering from hydrogen and deuterium targets. Our parameterization of low-W model was based on the Fermilab 15-foot chamber data. Historically the PYTHIA/JETSET program was tuned on the BEBC data. The AGKY model uses the KNO-based empirical model at low invariant masses and it uses the PYTHIA/JETSET program to simulation high invariance mass interactions. Therefore the MC prediction agrees better with the Fermilab data at low invariant masses and it agrees better with the BEBC data at high invariant masses.

Fig.2.5(a) shows the dispersion $D_- = (\langle n_-^2 \rangle - \langle n_- \rangle^2)^{1/2}$ of the negative hadron multiplicity as a function of $\langle n_- \rangle$. Fig.2.5(b) shows the ratio $D/\langle n_{ch} \rangle$ as a function of W^2. The dispersion is solely determined by the KNO scaling distributions shown in Fig.2.2. The agreement between data and MC predictions is satisfactory.

Fig.2.6(a) shows the average π^0 multiplicity $\langle n_{\pi^0} \rangle$ as a function of W^2. Fig.2.6(b) shows the dispersion of the distributions in multiplicity as a function of the average multiplicity of π^0 mesons. As we mentioned it is difficult to detect π^0's inside a hydrogen bubble chamber. Also shown in the plot are some measurements using heavy liquids such as neon and Freon. In principle, rescattering of the primary hadrons can occur in the nucleus. Some studies of inclusive negative hadron production in the hydrogen-neon mixture and comparison with data obtained by using hydrogen targets indicate that these effects are negligible [98]. The model is in good agreement with the data. $\langle n_{\pi^0} \rangle$ is $0(1/2)$ for $\nu p(\nu n)$ interactions when the hadronic invariant mass approaches the pion production threshold, which is consistent with the expectation from the reactions (2.10-2.12). The model predicts the same average π^0 multiplicity for νp and νn interactions for $W > 2\text{GeV}/c^2$.

Fig.2.7 shows the average π^0 multiplicities $\langle n_{\pi^0} \rangle$ as a function of the number of negative hadrons n_- for various W ranges. At lower W, $\langle n_{\pi^0} \rangle$ tends to decrease with n_-, probably because of limited phase space, while at higher W $\langle n_{\pi^0} \rangle$ is rather independent of n_- where there is enough phase space. Our model reproduces the correlation at lower W suggested by the data. However, another experiment measured the same correlation using neon-hydrogen mixture and their results indicate that $\langle n_{\pi^0} \rangle$ is rather independent of n_- for both $W > 4\text{GeV}/c^2$ and $W < 4\text{GeV}/c^2$ [99]. Since events with π^0 but with 0 or very few charged pions are dominant background events in the ν_e appearance analysis, it is very important to understand the correlation between the neutral pions and charged pions; this should be a goal of future experiments [100].

Fig.2.8 shows the average charged-hadron multiplicity in the forward and backward hemispheres as functions of W^2. The forward hemisphere is defined by the direction of the current in the total hadronic c.m.s. There is a bump in the MC prediction in the forward hemisphere for νp interactions at $W \sim 2\text{GeV}/c^2$ and there is a slight dip in the backward hemisphere in the same region. This indicates that the MC may overestimate the hadrons going forward in the hadronic c.m.s. at $W \sim 2\text{GeV}/c^2$ and underestimate the hadrons going backward. One consequence could be that the MC overestimates the energetic hadrons since the hadrons in the forward hemisphere of hadronic c.m.s. get more Lorentz boost than those in the backward hemisphere when boosted to the LAB frame. This may be caused by the way we determine the baryon 4-momentum and preferably select events with low p_T in the phase space decay. These effects will be investigated further for improvement in future versions of the model.

The production of strange particles via associated production is shown in Figures 2.9 and 2.10. The production of kaons and lambdas for the KNO-based model are in reasonable agreement with the data, while the rate of strange meson production from JETSET is clearly low. We have investigated adjusting JETSET parameters to produce better agreement with data. While it is possible to improve the agreement with strange particle production data, doing so yields reduced agreement with other important distributions, such as the normalized charged particle distributions.

Fig.2.11 shows the fragmentation functions for positive and negative hadrons. The fragmentation function is defined as: $D(z) = \frac{1}{N_{ev}} \cdot \frac{dN}{dz}$, where N_{ev} is the total number of interactions (events) and $z = E/\nu$ is the fraction of the total energy transfer carried by each final hadron in the laboratory frame. The AGKY predictions are in excellent agreement with the data.

Fig.2.12 shows the mean value of the transverse momentum with respect to the current direction of charged hadrons as a function of W. The MC predictions match the data reasonably well. In the naive QPM, the quarks have no transverse momentum within the struck nucleon, and the fragments acquire a P_T^{frag} with respect to the struck quark from the hadronization process. The average transverse momentum $\langle P_T^2 \rangle$ of the hadrons will then be independent of variables such as x_{BJ}, y, Q^2, W, *etc.*, apart from trivial kinematic constraints and any instrumental effects. Both MC and data reflect this feature. However, in a perturbative QCD picture, the quark acquires an additional transverse component, $\langle P_T^2 \rangle^{QCD}$, as a result of gluon radiation. The quark itself may also have a primordial $\langle P_T^2 \rangle^{prim}$ inside the nucleon. These QCD effects can introduce dependencies of $\langle P_T^2 \rangle$ on the variables x_{BJ}, y, Q^2, W, z, *etc.*

Fig.2.13 shows the mean value of the transverse momentum of charged hadrons as a function of x_F, where $x_F = \frac{p_L^*}{p_{Lmax}^*}$ is the Feynman-x. As is well known, $\langle p_T \rangle$ increases with increasing $|x_F|$ with a shape called the seagull effect. This effect is reasonably well modeled by the AGKY model.

Conclusions

In this section we have described the GENIE hadronic mutiparticle production model tuned for experiments in the few-GeV energy regime. The model exhibits satisfactory agreement with wide variety of data for charged, neutral pions as well as strange particles. Several upcoming expriments will have high-statistics data sets in detectors with excellent energy resolution, neutral particle containment, and particle identification. These experiments are in some cases considering possible running with cryogenic hydrogen and deuterium targets. These experiments will be operating in this few-GeV regime and have the potential to fill in several gaps in our understanding that will help improve hadronic shower modeling for oscillation experiments.

The upcoming generation of experiments have all the necessary prerequisites to significantly address the existing experimental uncertainties in hadronization at low invariant mass. These result from the fact that these detectors have good containment for both charged and neutral particles, high event rates, good tracking resolution, excellent particle identification and energy resolution, and the possibility of collecting data on free nucleons with cryogenic targets. The latter offers the possibility of addressing the challenge of disentangling hadronization modeling from intranuclear rescattering effects. Charged current

measurements of particular interest will include clarifying the experimental discrepancy at low invariant mass between the existing published results as shown in Fig.2.8, the origin of which probably relates to particle misidentification corrections [96]. This discrepancy has a large effect on forward/backward measurements, and a succesful resolution of this question will reduce systematic differences between datasets in a large class of existing measurements. In addition, measurements of transverse momentum at low invariant masses will be helpful in model tuning. Measurements of neutral particles, in particular multiplicity and particle dispersion from free targets at low invariant mass, will be tremendously helpful. The correlation between neutral and charged particle multiplicities at low invariant mass is particularly important for oscillation simulations, as it determines the likelihood that a low invariant mass shower will be dominated by neutral pions.

Figure 2.1: $\nu\mu$ charged current scattering from an isoscalar target. The shaded band indicates the estimated uncertainty on the free nucleon cross section. Data are from [53] (CCFRR), [54] (CDHSW), [55] (GGM-SPS), [56, 57] (BEBC), [58] (ITEP), [59] (CRS, SKAT), [60] (ANL), [61] (BNL) and [62] (GGM-PS).

Figure 2.2: νp (left) and νn interactions. The curve represents a fit to the Levy function. Data points are taken from [83].

Figure 2.3: Nucleon x_F distribution data from Cooper *et al.* [90] and the AGKY parametrization (solid line).

Figure 2.4: $\langle n_{ch} \rangle$ as a function of W^2. (a) νp events. (b) νn events. Data points are taken from [83, 93].

Figure 2.5: $D_- = (\langle n_-^2 \rangle - \langle n_- \rangle^2)^{1/2}$ as a function of $\langle n_- \rangle$. (b) $D/\langle n_{ch} \rangle$ as a function of W^2. Data points are taken from [83].

Figure 2.6: π^0 mesons as a function of W^2. (b) Dispersion of the distributions in multiplicity as a function of the average multiplicity of π^0 mesons. Data points are taken from [87, 95, 96]

Figure 2.7: π^0 multiplicity $\langle n_{\pi^0} \rangle$ as a function of the number of negative hadrons n_- for different intervals of W. Data points are taken from [96].

Figure 2.8: W^2: (a) νp, forward, (b) νp, backward, (c) νn, forward, (d) νn, backward. Data points are taken from [83, 96, 97].

Figure 2.9: [101, 102, 103].

Figure 2.10: [101, 102, 103].

Figure 2.11: $W^2 > 5(GeV/c^2)^2$, $Q^2 > 1(GeV/c)^2$. Data points are taken from [97].

Figure 2.12: W for the selections (a) $x_F > 0$, (b) $x_F < 0$, and (c) all x_F. Data points are taken from [104].

Figure 2.13: x_F for $\bar{\nu}p$. (a) $W < 4\text{GeV}/c^2$, (b) $W > 4\text{GeV}/c^2$. Data points are taken from [104].

2.5 Intranuclear Hadron Transport

The hadronization model describes particle production from free targets and is tuned primarily to bubble chamber data on hydrogen and deuterium targets [83, 89, 93, 95, 96, 97, 98, 99]. Hadrons produced in the nuclear environment may rescatter on their way out of the nucleus, and these reinteractions significantly modify the observable distributions in most detectors.

It is also well established that hadrons produced in the nuclear environment do not immediately reinteract with their full cross section. The basic picture is that during the time it takes for quarks to materialize as hadrons, they propagate through the nucleus with a dramatically reduced interaction probability. This was implemented in GENIE as a simple 'free step' at the start of the intranuclear cascade during which no interactions can occur. The 'free step' comes from a formation time of 0.342 fm/c according to the SKAT model [105].

Intranuclear hadron transport in GENIE is handled by a subpackage called INTRANUKE. IN-TRANUKE is an intranuclear cascade simulation and has gone through numerous revisions since the original version was developed for use by the Soudan 2 Collaboration [106]. The sensitivity of a particular experiment to intranuclear rescattering depends strongly on the detector technology, the energy range of the neutrinos, and the physics measurement being made. INTRANUKE simulates rescattering of pions and nucleons in the nucleus. When produced inside a nucleus, hadrons have a typical mean free path (MFP) of a few femtometers. Detectors in a neutrino experiment are almost always composed of nuclei today. Therefore, the hadrons produced in the primary interaction (what the neutrino does directly) often (e.g. ∼30% in iron for few GeV neutrinos) have a FSI. There are many possibilities from benign to dangerous. For example, a quasielastic (QE) interaction that emits a proton can end up with a final state of 3 protons, 2 neutrons, and a few photons with finite probability. For a 1 GeV muon neutrino QE interaction in carbon, the probability of a final state different than 1 proton is 35% (GENIE). A possibility even worse is a pion production primary interaction where the pion is absorbed. Such an event occurs for 20% (GENIE) of pion production events and can look like a QE event. At minimum, the wrong beam energy will be measured for these events as the topology is often mistaken. A high quality Monte Carlo code is the only way to understand the role of these events. Fig. 2.14 shows the pion energies that are relevant to a $\nu_\mu C$ experiment at 1 GeV; we must understand the interactions of pions of up to about 0.8 GeV kinetic energy. We see that the Δ resonance dominates the response for pion production, but provides only about half of all pions. Fig. 2.15 shows that the spectrum of pions is significantly altered by FSI.

The best way to understand the effects of FSI is to measure the *cross sections* for as many final states as possible with neutrino beams. At this time, the storehouse for this kind of data is very bare. Dedicated cross section experiments such as SciBooNE and MINERvA will bridge this gap, but we will always be dependent on hadron-nucleus and photon-nucleus experiments for some information. These experiments measure very useful properties of hadrons propagating in nuclei. Although hadron beams are always composed free particles, neutrino experiments need the properties of hadrons produced off-shell in the nucleus. (Pion photoproduction experiments provide useful bridge reactions. Pion FSI are always an important part of all theory calculations for these experiments; the models always come from pion-nucleus data.) The correct attitude is to validate FSI models for neutrino-nucleus with hadron-nucleus data, then use these models to make first predictions of the upcoming dedicated cross section experiments.

Various models are available. Quantum mechanical models for hadron-nucleus experiments would be the most correct, but difficulties in tracking multiple particles make such a calculation impossible. Semi-classical models have some success in describing pion-nucleus interaction data and are now being applied to neutrino interactions [12]. However, intranuclear cascade (INC) models [107, 108, 109, 110] provided the most important means to understand pion-nucleus data where the final state was highly inelastic, i.e. the kinds of data most important for neutrinos. In the semi-classical or INC models, the hadron sees a nucleus of (largely) isolated nucleons (neutrons and protons). The probability of interaction is governed

Figure 2.14: π^+ total cross section resulting from $\nu_\mu^{12}C$ interactions. Different lines show results including all sources, all resonances, and the Δ resonance alone. The nonresonant processes are significant in GENIE.

Figure 2.15: Comparison of the π^+ momentum distribution due to the bare resonance interaction and what is seen in the final state.

by the *free* cross section and the density of nucleons,

$$\lambda(E,r) = \frac{1}{\sigma_{hN,tot} * \rho(r)} \tag{2.13}$$

The actual class of interaction is then chosen according to the cross sections for various reactions for free nucleons, sometimes modified for nuclear medium effects.

Survey of models

Semi-classical models have advanced significantly due to the work of the Giessen group in building a new program called GiBUU [12]. The strong interaction section [111, 22] is the most complete part of the package. The dominant interaction of pions is through resonance formation and they are handled with care. Nucleons in the nucleus are corrected for binding with a local potential well and for Fermi motion with a local Fermi momentum. Resonance production is corrected for the nearby nucleons in a local density approximation. Nonresonant reactions are added by hand. Allowing for the nonlocality of the interaction is an important recent advance. The classical part of the model comes from the use of free cross sections with corrections rather than quantum mechanical amplitudes for interactions. Thus, GiBUU could be called a very sophisticated INC model. The passage of a hadron through the nuclear medium is then handled by a set of coupled integro-differential equations. Thus, required computer resources are significant.

GENIE, NEUT, and FLUKA have more standard INC models. They use free cross sections for interactions but also apply medium corrections of various kinds. These corrections are less complete and more empirical than what is found in GiBUU. These models are most applicable for higher energy hadrons (roughly pions with kinetic energy larger than 300 MeV and nucleons above 200 MeV), where the mean free path is long compared to the inter-nuclear spacing of roughly 1.8fm and the pion Compton wavelength.

Peanut (FLUKA) [112] received a major effort in 1995-9 and is very well adapted to describe processes from 10 MeV to 100 GeV. They include effects such as coherence time, refraction, and pre-equilibrium/compound nucleus processes which simulate known quantum mechanical features. NEUT FSI is based on the work of Salcedo, Oset, Vicente-Vacas, and Garcia-Recio [113]. This is a "Δ dominance" model such as were common in the 1980's when pion-nucleus physics was important in nuclear

physics. It has the advantage of doing a careful job simulating the pion-nucleus interaction through $\Delta(1232)$ intermediate states.

Systematics of hadron-nucleus data

Each nucleus has A nucleons (Z protons $+N$ neutrons). All nuclei of interest to neutrino physics are either bound or slightly unbound. Nuclear densities show saturation because of short range repulsion. Therefore, the typical nucleus is approximately a sphere of radius proportional to $A^{1/3}$. The charge density of light nuclei ($A<20$) is found to be Gaussian or modified Gaussian. Heavier nuclei are described by the Woods-Saxon shape,

$$\rho(r) = N_0 \frac{1}{1 + e^{(r-c)/z}} \tag{2.14}$$

where c describes the size and a describes the width of the surface of a nucleus. For example, c=4.1 fm and z=0.55 fm for ^{56}Fe. To good accuracy, c is the radius where the density falls to half the central value with $c \sim 1.2 fm * A^{1/3}$ and $z \sim 0.55 fm$.

Hadrons interact with nuclei in a variety of ways. We use historical definitions of final states that come from interpretation of experiments. In *elastic scattering*, the final state nucleus is in its ground state and the hadron has same charge as the beam particle. If the hadron scatters inelastically, the residual nucleus can be in the ground state or the nucleus can break apart. At low excitation energies ($< \sim 10$ MeV), the residual nucleus decays to a photon and the ground state. (This is important in analysis of SuperKamiokande data.) At higher excitation energies, one or more nucleons are emitted. Final state interactions increase this number. If there is a hadron of the same type but different charge in the final state, we call it *charge exchange*. For example, the reaction $\pi^- p \to \pi^0 n$ is very common inside nuclei. As a boson, the pion can disappear inside the nucleus. Pion initiated reactions with no pions in the final state are called *absorption*. (This provides an important background process to neutrino quasielastic scattering.) For incident nucleons, most of these labels apply exactly. Since they can't be absorbed, final states with 2 or more nucleons are called *spallation*. If the hadron has enough energy, a pion (a second pion if the initial hadron is a pion) can be produced in the nucleus. We call those events *pion production*.

For low energy incident particles, these definitions are clean. At higher energies, the states mix and confusion can result. For example, a reaction $\pi^{+12}C \to \pi^+ \pi^0 {}^{12}C$ can be inelastic, charge exchange, or pion production. Definitions we use call it pion production. A way to avoid difficulties is to measure inclusive cross sections; there, the energy and angular distribution of a particular particle are determined. In each case, various reactions are possible but models can be tested without ambiguity.

Because the MFP is so short for hadron interactions, elastic scattering cross sections look very diffractive. In fact, the angular distribution can be calculated with a quantum mechanical model using a black disk for the nucleus. This wave property is very difficult to simulate in a semi-classical or INC model.

Another consequence of the short MFP is seen in the total reaction cross section ($\sigma_{reac} = \sigma_{tot} - \sigma_{elas}$). For hadrons, this is close to the nuclear size, πR^2. For example, σ_{reac} for protons and neutrons of 0.4-1GeV is flat at a value of about $300mb=30fm^2$ for carbon and about $80fm^2$ for iron. These corresponds to a radius, R, of about 3 and 5 fm. These values are close to the radius where the nuclear density is about half of the central value. If we divide these values by $A^{1/3}$, the result is close to the commonly used value of 1.2 fm. The pion-nucleus reaction cross section at kinetic energies of about 85-315 MeV is dominated by the effect of the $\Delta(1232)$ resonance. Thus, the effective size of the nucleus here is at a radius where the density is about 1% of the central density. For total cross sections, the A dependence is often a power relation, $\sigma \propto A^\alpha$, but α will vary from the expected value of 2/3 due to more complicated dynamics. The total cross section for pion-nucleus has a power of about 0.8 for a wide range in energy. The A dependence of α[114, 115] varies between 0.55 and 0.8 for the components of the total reaction cross section as a function of energy and process.

Nevertheless, many inelastic cross sections have prominent contributions from *quasifree* interactions. Here, the hadrons in the final state have the kinematics as though they came from a single interaction between the incident particle and a nucleon in the nuclear medium. The name comes from the fact that nucleons in the nuclear medium are in a bound state and therefore not free. If the nucleon were free, the scattered particles would have a single energy at each angle. The struck nucleons have momentum (called Fermi motion), giving particles a range of momentum at a given angle. The largest momentum a nucleon can have is well-defined in the Fermi Gas model, is approximate in real nuclei. It is called the *Fermi momentum* and its value is approximately 250 MeV/c. In heavy nuclei, the average binding energy is about 25 MeV. Thus, the peak due to quasifree scattering from a bound nucleon is shifted by about 40 MeV from the free case and the width is roughly 100 MeV.

This process has been widely studied for electron and pion probes. If it could be studied with neutrinos, the same structure would be seen. The so-called quasielastic peak is prominent in the inclusive scattering cross section. At high excitation energies (lower kinetic energy for the scattered particle), a second peak is found for quasifree pion production from a bound nucleon. Final state interactions are more important in the details in this case. Consider the case of π^+ interactions in carbon at 245 MeV. Evidence for quasifree pion scattering is strong. A scattered π^+ is tagged on one side of the beam and the spectrum of protons is measured on the other side. A prominent peak is seen close to the angle where protons would be if the target was a free proton. The same correlation is seen between 2 protons where the π^+ is absorbed on a quasideuteron in the nuclear medium. Strong evidence for quasifree pion scattering and absorption is seen. Calculations with an INC model are in excellent agreement with these data.

The energy distribution of π^+ detected at 130° [116] shows a peak close to where scattering from a free proton would be seen. Since Fig. 2.16 is for a H_2O target, scattering from H is seen as a gap at about 130 MeV (cross section is too large to show). Pions interacting with oxygen nuclei produce a peak at about 100 MeV. Calculations show it is dominated by events with a single scattering (S). At low energies, the distribution is modified by events with more than one scattering (M). At forward angles, the contributions from multiple scattering aremore important.

If incident particles have a higher energy, complications can be found. With light targets, FSI effects are small and quasifree scattering and pion production peaks are seen. However, INC calculations have trouble getting the shape right, particularly in the region between the peaks. Fig. 2.17 is for π^- scattering from ^{12}C at 500 MeV [117]. For π^+ absorption, the quasifree process would be $\pi^+ d \to pp$ since pions are highly unlikely to be absorbed on a single nucleon. LADS data [118] for π^+ absorption in Ar (A=40) shows the largest strength for the pp final state but this is less than half of the total cross section.

INC models

Prominence of the quasifree reaction mechanism shows why INC models are valuable. These models assume the nucleus is an ensemble of nucleons which have Fermi motion and binding energy. The incident particle interacts in a series of encounters with single nucleons called a cascade (see Figs. 2.18, 2.19).

All interactions are governed by the cross section for the free process, e.g. $\pi^+ n \to \pi^+ n$ or $pp \to pp$. Probability of interaction is governed by a mean free path according to Eqn. 2.13. Cross sections for pions, kaons, protons, and photons interacting with free nucleons are fit with a partial wave analysis with results provided by the GWU group [119, 120]. Nucleon densities come from compilations; note that neutron and protons have very similar densities even for nuclei such as lead.

The problems with INC models must be considered. Since interactions are governed by cross sections rather than quantum mechanical amplitudes, the nuclear model is often very simple. The simplest and most general nuclear model is the Fermi gas which is the basis for all neutrino-nucleus event generator models. Effects of nucleon correlations must be included empirically. Both the struck nucleon and the scattered hadron are likely to be off-shell. Although this effect has been shown to be 'moderate', it is difficult to simulate in a semi-classical model. Thus, there is no definite prescription for an INC model;

Figure 2.16: Inclusive π^+ scattering data from Ingram, et al. compared with separate curves for single and multiple scattering contributions.

Figure 2.17: Inclusive π^- scattering data from Zumbro, et al. compared with INC calculations of Mashnik, et al.

Figure 2.18: Schematic diagram for reaction involving typical FSI process.

Figure 2.19: Schematic diagram for reaction where pion is produced then absorbed in the same nucleus.

Figure 2.20: Mashnik, et al. INC calculations compared with McKeown, et al. data.

Figure 2.21: Mashnik, et al. INC calculations compared with Iwamoto, et al. data.

many versions exist with a wide range of applicability.

The successes of INC models are large. For many reactions, they are the only models available for comparison. They were first used for pion production in proton-nucleus interactions by Metropolis and Harp [107]. A general INC model (CEM03) developed by Mashnik and collaborators [108, 109, 110] has been applied with success to a wide range of pion- and proton-nucleus data [121]. Examples are shown in Figs. 2.17, 2.20 and 2.21; we will show similar comparisons for the GENIE FSI model.

The FSI model in FLUKA is PEANUT. This uses a more sophisticated INC model than CEM03. Various nuclear and quantum mechanical corrections are applied. The result is impressive agreement with a wide variety of data.

Treatment of pion absorption is somewhat different in the INC models than the Δ dominance models. In the latter, pions first rescatter off a nucleon (off-shell) and then absorbed on another. There are other mechanisms which should be included. Salcedo, Oset, Vicente-Vacas, and Garcia-Recio [113] include both S-wave absorption and 3-body absorption. In INC, the fundamental process for pion absorption is $\pi^+ d \to pp$ and this is often the only process included. Since the density of nucleons is much smaller in deuterium as compared with real nuclei, an empirical factor (with a value often about 3) must be included.

The INTRANUKE / hA FSI model

The first FSI model is in the spirit of the other models in GENIE. It is simple and empirical, data-driven. Rather than calculate a cascade of hadronic interactions as is done in a complete INC model, we use the total cross section for each possible nuclear process for pions and nucleons as a function of energy up to 1.2 GeV. Thus, it is called hA. The emphasis is on iron because the first application was to MINOS where production of high energy pions is important. At low energies (50-300 MeV), there is sufficient data [114, 115, 122, 123, 124] for a good description. At high energies, only a few data points are available. Here, we use results obtained for the CEM03 model. Although the calculations are complete, they are not in good agreement with the existing total cross section data. Therefore, the calculations are normalized

Figure 2.22: π^+Fe reactions used in GENIE hA model. Final states are chosen according to these values.

Figure 2.23: Same for pFe reactions.

to the data at low energies. Elastic data at high energy are used to extrapolate the model to 1.2 GeV.

The hA model also handles proton and neutron rescattering. The same reactions are possible except that neither can be absorbed. Still, multinucleon knockout is highly probable. Although much less data is available for nucleons than pions, CEM03 was tuned primarily for them.

The values used for π^+ and p are shown in Figs. 2.22 and 2.23. Data values are used for energies below 315 MeV for all cross sections. Total cross section data is available across the entire range. Data for total and total reaction cross sections are used across the entire range in energy. Cross sections for targets other than iron are obtained by scaling by $A^{2/3}$. As discussed above, this is a reasonable approximation. Because such a large range is covered, processes such as pion production must be included. Here, we use the CEM03 calculations.

The total cross section is calculated from the mean free path and can be checked against data. In addition the accuracy of the $A^{2/3}$ scaling can be checked with data from another target. We show the total and component cross sections for the model compared with carbon data in Figs. 2.24 and 2.25. (Agreement for iron has less information and is equal in quality.)

All the data points in Figs. 2.24 and 2.25 have error bars. These are either taken from the data or estimated. These provide the range of values sampled during reweighting exercises. This is an excellent way to estimate model dependent errors in a neutrino oscillation experiment (see 'Event Reweighting' chapter). The ability to reweight is an important feature of this model.

This is the default FSI model in GENIE v2.4.0, the public version as of now. It uses identical cross section for π^+ and π^- and for p and n. For isoscalar targets (e.g. ^{12}C and ^{16}O, this is no issue for the pions because of isospin symmetry. For targets such as lead, this is a 10% effect. The charges of particles in the final state tend to reflect the charge of the probe. For example, final states for π^+ have more protons than neutrons while the opposite holds for π^-. Cross sections for π^0 beams can't be measured. This code uses isospin symmetry to calculate them from the charged pions. The total reaction cross sections for p and n are very similar, plots are shown in the next section. Charges of final state particles tend to be more positive for incident protons.

Figure 2.24: π^+C reactions. GENIE hA model is used. Total cross section is determined with proper mean free path in a carbon nucleus.

Figure 2.25: Component cross sections come from the corresponding iron cross sections scaled by $A^{2/3}$.

Pion absorption and nucleon spallation reactions can knock out large numbers of nucleons. This is seen strongly in data. More detailed calculations (see below) show an average of 10 nucleons ejected from iron in pion absorption. To simplify the code, the hA model limits this to 5. For MINOS, this is never an issue.

Angular and energy distributions of particles are estimated. For elastic scattering, template angular distributions from relevant data are used. These distributions are very forward peaked, so it's not an important simplification. For final states with more than 1 hadron, particles are distributed by phase space. This gives the correct limits, but the energy distribution changes somewhat when the Δ resonance dominates. The effect of these approximations have not yet been simulated, but they are unlikely to be an important effect in the MINOS experiment. One of the most significant errors is in the treatment of the quasielastic scattering. Only the incident particle is put in the final state and it's energy and angle distribution are both flat.

Since the elastic cross section can't be generated in an INC model, it has to be added on. For the hA model, we chose an empirical method. The size of the nucleus is increased by ΔR which is proportional to the de Broglie wavelength. This nicely matches the data for all energies.

Almost all of the problems in the last paragraphs will be fixed in GENIE v2.6.0. Changes due to isospin in either hadron or nucleus will be greatly improved. The number of final states sampled will be increased. Inelastic final states will be assumed to be dominated by quasielastic events. (This approximation can be checked against data and will be discussed in the next section.)

The INTRANUKE / hN FSI model

The second FSI model in GENIE (hN) is a full INC model. It includes interactions of pions, nucleons, kaons, and photons in all nuclei. The basis is the angular distributions as a function of energy for about 14 reactions from threshold to 1.2 GeV. All this information comes from the GWU group [119, 120]. A preliminary version of the hN model is scheduled to be in GENIE v2.6.0, but the hA model will still be

Figure 2.26: pFe reactions from the GENIE hN model.

Figure 2.27: Total reaction cross sections for nC reactions from the GENIE hN model.

the default.

As a full INC model, all reactions on all nuclei can be calculated. None of the restrictions that apply to hA model are relevant. Although the choice of interaction points through the MFP is identical in the 2 models, the cascade is fully modeled in the hN model. For example, there is a small but finite probability of knocking out every nucleon in an event.

One new feature of this code is the inclusion of nucleon pre-equilibrium and compund nuclear processes. The present model is simple, but effective. This is important to give an improved description of the vertex energy deposition.

The code was designed to minimize the number of parameters. One parameter scales the absorption MFP and is fit to the pion total absorption cross section. Separate values for the ΔR values for pions and nucleons are fit to the total reaction cross sections. All particles get a free step when they are produced; this simulates the effect of Δ resonance propagation in a simple way. It is used to adjust the normalization of certain inclusive scattering distributions. A shift in the energy of nucleons in the nucleus is used to put the quasielastic peak (see Fig. 2L) (similar to what is used in electron scattering).

The validation of this new code comes in 2 parts- the total cross sections for various processes (e.g. Fig. 2.24) which test the overall propagation of particles and the inclusive cross sections (e.g. Figs. 2.17, 2.20 and 2.21). Each is important. Previous validations emphasize the total cross sections because this sets the overall flow of particles into each topology. Previous neutrino experiments emphasize topology. Future experiments are expected to put emphasis into the distribution of particles in energy and angle as beam and detector technology improve.

The component total cross section data is limited to hadron energy of less than $\sim$350 MeV. The exception is the total reaction cross section which has been measured for π^+, π^-, p, and n up to roughly 1 GeV. Figs. 2.26 and 2.27 show σ_{reac} for protons in iron and neutrons in carbon respectively. The energy dependence is flat and we see the cross section approximately equal to the nuclear area as discussed in the introduction. Agreement of the model is excellent.

In Figs. 2.28 and 2.29, we show σ_{reac} for pions. The agreement is excellent except at low energies for heavier targets; this is still under study.

With the significant interest in absorption, we show 2 examples of that total cross section in Figs. 2.30 and 2.31. Overall agreement is very good, but the problem in σ_{reac} at low energies for heavy targets is shown to be in the absorption channel.

Continuing with absorption, we show 2 examples similar to Figs. 2.20 and 2.21 in Figs. 2.32 and 2.33. The agreement shown here is excellent as the details of pion reactions are explored across a wide

Figure 2.28: π^+C using the GENIE hN model compared to data.

Figure 2.29: Total reaction cross sections for π^+Pb using the GENIE hN model.

Figure 2.30: π^+C using the GENIE hN model compared to data.

Figure 2.31: Total reaction cross sections for π^+Pb using the GENIE hN model.

Figure 2.32: π^+ interacting in various nuclei. In each distribution, protons are detected at $30°$. Data is from McKeown, et al. These protons come from both absorption and scattering processes.

Figure 2.33: Inclusive cross sections for neutrons emitted from 870 MeV π^+ interacting in iron. In each case, neutrons are detected at different angles. Data is from Iwamoto, et al. These neutrons come from predominantly the absorption process.

kinematic range.

The last example of this new code is for scattering processes. When hadrons interact in the nuclear medium, the quasifree scattering process is important; that has been seen in numerous data sets. In Figs. 2.34 and 2.35 we show examples for pion and proton scattering. For the pion case, a back angle is shown; here, the quasielastic mechanism dominates. For protons, the beam energy is large enough that the multiple scattering process is sampled over a wide range in energy. The agreement is excellent.

Conclusions

We have reviewed strong interactions as they will be applied to neutrino experiments of the near future. The basic premise is that hadron-nucleus experiments are the best way (definitely now, likely also in the future) to validate FSI models. General properties have been identified from data, the general blackness of nuclei to hadrons along with the importance of quasifree mechanisms.

Various models were discussed with a focus on INC models. Although, they are not the most theoretically viable, the role of INC models is significant because they can 'easily' describe a wide range of data.

The 2 FSI models in GENIE are described in some detail. The hA model is simpler and more empirical. Although it isn't the most accurate, it is very fast and straightforward to reweight. The hN model is a full INC calculation which is much more accurate. In v2.4, the hA model is the only FSI model. For version 2.6, both will be included but hA will still be the default. The hA model will be applicable to all nuclei from helium to lead for kinetic energies up to 1.2 GeV for pions and nucleons. Its main value will be for high energy neutrinos and in reweighting. The hN model is nearly complete for this round. It will be valid for energies above 50 MeV and will provide a very complete description of many final states.

Thus, each GENIE FSI model has independent validity. An important component of any simulation is the estimation of systematic errors. A comparison of the results using each model can show model dependence. Varying parameters inside the hA model is the best way to assess systematic errors due to FSI.

Figure 2.34: π^+ scattered at 130° from 240 MeV π^+ interacting with oxygen. At this back angle, the spectrum of π^+ is dominated by the quasifree mechanism. Data is from Ingram, et al.

Figure 2.35: Inclusive cross sections for protons scattered at 20° from 800 MeV protons interacting with calcium. Data is from McGill, et al and Chrien, et al. There is a known absolute normalization difference between the 2 experiments but it is not available to us.

Chapter 3

The GENIE Software Framework

[to be added in future revision]

Chapter 4

Downloading & Installing GENIE

4.1 Understanding the versioning scheme

SVN Tags

In the GENIE version numbering scheme, releases are tagged in the SVN source-code repository as **R-major_minor_revision**[1]. When a number of significant functionality improvements or additions have been made, the major index is incremented. The minor index is incremented in case of significant fixes and/or minor feature additions. The revision number is incremented for minor bug fixes and updates.

Version number semantics

- Versions with even minor number (eg 2.0.*, 2.4.*) correspond to stable, fully validated physics production releases[2].

- Versions with odd minor number (eg. 2.3.*, 2.5.*) correspond to release candidates tagged during the validation stage preceding the release of a production version.

- Production versions, and candidate releases, always have an even revision number.

- The SVN head has a nominal version number of 999.999.999.

Release codenames

The major production-quality releases are code-named after modern extinct or endangered species (series of production releases: **Auk**, **Blueback**, **Cheetah**, **Dodo**, **Elk**, **Fox**, **Gazelle**, **Hippo**, **Ibex**,...).

Release qualifiers

The GENIE releases are marked as:

- **pro** : Validated production-quality versions recommended for physics studies.

- **old**: Older 'pro' versions that have been greatly superseded by newer versions. Versions marked as 'old' become unsupported. We appreciate that experiments get highly attached on specific versions due to the enormous amount of work invested in generating high statistics samples and calculating

---

[1]For example, tag R-1_99_1 corresponds to GENIE vrs 1.99.1, tag R-2_0_2 corresponds to GENIE vrs 2.0.2 etc.
[2]To the dismay of mathematicians, our versioning scheme uses 0 as am even number.

MC-dependent corrections and systematics. We strive to support 'pro' versions for a minimum of two years.

- **rc**: Release candidates. You may not use for physics studies.

- **special**: Special releases prepared for a particular study or event such as a) the evaluation of an experiment systematic with an appropriately modified version of GENIE, or b) a GENIE tutorial or a summer / winter school. You may not use these releases outside the intended context.

4.2   Obtaining the source code

The official GENIE source code is maintained at a SubVersion repository hosted at HepForge[3]. The development version and a host of frozen physics releases are available from the repository. Alternatively, you can download compressed archives stored at the HepForge archive area, or you can create and download such archives using the web interface to the GENIE SubVersion repository. Details are given below. Further general information can be found at

- The HepForge documentation page: *http://www.hepforge.org/docs/*

- The SVN book: *http://svnbook.red-bean.com/*

The code repository can be accessed anonymously via HTTP, without a HepForge account. You need to have a SubVersion client installed and you probably already do. If not, binaries are readily available for most platforms (see *http://subversion.apache.org/*).

You can check-out the generator SVN trunk by typing:

```
$ svn co http://svn.hepforge.org/genie/generator/trunk <local_dir>
```

You can check-out frozen releases by typing:

```
$ svn co http://svn.hepforge.org/genie/generator/branches/<genie_tag> <local_dir>
```

Make the appropriate substitutions for `<genie_tag>` and `<local_dir>`. To view the available `<genie_tag>`s see the GENIE release table on the web, or just type:

```
$ svn list http://svn.hepforge.org/genie/generator/branches/
```

Alternatively, compressed archives for recent stable version releases are posted at the HepForge archive area:
```
http://www.hepforge.org/downloads/genie
```

You can also download a compressed archive of the latest development version (created automatically upon your request) using the repository's web interface. Visit the repository trunk at:
http://projects.hepforge.org/genie/trac/browser/trunk
Then click on 'Download in other formats: Zip Archive' towards the end of the page.

Write access to the Generator repository, as well as to other GENIE products including the Comparisons and the Tuning require a HepForge account and it is permitted only for GENIE collaborators.

---

[3]http://www.hepforge.org

Special limited accounts may be setup for regular GEBINE contributors.

4.3   3rd Party Sofwtare

A typical GENIE installation[4] requires the following external packages[5]:

- ***ROOT***

- ***GSL***
  The GNU Scientific Library

- ***LHAPDF***
  The Les Houches Accord PDF interface, a PDFLIB successor.

- ***PYTHIA6***
  The well known LUND Monte Carlo package used by GENIE for particle decays and string frag-
  mentation (for neutrino interactions of high invariant mass).

- ***log4cpp***
  A C++ library for message logging.

- *libxml2*
  The C XML library for the GNOME project.

The installation of external packages is described in detail in their corresponding web pages. Additional
detailed instructions can also be found at Appendix E of this manual.

4.4   Preparing your environment

A number of environmental variables need to bee set or updated before using GENIE.

- Set the '`GENIE`' environmental variable to point at the top level GENIE directory

- Set the '`ROOTSYS`' environmental variable to point at the top level ROOT directory

- Set the '`LHAPATH`' environmental variable to point to LHAPDF's PDF data files

- Append '$`ROOTSYS`/*bin*' and '$`GENIE`/*bin*' to your '`PATH`'

- Append '$`ROOTSYS`/*lib*', '$`GENIE`/*lib*' and the paths to the log4cpp, libxml2, LHPADF and PYTHIA6
  libraries to your '`LD_LIBRARY_PATH`' environmental variable (or to your '`DYLD_LIBRARY_PATH`' en-
  vironmental variable if you are using GENIE on MAC OS X).

It is more convenient to create a GENIE setup script and execute it before using GENIE.
A setup script should look like the following:

---

[4]A minimal installation that can be used for event generation / physics studies.
[5]The implicit assumption here is that you start with a 'working system' where some basic tools, such as the gcc compiler
suite, make, autoconf, PERL, CVS and SVN clients etc, are already installed. Instructions are given assuming that you are
using the bash shell but it is trivial to adapt these instructions for your own shell.

```
#!/bin/bash

export GENIE=/path/to/genie/top/directory

export ROOTSYS=/path/to/root/top/directory
export LHAPATH=/path/to/lhapdf/PDFSets/

export PATH=$PATH:\
$ROOTSYS/bin:\
$GENIE/bin

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:\
/path/to/log4cpp/library:\
/path/to/libxml2/library:\
/path/to/lhapdf/libraries:\
/path/to/pythia6/library:\
$ROOTSYS/lib:\
$GENIE/lib
```

Assuming that the above script is named '*genie_ setup*', you can execute it by typing:

```
$ source genie_setup
```

4.5   Configuring GENIE

A configuration script is provided with the GENIE source code to help you configure your GENIE installation (enable / disable features and specify paths to external packages). To see what configuration options are available, type:

```
$ cd $GENIE
$ ./configure --help
```

This will generate a screen output that looks like the following:

```
FLAG                DESCRIPTION

--prefix            Install path (for make install)

enable/disable options
prefix with either --enable or --disable
(eg. --enable-lhapdf --disable-flux-drivers)

profiler            GENIE code profiling using Google perftools.
doxygen-doc         Generate doxygen documentation at build time.
dylibversion        Adds version number in dynamic library names.
lowlevel-mesg       Disable (rather than filter out) some prolific
                    debug level messages known to slow GENIE down.
```

```
debug               Adds -g compiler option to request debug info.
lhapdf              Use the LHAPDF library.
cernlib             Use the CERN libraries.
flux-drivers        Enable built-in flux drivers.
geom-drivers        Enable built-in detector geometry drivers.
mueloss             Muon energy loss modeling.
validation-tools    GENIE physics model validation tools.
test                Build test programs.
t2k                 Enable T2K-specific event generation application.
fnal                Enable FNAL experiment-specific event generation application.
atmo                Enable atmospheric neutrino event generation application.
nucleon-decay       Enable nucleon decay event generation application.
rwght               Enable event reweighting tools.
masterclass         Enable GENIE neutrino masterclass application.


with options for 3rd party software
prefix with --with (eg. --with-lhapdf-lib=/some/path)


optimiz-level       Compiler optimiz. level (O,O2,O3,OO,Os)
profiler-lib        Path to profiler library
doxygen-path        Path to doxygen binary
pythia6-lib         Path to PYTHIA6 library
cern-lib            Path to CERN libraries
lhapdf-inc          Path to LHAPDF includes
lhapdf-lib          Path to LHAPDF libraries
libxml2-inc         Path to libxml2 includes
libxml2-lib         Path to libxlm2 library
log4cpp-inc         Path to log4cpp includes
log4cpp-lib         Path to log4cpp library
```

By default all options required for a minimal installation that can be used for physics event generation are enabled and non-essential features are disabled. Typically, the folowing should be sufficient for most users:

```
$ cd $GENIE
$ ./configure
```

Not specifying any configuration option (like above) is equivalent to specifying:
```
--disable-profiler
--disable-doxygen-doc
--enable-dylibversion
--disable-lowlevel-mesg
--disable-debug
--enable-lhapdf
--disable-cernlib
--enable-flux-drivers
--enable-geom-drivers
--enable-mueloss
--disable-validation-tools
--disable-test
```

```
--disable-t2k
--disable-fnal
--disable-atmo
--disable-nucleon-decay
--disable-rwght
--disable-masterclass
```

The default optimization level is set to O2 and `--prefix` is set to /usr/local.

The configuration script can, in principle, *auto-detect the paths* to required external packages installed at your system if no path is given explicitly. On some occasions, before scanning your system for external products, the configuration script will check whether some rather standard environmental variables have been set (from example, before searching for the PYTHIA6 / JETSET library, the configure script will check whether a '`PYTHIA6`' environmental variable has been set. See '`./configure --help`' for more information).

Obviously, if you want greater control over the configuration options (so that you do not depend on pre-set defaults that may one day change), if you want to modify some other default options or if the script fails to discover some external product path, then do set the configure script options explicitly.

4.6   Building GENIE

Once GENIE has been properly configured, you are ready to build it. Just type:

```
$ cd $GENIE
$ gmake
```

On successful completion you should be able to find many libraries located in *$GENIE/lib* and some applications and scripts in *$GENIE/bin.*

You may stop the building procedure here and start using GENIE now! However, some users may prefer to take their installation one step further and type:

```
$ gmake install
```

If */some/path* was the location specified via the `--prefix` configuration flag, then '`gmake install`' will:

- move all executables and scripts to */some/path/bin,*
- move all libraries to */some/path/lib,* and
- move all headers to */some/path/include/GENIE.*

If you do run '`gmake install`', before running GENIE you need to update your '`LD_LIBRARY_PATH`' (or '`DYLD_LIBRARY_PATH`' on MAC OS X) and '`PATH`' environmental variables accordingly.

Whether you stop the installation procedure after the '`gmake`' or '`gmake install`' step is probably more a matter of personal taste [6]. Whatever you choose should work given that your system's paths have been properly set.

---

[6]I find it easier to manage multiple GENIE installations if I stop after the '`gmake`' step.

Assuming now that the GENIE installation has been completed without apparent errors, we are going to provide instructions for a couple of simple post-installation tests to verify that GENIE has been properly built.

4.7 Performing simple post-installation tests

Here are few simple things you can do in order to try out your installation:

1. Generate a $\nu_\mu + O^{16}$ (ν_μ PDG code: 14, O^{16} PDG code: 1000080160) event sample (10k events) between 0 and 10 GeV, using a simple histogram-based description of the T2K ν_μ flux (ROOT *TH1D* object 'h30000' stored in '*\$GENIE/data/flux/t2kflux.root*'). Use pre-calculated cross-sections (later, you will learn how to calculate these on your own) which can be downloaded from `http://www.hepforge.org/archive/genie/data/`.

   The commands used here will be explained in the next section:

   ```
   $ gevgen -n 10000 -p 14 -t 1000080160 -e 0,10 --run 100
   -f $GENIE/data/flux/t2kflux.root,h30000
   --seed 2989819 --cross-sections /some/path/xsec.xml
   ```

   A '*genie-mcjob-<run number>.status*' status file is created. It is updated periodicaly with job statistics and the most recent event dump. When the job is completed a '*gntp.<run number>.ghep.root*' file, containing the generated event tree, is written-out. To print-out the first 200 events from the event file you just generated, type:
   ```
   $ gevdump -f gntp.100.ghep.root -n 200
   ```

2. Generate a 10,000 event sample of $\pi^+ + O^{16}$ interactions for π^+'s of 200 MeV kinetic energy. (π^+ PDG code: 211, O^{16} PDG code: 1000080160):

   ```
   $ gevgen_hadron -n 10000 -p 211 -t 1000080160 -k 0.2 --seed 9839389
   ```

If everything seems to work then the GENIE is really 'out of the bottle'. Continue reading the Physics and User Manual to find out more about running the GENIE applications bundled in your installation.

Chapter 5

Generating Neutrino Event Samples

5.1 Introduction

[to be added in future revision]

5.2 Preparing event generation inputs: Cross-section splines

When generating neutrino interaction events, most CPU-cycles are spent on calculating neutrino inter-action cross sections. In order to select an interaction channel for a neutrino scattered off a target at a particular energy, the differential cross section for each possible channel is integrated over the kinematic phase space available at this energy. With $\sim 10^2$ possible interaction modes per initial state and with $\sim 10^5$ differential cross section evaluations per cross section integration then $\sim 10^7$ differential cross section evaluations are required just in order to select an interaction channel for a given initial state. Had you been simulating events in a realistic detector geometry ($\sim 10^2$ different isotopes) then the number of differential cross section evaluations, before even starting simulating the event kinematics, would rise to $\sim 10^9$. It is therefore advantageous to pre-calculate the cross section data. The event generation drivers can be instructed to to load the pre-computed data and estimate the cross section by numerical interpolation, rather than by performing numerous CPU-intensive differential cross section integrations. The cross section data are written out in XML format and, when loaded into GENIE, they are used for instantiating *Spline* objects.

5.2.1 The XML cross section splines file format

The XML file format is particularly wekk-suited for moving data between different GENIE applications. This is the only intended usage of these files. If you wish to use GENIE's cross section splines in another context, eg. within your analysis code, then we recommend converting them from XML to ROOT format using utilities provided by GENIE (See Section 5.2.5). Although you should never have to read the XML cross section file, it is generally usefull that you do have an understanding of how it is structured so as to be able to diagnose problems.

All XML splines are stored within '`<genie_xsec_spline_list>`' tags:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by genie::XSecSplineList::SaveSplineList() -->
<genie_xsec_spline_list version="2.00" uselog="1">
```

```
... ... ...
... ... ...
</genie_xsec_spline_list>
```

The 'uselog="1"' flag indicates that the spline knots are spaced 'logarithmically' in energy (This is the default GENIE option so that there is higher knot density where the cross section changes more rapidly). The data for each spline are stored within '`<spline>`' tags[1]:

```
<spline
    name   = "{algorithm/reaction; string}"
    nknots = "{number of knots; int}">
<knot>
  <E>     {energy; double}         </E>
  <xsec> {cross section; double} </xsec>
</knot>
<knot>
  <E>     {energy; double}         </E>
  <xsec> {cross section; double} </xsec>
</knot>
... ...
</spline>
```

Each spline is named by combining the names of the cross section algorithm and its configuration with a string interaction code. These rather long names are built automatically by GENIE and used for retrieving the correct spline[2] from the spline pool. For example, a spline named '*genie::DISPartonModelPXSec/CC-Default/nu:-12;tgt:1000260560;N:2112;q:-1(s);proc:Weak[CC],DIS*' indicates that it was computed using the cross section algorithm '*genie::DISPartonModelPXSec*' run in the '*CC-Default*' configuration for an interaction channel with the following string code: '*nu:-12;tgt:1000260560;N:2112;q:-1(s);proc:Weak[CC],DIS*' (indicating a DIS CC $\nu_\mu Fe^{56}$ scattering process of a sea $\bar{d}$ quark in a bound neutron). The spline knots are listed in increasing energy, going up to a maximum value specified during the spline construction. One of the knots falls exactly on the energy threshold for the given process so as to improve the accuracy of numerical interpolation around threshold. The energy and cross section values are given in the natural system of units ($\hbar = c = 1$) used internally within GENIE (Note that the more widespread cross section units, $10^{-38} cm^2$, are used when the cross section data are exported to a ROOT format for inclusion in user analysis code. See Section5.2.5).

5.2.2 Downloading pre-computed cross section splines

Cross section spline XML files are kept in: *http: // www. hepforge. org/ archive/ genie/ data/*

You need to select the file corresponding to the version of GENIE you are using.

Typically I post cross section spline files for all modeled processes for ν_e, $\bar{\nu}_e$, ν_μ, $\bar{\nu}_\mu$, ν_τ, $\bar{\nu}_\tau$ scattered off free-nucleons (p, n) and off a large set of nuclear targets (the $\sim$ 40 isotopes that can be found in the T2K detector geometries[3]). Using the posted free-nucleon cross section data is easy / fast to calculate

---

[1] In the description below, the curly braces within tags are to be 'viewed' as a single value of the specified type with the specified semantics.

[2] GENIE takes the safest route and checks both the 'reaction mode' and 'cross section algorithm'. It will not use cross section spline data calculated by a cross section algorithm A, if an alternative cross section algorithm B is currently in use.

[3] N^{14}, N^{15}, O^{16}, O^{17}, O^{18}, Al^{27}, C^{12}, C^{13}, H^2, Cl^{35}, Cl^{37}, Pb^{204}, Pb^{206}, Pb^{207}, Pb^{208}, Cu^{63}, Cu^{65}, Zn^{64}, Zn^{66}, Zn^{67}, Zn^{68}, Zn^{70}, Ar^{36}, Ar^{38}, Ar^{40}, Si^{28}, Si^{29}, Si^{30}, B^{10}, B^{11}, Na^{23}, Fe^{54}, Fe^{56}, $Fe57$, Fe^{58}, Co^{59}.

cross section splines for any set of nuclear targets.

Any reasonable request for providing additional cross section splines will be satisfied.

5.2.3 Generating cross section splines

Cross section spline calculation is very CPU-intensive. It is recommended that, for the default GENIE configuration, you use the officially distributed files. However, the information provided in this section will allow you to generate your own cross section spline files, should you need to.

The *gmkspl* spline generation utility

Name

gmkspl – A GENIE utility for generating the cross section splines for a specified set of modeled processes for a specified list of initial states. The cross section splines are written out in an XML file in the format expected by all other GENIE programs.

Source

The source code for this utility may be found in '*$GENIE/src/stdapp/gMakeSplines.cxx*'.

Synopsis

```
$ gmkspl -p neutrino_code <-t target_codes, -f geometry> [-n nknots] [-e max_energy]
[<--output-cross-sections | -o> xml_file] [--input-cross-sections xml_file]
[--seed rnd_seed_num] [--event-generator-list list_name] [--message-thresholds xml_file]
```

where [] marks optional arguments, and <> marks a list of arguments out of which only one can be selected at any given time.

Description

The following options are available:

-p Specifies the neutrino PDG codes.

Multiple neutrino codes can be specified as a comma separated list.

-t Specifies the target PDG codes.

Multiple target PDG codes can be specified as a comma separated list. The PDG2006 conventions is used (10LZZZAAAI). So, for example, O^{16} code = 1000080160, Fe^{56} code = 1000260560. For more details see Appendix D.

-f Specifies a ROOT file containing a ROOT/GEANT detector geometry description.

-n Specifies the number of knots per spline.

By default GENIE is using 15 knots per decade of the spline energy range and at least 30 knots overall.

-e Specifies the maximum neutrino energy in the range of each spline.

   By default the maximum energy is set to be the declared upper end of the validity range of the event generation thread responsible for generating the cross section spline.

–output-cross-sections, -o Specifies the name (incl. full path) of an output cross-section XML file.

   By default GENIE writes-out the calculated cross section splines in an XML file named '*xsec_ splines.xml*' created at the current directory.

–input-cross-sections Specifies the name (incl. full path) of the output XML file.

   An input cross-section file could be specified when it is possible to recycle previous calculations. It is, sometimes, possible to recycle cross-section calculations for scattering off free nucleons when calculating nuclear cross-sections.

–seed Specifies the random number seed for the current job.

   This setting will only ne relevant if MC intergation methods are employed for cross-section calculation.

–event-generator-list List of event generators to load.

   The list of event generators to load affects the list of processes that can be simulated and, for which, cross-section calculations need to be calculated by this application. By default, GENIE is loading a list of of tuned and fully-validated generators which allow comprehensive neutrino interaction modelling the medium-energy range. Valid settings are the XML block names appearing in *$*GENIE*/config/EventGeneratorListAssembler.xml*'. Please, make sure you read Sec. 5.4 explaining why, almost invariantly, for physics studies you should be using a comprehensive collection of event generators.

–message-thresholds Specifies the GENIE verbosity level.

   The verbosity level is controlled with an XML file allowing users to customize the threshold of each message stream. See '*$*GENIE*/config/Messenger.xml*' for the XML schema. The '*Messenger.xml*' file contains the default thresholds used by GENIE. The '*Messenger_ laconic.xml*' and '*Messenger_ rambling.xml*' files define, correspondingly, less and more verbose configurations.

Examples

1. To calculate cross-sections for ν_μ (PDG code: 14) and $\bar{\nu}_\mu$ (PDG code: -14) scattering off Fe^{56} (PDG code: 1000260560), and build splines with 150 knots in the energy range up to 20 GeV, type

   ```
   $ gmkspl -p 14,-14 -t 1000260560 -n 150 -e 20
   ```

   The cross section splines will be saved in an output XML file named '*xsec_ splines.xml* ' (default name).

2. To calculate the CCQE cross-section for ν_μ (PDG code: 14) and $\bar{\nu}_\mu$ (PDG code: -14) scattering off all the targets in the input ROOT geometry file '*/data/mygeometry.root*' and write out the splines in a file named '*mysplines.xml*', type

```
$ gmkspl -p 14,-14 -f /data/mygeometry.root -o mysplines.xml --event-generator-list CCQE
```

Generating cross-section splines is a CPU-intensive task as a large number of processes (see Fig. 5.1) and numerical integration of steeply peaked differential cross-sections over extended, multi-dimensional kinematical phase spaces. When cross-section calculations are needed for multiple targets, it is often impractical to generate all splines in a single job. The task is typically split into smaller jobs which can be run on parallel in a batch farm. Batch submission scripts used by GENIE developers can be found in '*$GENIE/src/scripts/production/batch/*' and easily adapted to match user needs. Detailed documentation is available within the scripts. The multiple XML outputs of all the *gmkspl* jobs can be merged into a single XML file using GENIE's *gspladd* utility. (See Section 5.2.3.) It is worth highlighting that, for faster results, it is preferable if one organizes the jobs as 'single neutrino + multiple nuclear targets' rather than 'multiple neutrinos + single nuclear target': In the former case intermediate, CPU-intensive free-nucleon cross-section calculations, for the given neutrino species, will be recycled in the nuclear target cross-section calculations. For even faster results one can calculate the free-nucleon cross-section splines first, then feed the output into a nuclear cross-section spline calculation. Because of the way nuclear effects are currently handled, nuclear cross-section calculations can recycle CPU-intensive free-nucleon calculations resulting in a dramatic speed improvement. To feed-in free-nucleon cross-sections in a nuclear cross-section calculation job, use the *gmkspl* –input-cross-sections option. Note that, if you feed-in cross-sections, the calculated cross-sections can not extend higher in energy than the input cross-sections.

The *gspladd* spline merging utility

Name

gspladd – A GENIE utility for merging many separate XML cross section files into a single XML file.

Source

The source code for this utility may be found in '*$GENIE/src/stdapp/gSplineAdd.cxx*'.

Synopsis

```
$ gspladd -f file_list -d directory_list -o output_file
```

Description

The following options are available:

-f Specifies input XML files. Multiple input files can be specified as a comma separated list.

-d Specifies input directories. Multiple input files can be specified as a comma separated list. All XML files found in each directory will be included.

-o Specifies the name of the output XML file.

Figure 5.1: Cross section splines just for $\nu_\mu Fe^{56}$ processes modeled in GENIE. The large number of splines and the fine numerical integration stepping makes spline calculation a very CPU-intensive process.

Notes

- At least 2 XML files must be specified as inputs for the *gspladd* application to work.

Examples

1. To merge '*/data/iron/xsec.xml*' and '*/data/oxygen/xsec.xml*' into '*./xsec_ all.xml*', type:

   ```
   $ gspladd -f /data/iron/xsec.xml,/data/oxygen/xsec.xml -o xsec_all.xml
   ```

2. To merge '*./xsec_Fe56.xml*' and all the cross section spline files found in '*/scratch/job1*' and '*/scratch/job2*' into '*./xsec_ all.xml*', type

   ```
   $ gspladd -f xsec_Fe56.xml -d /scratch/job1/,/scratch/job2 -o xsec_all.xml
   ```

5.2.4 Re-using splines for modified GENIE configurations

You should *never* be doing that (unless you are absolutely sure about what you are doing). The safest assumption is that changes in GENIE, either a change of default model parameter or a change of a default model, *invalidates* previously generated cross section splines as the cross section models (used for generating these splines) may be affected.

5.2.5 Using cross section splines in your analysis program

As seen before, GENIE's *gmkspl* utility writes-out cross section values in XML format. While this format is particularly well-suited for moving data between GENIE components, it is not the most usefull format from the perspective of a user who wishes to read and interpolate these cross section data in different contexts within his/her analysis code.

GENIE provides the *gspl2root* utility to convert XML cross section splines into a ROOT formats. The XML cross section data for each process and initial state are converted into a single ROOT *TGraph* objects. All ROOT *TGraph* objects corresponding to the same initial state are written-out in the same ROOT *TDirectory* which is named after the given initial state. Multiple *TDirectory* objects can be saved in a single output ROOT file. ROOT *TGraph* objects support numerical interpolation via the '*TGraph::Eval(double)*' method, so, essentially, **one can write-out all GENIE cross section 'functions' one needs into a single ROOT file**. More details on this particularly useful feature are given next.

The *gspl2root* spline file conversion utility

Name

gspl2root - A GENIE utility to convert XML cross section files into a ROOT format.

Source

The source code for this utility may be found in '*$GENIE/src/stdapp/gSplineXnml2Root.cxx*'.

Synopsis

```
$ gspl2root
   -f input_xml_file
   -p neutrino_pdg_code -t target_pdg_code
   [-e maximum_energy] [-o output_root_file] [-w]
```

where [] denotes an optional argument.

Description

The following options are available:

-**f** Specifies the input XML cross section spline file.

-**p** Specifies the neutrino PDG code.

-**t** Specifies the target PDG code (format: 10LZZZAAAI).

-**e** Specifies the maximum energy for the generated graphs.

-**o** Specifies the output ROOT file name.

-**w** Instructs *gspl2root* to write-out plots in a postscipt file.

Notes

- The spline data written-out have the energies given in *GeV* and the cross sections given in in $10^{-38}cm^2$.

Examples

1. In order to extract all $\nu_\mu+n$, $\nu_\mu+p$ and $\nu_\mu+O^{16}$ cross section splines from the input XML file '*mysplines.xml*', convert splines into a ROOT format and save them into a single ROOT file '*xsec.root*', type:

   ```
   $ gspl2root -f mysplines.xml -p 14 -t 1000000010 -o xsec.root
   $ gspl2root -f mysplines.xml -p 14 -t 1000010010 -o xsec.root
   $ gspl2root -f mysplines.xml -p 14 -t 1000080160 -o xsec.root
   ```

   A large number of graphs (one per simulated process and appropriate totals) will be generated in each case. Each set of plots is saved into its own ROOT *TDirectory* named after the specified initial state.

   The stored graphs can be used for cross section interpolation. For instance, the '*xsec.root*' file generated in this example will contain a 'nu_mu_O16' *TDirectory* (generated by the last command) which will include cross section graphs for all $\nu_\mu+O^{16}$ processes. To extract the $\nu_\mu+O^{16}$ DIS CC cross section graph for hit u valence quarks in a bound proton and evaluate the cross section at energy E, type:

```
root[0] TFile file(''xsec.root'',''read'');
root[1] TDirectory * dir = (TDirectory*) file->Get("nu_mu_O16");
root[2] TGraph * graph = (TGraph*) dir->Get("dis_cc_p_uval");
root[3] cout << graph->Eval(E) << endl;
```

5.3 Simple event generation cases

This section will introduce *gevgen,* a generic GENIE event generation application. This particular application has access to the full suite of GENIE physics models but will only handle relatively simple flux and geometry setups. It doesn't use any of the atmospheric, JPARC, NuMI or other specialized flux drivers included in GENIE and doesn't use ROOT/Geant-4 based detector geometries. A reader interested in the more specialized event generation applications included in GENIE can jump to Chapter 6.

5.3.1 The *gevgen* generic event generation application

Name

gevgen - A generic GENIE event generation application for simple event generation cases. The application handles event generation for neutrinos scattered off a given target (or 'target mix'). It doesn't support event generation over ROOT/Geant4-based detector geometries. It handles mono-energetic flux neutrinos or neutrino fluxes described in simple terms (either via a functional form, a vector file or a ROOT *TH1D* histogram).

Source

The source code for this utility may be found in '*$GENIE/src/stdapp/gEvGen.cxx*'.

```
Synopsis
```

```
$ gevgen [-h] [-r run#] -n nev -p neutrino_pdg -t target_pdg -e energy [-f flux]
   [-w] [-seed random_number_seed] [--cross-section xml_file] [--event-generator-list list_name]
   [--message-thresholds xml_file] [--unphysical-event-mask mask] [--event-record-print-level level]
   [--mc-job-status-refresh-rate rate] [--cache-file root_file]
```

where [] denotes an optional argument.

Description

The following options are available:

- **-h** Prints-out help on *gevgen* syntax and exits.

- **-r** Specifies the MC run number.

- **-n** Specifies the number of events to generate.

- **p** Specifies the neutrino PDG code.

- **-t** Specifies the target PDG code(s).

  The PDG2006 convention is used (10LZZZAAAI). So, for example, O^{16} code = 1000080160, Fe^{56} code = 1000260560. For more details see Appendix D.

Multiple targets (a 'target mix') can be specified as a comma-separated list of PDG codes, each
followed by its corresponding weight fraction in brackets as in:
'`code1[fraction1],code2[fraction2],...`'.
For example, to use a target mix of 95% O16 and 5% H type:
'`-t 1000080160[0.95],1000010010[0.05]`'.

- **-e** Specifies the neutrino energy or energy range.

  For example, specifying '`-e 1.5`' will instruct *gevgen* to generate events at 1.5 GeV.

  If what follows '`-e`' is a comma separated pair of values then gevgen will interpret that as an
  'energy range'. For example, specifying '`-e 0.5,2.3`' will be interpreted as the [0.5 GeV, 2.3 GeV]
  range. If an energy range is specified then *gevgen* expects the '`-f`' option to be set as well so as to
  describe the energy spectrum of flux neutrinos over that range (see below).

- **-f** Specifies the neutrino flux spectrum.

  This generic event generation driver allows to specify the flux in any one of three simple ways:

  - As a 'function'.
    For example, in order to specify a flux that has the $x^2 + 4e^{-x}$ functional form, type:
    '`-f 'x*x+4*exp(-x)'`'

  - As a 'vector file'.
    The file should contain 2 columns corresponding to energy (in GeV), flux (in arbitrary units).
    For example, in order to specify that the flux is described by the vector file '*/data/fluxvec.data*',
    type:
    '`-f /data/fluxvec.data`'

  - As a '1-D histogram (*TH1D*) in a ROOT file'.
    The general syntax is: '`-f /full/path/file.root,object_name`'.
    For example, in order to specify that the flux is described by the 'nue' *TH1D* object in
    '*/data/flux.root*', type:
    '`-f /data/flux.root,nue`'

- **-w** Forces generation of weighted events.

  This option is relevant only if a neutrino flux is specified via the '`-f`' option. In this context
  'weighted' refers to an event generation biasing in selecting an initial state (a flux neutrino and
  target pair at a given neutrino energy). Internal weighting schemes for generating event kinematics
  can still be enabled independently even if '`-w`' is not set. Don't use this option unless you under-
  stand what the internal biasing does and how to analyze the generated sample. The default option
  is to generated unweighted events.

- **--seed** Specifies the random number seed for the current job.

- **--cross-sections** Specifies the name (incl. full path) of an input XML file with pre-computed
  neutrino cross-sections

- **–event-generator-list** Specifies the list of event generators to use in the MC job.

  By default, GENIE is loading a list of of tuned and fully-validated generators which allow comprehensive neutrino interaction modelling the medium-energy range. Valid settings are the XML block names appearing in *$GENIE/config/EventGeneratorListAssembler.xml'*. Please, make sure you read Sec. 5.4 explaining why, almost invariantly, for physics studies you should be using a comprehensive collection of event generators.

- **–message-thresholds** Specifies the GENIE verbosity level.

  The verbosity level is controlled with an XML file allowing users to customize the threshold of each message stream. See *'$GENIE/config/Messenger.xml'* for the XML schema. The *'Messenger.xml'* file contains the default thresholds used by GENIE. The *'Messenger_laconic.xml'* and *'Messenger_rambling.xml'* files define, correspondingly, less and more verbose configurations.

- `--unphysical-event-mask` Specify a 16-bit mask to allow certain types of unphysical events to be written in the output event file.

  By default, all unphysical events are rejected.

- `--event-record-print-level` Allows users to set the level of information shown when the event 94 record is printed in the screen.

  See GHepRecord::Print() for allowed settings.

- `--mc-job-status-refresh-rate` Allows users to customize the refresh rate of the status file.

- `--cache-file` Allows users to specify a ROOT file so that results of calculation cached throughout a MC job can be re-used in subsequent MC jobs.

Examples

1. To generate 20,000 ν_μ (PDG code: 14) scattered off Fe^{56} (PDG code: 1000260560) at an energy of 6.5 GeV, reading pre-computed cross-sections from *'/data/gxsec.xml'*, and using a random number seed of 171872, type:

   ```
   $ gevgen -n 20000 -e 6.5 -p 14 -t 1000260560 -cross-sections /data/gxsec.xml --seed 171872
   ```

2. To generate a similar sample as above, but with the ν_μ energies, between 1 and 4 GeV, selected from a spectrum that has the $x^2 e^{(-x^2+3)/4}$ functional form, type:

   ```
   $ gevgen -n 20000 -e 1,4 -p 14 -t 1000260560 -cross-sections /data/gxsec.xml --seed 171872
   -f 'x*x*exp((-x*x+3)/4)'
   ```

3. To generate a similar sample as above, but with the neutrino flux described via the *'/path/flux.data'* input vector file, type:

   ```
   $ gevgen -n 20000 -e 1,4 -p 14 -t 1000260560 -cross-sections /data/gxsec.xml --seed 171872
   -f /path/flux.data
   ```

4. To generate a similar sample as above, but with the neutrino flux described a ROOT *TH1D* histogram called 'nu_flux' stored in '*/path/file.root*', type:

```
$ gevgen -n 20000 -e 1,4 -p 14 -t 1000260560 -cross-sections /data/gxsec.xml --seed 171872
-f /path/file.root,nu_flux
```

Note that the event generation driver will use only the input histogram bins that fall within the specified (via the '`-e`' option) energy range. In the example shown above, all the neutrino flux bins that do not fall in the 1 to 4 GeV energy range will be neglected. The bins including 1 GeV and 4 GeV will be taken into account. So the actual energy range used is: from the lower edge of the bin containing 1 GeV to the upper edge of the bin containing 4 GeV.

5. To generate a similar sample as above, but, this time, on a target mix that is made of 95% O16 (PDG code: 1000080160) and 5% H (1000010010), type:

```
$ gevgen -n 30000 -e 1,4 -p 14 -cross-sections /data/gxsec.xml --seed 171872
-t 1000080160[0.95],1000010010[0.05] -f /path/file.root,nu_flux
```

Output files   Typically, event generation jobs produce two files:

- During job an ascii status file which contains MC job statistics and the most recent event dump is being updated periodically. The status file is typically named '*genie-mcjob-`<run_number>`.status*' and is located in the current directory. Use –mc-job-status-refresh-rate to adjust the refteshrate of this file.

- The generated events are stored in an output ROOT file, in GENIE's native GHEP format. The event file is typically named '`<prefix>`.`<run_number>`.*ghep.root*' and is located in the current directory. In addition to the generated event tree, the output file contains a couple of ROOT folders, 'gconfig' and 'genv', containing, respectivelly, snapshots of your GENIE configuration and running environment. Chapter 7 describes how to set-up an 'event loop' and analyze the generated event sample.

5.4   Obtaining special samples

5.4.1   Switching reaction modes on/off

The default behaviour of GENIE is to generate 'comprehensive unweighted' event samples. All modelled processes are included and the frequency of process P as well as the occupancy of different parts of the kinematical phase space $\{K^n\}^4$ reflects the value of the differetial cross section $d^n\sigma_P/d\{K^n\}$).

An easy way to obtain special samples is by setting the –**event-generator-list** option available in most GENIE applications. The option controls the list of event generators loaded into a particular GENIE MC job. Valid settings for this option can be found in '$GENIE/*config/EventGeneratorListAssembler.xml*' (the name of each `<param_set>` ... `</param_set>` XML block). New parameter sets can be trivially added by the user.

Please note that this is primarily a GENIE developer option which users should handle with care. In the overwhelming majority of cases, it is only poor understanding of neutrino interaction physics that may lead one thinking that a particular setting is appropriate for generating the special sample one requires.

---

[4]Such as, for example, $\{W, Q^2\}$ or $\{x, y\}$

In general, we do not recommend switching-off generator-level reaction modes. These modes should be treated by the user as internal, generator-specific "labels". No detector measures generator-level reaction modes like $CCQE$ or NC resonance production. Detectors measure final states / topologies like, for example, $\{1\mu^-, 0\pi\}$, $\{1\mu^-, 1\pi^+\}$, $\{0\mu^-, 1\pi^0\}$, $\{1 \text{ track, 1 shower}\}$, $\{1 \text{ } \mu\text{-like ring}\}$ etc depending on granularity, thresholds and PID capabilities. No final state / topology is a proxy for any particular reaction mode (and vice versa). Intranuclear re-scattering in particular causes significant migration between states (see Table 9.4).
Examples:

1. $\{1\mu^-, 0\pi\}$ is mostly ν_μ $CCQE$ but this particular final state can also come about, for example, by ν_μ resonance production followed by intranuclear pion absorption.

2. ν_μ $CCQE$ yields mostly $\{1\mu^-, 0\pi\}$ final states but, occasionaly, can yield $\{1\mu^-, 1\pi\}$ if the recoil nucleon re-interacts.

3. $NC1\pi^0$ final states can be caused by all

    (a) NC elastic followed by nucleon rescattering,

    (b) NC resonance neutrino-production,

    (c) NC non-resonance background,

    (d) low-W NC DIS,

    (e) NC coherent scattering.

   Each such $NC1\pi^0$ source contributes differently to the observed pion momentum distribution.

5.4.2 Event cherry-picking

The *gevpick* cherry-picking utility

Name

gevpick - Reads a list of GENIE event files (GHEP format), 'cherry-picks' events with a given topology and writes them out in a separate file. The output tree contains two additional branches to aid book-keeping by maintaining a 'link' to the source location of each cherry-picked event. For each such event we store a) the name of the original file and b) its original event number.

Source

The source code for this application is in '`$GENIE`/*src/stapp/gEvPick.cxx*'

Synopsis

```
gevpick
   -i input_file_list
   -t cherry_picked_topology
   [-o output_file_name]
```

where [] denotes an optional argument.

Description

The following options are available:

-i Specifies the input file(s).
Wildcards accepted, eg '-i ''/data/genie/pro/gntp.*.ghep.root'''.

-t Specifies the event topology to cherry-pick. The event topology to cherry-pick can be any of the following strings:

- 'all': Selet all events (basically merges all files into one)

- 'numu_cc_1pip': Selects ν_μ CC events with 1 π^+ (and no other pion) in final state.

- 'numu_cc_1pi0': Selects ν_μ CC events with 1 π^0 (and no other pion) in final state.

- 'numu_cc_1pim': Selects ν_μ CC events with 1 π^- (and no other pion) in final state.

- 'numu_nc_1pip': Selects ν_μ NC events with 1 π^+ (and no other pion) in final state.

- 'numu_nc_1pi0': Selects ν_μ NC events with 1 π^0 (and no other pion) in final state.

- 'numu_nc_1pim': Selects ν_μ NC events with 1 π^- (and no other pion) in final state.

- 'numu_cc_hyperon': Selects ν_μ CC events with at least 1 hyperon (Σ^+, Σ^0, Σ^-, Λ^0, Ξ^0, Ξ^-, Ω^-) in the final state.

- 'numubar_cc_hyperon': Selects $\bar{\nu}_\mu$ CC events with at least 1 hyperon in the final state.

- 'cc_hyperon': Selects CC events with at least 1 hyperon in the final state.

-o Specifies the output file name. This in an optional argument. If unset, the output file name will be constructed as: *'gntp.<topology>.ghep.root'*.

Examples

1. Read all events in all '/data/pro2010a/*.ghep.root' files and cherry-pick ν_μ $NC1\pi^0$ events:

   ```
   $ gevpick -i ''/data/pro2010a/*ghep.root'' -t numu_nc_1pi0
   ```

   The cherry-picked event sample gets saved in the '*gntp.numu_ nc_ 1pi0.ghep.root*' file output (default name)

Cherry-picking a new topology

More topologies can be trivially added. Please send your request to the GENIE authors.

Chapter 6

Using a Realistic Flux and Detector Geometry

6.1 Introduction

The main task of GENIE is to simulate the complex physics processes taking place when a neutrino is scattered off a nuclear target. The generator employs advanced, heavily validated models to describe the primary scattering process, the neutrino-induced hadronic multiparticle production and the intra-nuclear hadron transport and re-scattering.

Event generation for realistic experimental setups presents neutrino generators with additional computational challenges. The physics generator is required to handle a large number of nuclear targets (ranging from as light as H^1 to as heavy as Pb^{208}). Moreover, when simulating neutrino interactions in detectors (such as the JPARC and NuMI near detectors) exposed to a non-uniform neutrino flux changing rapidly across the detector volume, it is particularly important to take into account both the detailed detector geometry and the spatial dependencies of the flux. This ensures the proper simulation of backgrounds and avoids introducing highly non-trivial MC artifacts.

The GENIE framework provides many off-the-shelf components for simulating neutrino interactions in realistic experimental setups. New components, encapsulating new neutrino fluxes or detector geometry descriptions, can be trivially added and seamlessly integrated with the GENIE neutrino interaction physics descriptions.

6.2 Components for building customized event generation applications

GENIE provides off-the-shelf components for generating neutrino interactions under the most realistic assumptions integrating the state-of-the-art GENIE neutrino interaction modeling with detailed flux and detector geometry descriptions. GENIE provides an event generation driver class, *GMCJDriver*, that can be used to setup complicated Monte Carlo jobs involving arbitrarily complex, realistic beam flux simulations and detector geometry descriptions. These flux descriptions are typically derived from experiment-specific beam-line simulations while the detector geometry descriptions are typically derived from CAD engineering drawings mapped into the Geant4, ROOT or GDML geometry description languages. Obviously, flux and detector geometry descriptions can take many forms, driven by experiment-specific choices. GENIE standardizes the geometry navigation and flux driver interfaces. These interfaces define a) the

operations that GENIE needs to perform on the geometry and flux descriptions and b) the information GENIE needs to extract from these in order to generate events.

Concrete implementations of these interfaces are loaded into the GENIE event generation drivers, extending GENIE event generation capabilities and allow it to seamlessly integrate new geometry descriptions and beam fluxes.

6.2.1   The flux driver interface

In GENIE every concrete flux driver implements the *GFluxI* interface. The interface defines what neutrino flux information is needed by the event generation drivers and how that information is to be obtained. Each concrete flux driver implements the following methods.

- *const PDGCodeList & GFluxI::FluxParticles (void)*
  Declare the list of flux neutrinos that can be generated. This information is used for initialization purposes, in order to construct a list of all possible initial states in a given event generation run.

- *double GFluxI::MaxEnergy (void)*
  Declare the maximum energy. Again this information is used for initialization purposes, in order to calculate the maximum possible interaction probability in a given event generation run. Since neutrino interaction probabilities are tiny and in order to boost the MC performance, GENIE scales all interaction probabilities in a particular event generation run so that the maximum possible interaction probability is 1. That maximum interaction probability corresponds to the total interaction probability (summed over nuclear targets and process types) for a maximum energy neutrino following a trajectory that maximizes the density-weighted path-lengths for each nuclear target in the geometry. GENIE adjusts the MC run normalization accordingly to account for that internal weighting.

- *bool GFluxI::GenerateNext (void)*
  Generate a flux neutrino and specify its pdg code, its weight (if any), its 4-momentum and 4-position. The 4-position is given in the detector coordinate system (as specified by the input geometry). Each such flux neutrino is propagated towards the detector geometry but is not required to cross any detector volume. GENIE will take that neutrino through the geometry, calculate density-weighted path-lengths for all nuclear targets in the geometry, calculate the corresponding interactions probability off each nuclear target and decide whether that flux neutrino should interact. If it interacts, an appropriate *GEVGDriver* will be invoked to generate the event kinematics.

- *int GFluxI::PdgCode (void)*
  Returns the PDG code of the flux neutrino generated by the most recent *GFluxI::GenerateNext (void)* call.

- *double GFluxI::Weight (void)*
  Returns the weight of the flux neutrino generated by the most recent *GFluxI::GenerateNext (void)* call.

- *const TLorentzVector & GFluxI::Momentum (void)*
  Returns the 4-momentum of the flux neutrino generated by the most recent *GFluxI::GenerateNext (void)* call.

- *const TLorentzVector & GFluxI::Position (void)*
  Returns the position 4-vector of the flux neutrino generated by the most recent *GFluxI::GenerateNext (void)* call.

- *bool GFluxI::End(void)*
  Notify that no more flux neutrinos can be thrown. This flag is typically raised by flux drivers that simply read-in beam-line simulation outputs (as opposed to run the beam simulation code on the fly) so as to notify GENIE that the end of the neutrino flux file has been reached (after, probably, having been recycled N times). The flag allows GENIE to properly terminate the event generation run at the end-of-flux-file irrespective of the accumulated number of events, protons on target, or other metric of exposure.

The above correspond the the common set of operations /information that GENIE expects to be able to perform / extract from all concrete flux drivers. Specialized drivers may define additional information that can be utilized in the experiment-specific event generation drivers. One typical example of this is the flux-specific pass-through information, that is information about the flux neutrino parents such as the parent meson PDG code, its 4-momentum its 4-position at the production and decay points that GENIE simply attaches to each generated event and passes-through so as to be used in later analysis stages.

6.2.2 The geometry navigation driver interface

In GENIE every concrete geometry driver implements the *GeomAnalyzerI* interface. The interface specifies what information about the input geometry is relevant to the event generation and how that information is to be obtained. Each concrete geometry driver implements methods to

- *const PDGCodeList & GeomAnalyzerI::ListOfTargetNuclei (void)*
  Declare the list of target nuclei that can be found in the geometry. This information is used for initialization purposes, in order to construct a list of all possible initial states in a given event generation run.

- *const PathLengthList & GeomAnalyzerI::ComputeMaxPathLengths (void)*
  Compute the maximum density-weighted path-lengths for each nuclear target in the geometry. Again, this is information used for initialization purposes. The computed 'worst-case' trajectory is used to calculate the maximum possible interaction probability in a particular event generation run which is being used internally to normalize all computed interaction probabilities.

- *const PathLengthList & GeomAnalyzerI::ComputePathLengths (const TLorentzVector & x, const TLorentzVector & p)*
  Compute density-weighted path-lengths for all nuclear targets, for a 'ray' of a given 4-momentum and starting 4-position. This allows GENIE to calculate probabilities for each flux neutrino to be scattered off every nuclear target along its path through the detector geometry.

- *const TVector3 & GeomAnalyzerI::GenerateVertex (const TLorentzVector & x, const TLorentzVector & p, int tgtpdg)*
  Generate a vertex along a 'ray' of a given 4-momentum and starting 4-position on a volume containing a given nuclear target. This allows GENIE to place a neutrino interaction vertex within the detector geometry once an interaction of a flux neutrino off a selected nuclear target has been generated.

6.2.3 Setting-up GENIE MC jobs using fluxes and geometries

```
{
 ...

 // get flux driver
```

```
GFluxI * flux_driver = new ... ;

// get geometry driver
GeomAnalyzerI * geom_driver = new ... ;

// create the GENIE monte carlo job driver
GMCJDriver* mcjob_driver = new GMCJDriver;
mcjob_driver->UseFluxDriver(flux_driver);
mcjob_driver->UseGeomAnalyzer(geom_driver);
mcjob_driver->Configure();

...
}
```

6.3 Built-in flux drivers

GENIE currently contains a host of concrete flux drivers that allow GENIE to be used in many realistic, experiment-specific situations:

- *GJPARCNuFlux*: An interface to the JPARC neutrino beam simulation [125] used at SK, nd280, and INGRID.

- *GNuMIFlux*: An interface to the NuMI beam simulations [126] used at MINOS, NOvA, MINERvA and ArgoNEUT.

- *GBartolAtmoFlux*: A driver for the BGLRS atmospheric flux by G. Barr, T.K. Gaisser, P. Lipari, S. Robbins and T. Stanev [127].

- *GFlukaAtmo3DFlux*: A driver for the FLUKA 3-D atmospheric neutrino flux by A. Ferrari, P. Sala, G. Battistoni and T. Montaruli [128].

- *GAstroFlux*: A driver for astrophysical neutrino fluxes. Handles both diffuse fluxes and point sources. (Under development.)

- *GCylindTH1Flux*: A generic flux driver, describing a cylindrical neutrino flux of arbitrary 3-D direction and radius. The radial dependence of the neutrino flux is configurable (default: uniform per unit area). The flux driver may be used for describing a number of different neutrino species whose (relatively normalised) energy spectra are specified as ROOT 1-D histograms. This driver is being used whenever an energy spectrum is an adequate description of the neutrino flux.

- *GSimpleNtpFlux*: An interface for a simple ntuple-based flux that can preserve energy-position correlations without the format being tied to any particular experimental setup (though individual files are very much so).

- *GMonoEnergeticFlux*: A trivial flux driver throwing mono-energetic flux neutrinos along the +z direction. More that one neutrino species can be included, each with its own weight. The driver is being used in simulating a single initial state at a fixed energy mainly for probing, comparing and validating neutrino interaction models.

New concrete flux drivers (describing the neutrino flux from other beam-lines) can be easily developed and they can be effortlessly and seamlessly integrated with the GENIE event generation framework.

6.3.1 JPARC neutrino flux driver specifics

GJPARCNuFlux provides an interface to the JPARC neutrino beam simulations (JNUBEAM [125]) used at SK, nd280, and INGRID.

[expand]

6.3.2 NuMI neutrino flux driver specific

GNuMIFlux provides an interface to the NuMI beam simulations used at MINOS, NOvA, MINERvA and ArgoNeut. This interface can handle all three of the formats used so far in simulating the NuMI beamline: Geant3-based gnumi, g4numi and flugg. It can also handle the FNAL booster flux when that is formatted into one of the standard ntuple layouts. These beam simulation files record hadron decays and sufficient information to calculate new weights and energies for different positions relative to the beam orgin.

The driver generates a flux to cover a user specified detector "window" after undergoing a coordinate transformation from the beam system to that of a particular detector. The detector specific windows and transformations are encapsulated in the '`$GENIE`/*src/FluxDriver/GNuMINtuple/GNuMIFlux.xml*' file. Users can extend what is available by modifying this file and putting a copy in a location specified by `GXMLPATH`="*/path/to/location*". Additional "*param_set*" sections allow new configurations and these can be based on modifications of select parameters of an existing "param_set" entry. Extensive documentation of the setable parameters can be found in the XML file itself.

When the *GNuMIFlux* is invoked it must be configured by passing the method *GNuMIFlux::LoadBeamSimData()* an input filename string and a config name. The input file name may include wildcards on the file name but not the directory path. The config name selects a "param_set" from the XML file. The *GNuMIFlux* object will by default declare the list of flux neutrinos that it finds in the input files; this can be overridden to have it ignore entries for flavors the user is not interested in.

6.3.3 FLUKA and BGLRS atmospheric flux driver specifics

GFlukaAtmo3DFlux and *GBartolAtmoFlux* provide, respectivelly, an interface to the FLUKA-3D (A. Ferrari, P. Sala, G. Battistoni and T. Montaruli [128]) and BGLRS (G. Barr, T.K. Gaisser, P. Lipari, S. Robbins and T. Stanev [127]) atmospheric neutrino flux simulations.

Both classes inherit all their functionallity from the *GAtmoFlux* base class from which they derive. *GFlukaAtmo3DFlux* and *GBartolAtmoFlux* merely define the appropriate binning for each flux simulation: The FLUKA flux is given in 40 bins of $cos\theta$, where θ is the zenith angle, from -1 to 1 (bin width = 0.05) and 61 equally log-spaced energy bins (20 bins per decade) with a minimum energy of 100 MeV. The BGLRS flux is given in 20 bins of $cos\theta$ from -1 to 1 (bin width = 0.1) and 30 equally log-spaced energy bins (10 bins per decade) with a minimum energy of 10 GeV. For more details please visit the FLUKA[1] and BGLRS[2] flux web sites.

Both the FLUKA and BGLRS flux simulations are distributed as ascii data files for various locations and solar activity levels. There is one data file per atmospheric neutrino flavor. You can specify the input files for each neutrino flavor using the '*void GAtmoFlux::SetFluxFile(int neutrino_code, string filename)*' method. The expected input code is the PDG one and the input filename should include the full path to the file. You can specify flux files for an arbitrary set of flux neutrino flavors. Neutrino flavors for which you have not specified a flux file will be omitted from the atmospheric neutrino event generation job. Once you have specified flux files for all neutrino flavors you wish to include you need to call the '*void GAtmoFlux::LoadFluxData()*' method.

---

[1] *http://pcbat1.mi.infn.it/~battist/neutrino.html*
[2] *http://www-pnp.physics.ox.ac.uk/~barr/fluxfiles/*

By default, the flux neutrino position and momentum 4-vectors are generated in the Topocentric Horizontal Coordinate System (+z: Points towards the local zenith / +x: On same plane as local meridian, pointing south . +y: As needed to make a right-handed coordinate system / Origin: Input geometry centre). A rotation to a user-defined topocentric coordinate system can be enabled by invoking the '*void GAtmoFlux::SetUserCoordSystem (TRotation &)*' method. For a given direction, determined by the zenith angle θ and azimuth angle ϕ, the flux generation surface is a circular area, with radius R_T, which is tangent to a sphere of radius R_L centered at the coordinate system origin. These two radii can be set using the '*void GAtmoFlux::SetRadii (double RL, double RT)*' method. Obviously, R_T and R_L must be appropriately chosen so that the flux generation surface is always outside the input geometry volume and so that, for every given direction, the 'shadow' of the generation surface covers the entire geometry (see Fig. 6.1).

Energy cuts can be specified using the '*void GAtmoFlux::ForceMinEnergy(double Emin)*' and '*void GAtmoFlux::ForceMaxEnergy(double Emax)*' methods. Finally, the atmospheric neutrino flux drivers can generate both weighted and unweighted flux neutrinos (with the unweighted-mode used as default). In the weighted-mode the energy is generated logarithmically and the zenith angle cosine is generated uniformly and, after a neutrino species has been selected, the event weight is set to be the flux histogram bin content for the given neutrino species and for the given energy and zenith angle cosine. Using a weighted-mode may be . The user choice can be registered using the '*void GAtmoFlux::GenerateWeighted(bool option)*' method.

6.3.4   Generic histogram-based flux specifics

The *GCylindTH1Flux* is generic flux driver, describing a cylindrical neutrino flux of arbitrary 3-D direction and radius. The direction of the flux rays (in 3-D) can be specified using the '*void GCylindTH1Flux::SetNuDirection(co TVector3 &)*' method while the radius of the cylinder is specified using '*void GCylindTH1Flux::SetTransverseRadius(double* The flux generation surface is a circular area defined by the intersection of the flux cylinder with a plane which is perpendicular to the flux ray direction. To fully specify the flux neutrino generation surface the user needs to specify the centre of that circular area (see 'beam spot' in Fig. 6.2) using the '*void GCylindTH1Flux::SetBeamSpot(const TVector3 & spot)*' method. Obviously the 'beam spot' should be placed upstream of the detector volume.

The radial dependence of the neutrino flux can be configured using the '*void GCylindTH1Flux::SetRadialDependence(str rdep)*' method. The expected input is the functional form of the R_T-dependence (with R_T denoted as x). By default, the driver is initialized with *SetRadialDependence*("x"), so flux neutrinos are generated uniformly per unit area.

The flux driver may be used for describing a number of different neutrino species whose (relatively normalised) energy spectra are specified as ROOT 1-D histograms (*TH1D*). To input the energy distribution of each neutrino species use *GCylindTH1Flux::AddEnergySpectrum (int nu_pdgc, TH1D * spectrum)*'.

Obviously, when using *GCylindTH1Flux*, no energy-position correlation is present. This may or may-not be a good approximation depending on the specifics of your experimental setup and analysis. If energy-position correlation is important (and known) then consider using the *GSimpleNtpFlux* flux driver. This correlation is also built-in in the specialized JPARC, NuMI and atmospheric flux drivers, described in this chapter, which you should be utilizing if relevant to your application.

6.3.5   Generic ntuple-based flux specifics

The *GSimpleNtpFlux* flux driver provides an interface for a simple ntuple-based flux that can preserve energy-position correlations without the format being tied to any particular experimental setup (though individual files are very much so). The basic entry consists a TTree branch with the elements:

- **px**, **py**, **pz**, **E**: 4-momentum components.

Figure 6.1: Construction of flux generation surface for the atmospheric neutrino flux drivers. For a given direction, determined by the zenith angle θ and azimuth angle ϕ, the flux generation surface is a circular area, with radius R_T, which is tangent to a sphere of radius R_L centered at the coordinate system origin. R_T and R_L must be appropriately chosen so that the flux generation surface is always outside the input geometry volumes and so that, for every given direction, the 'shadow' of the generation surface covers the entire geometry. See text for more details.

Figure 6.2: Geometrical setup for the *GCylindTH1Flux* flux drivers. The diriver allows you to set the beam direction (in 3-D), and the radius R_T of the flux generation surface. To fully specify the position of the flux generation surface in 3-D the driver allows you to set the 'beam spot' 3-vector. Additionally the R_T-dependence can be configured. Multiple neutrino species can be generated in the flux surface, each one with its own energy distribution and relative normalization. See text for more details.

- **vtxx**, **vtxy**, **vtxz**: Neutrino ray origin info (detector coordinates).

- **dist**: Distance from hadron decay to ray origin.

- **wgt**: Neutrino weight (generally 1.0).

- **metakey**: Reference back to meta-data. The "metadata" branch has an entry per file recording general info such as the list of neutrino flavors found in the entries, the number of protons-on-target represented by the file (in the case of accelerator based fluxes), the maximum energy, the minimum and maximum weights, the flux window and a vector of strings for a record of the list of files used to generate the GSimpleNtp file.

Additional information can be stored in conjunction with the individual entries either by supplemental classes for branches (ala the optional "numi" branch), or via the flexible "aux" branch which allows arbitrary vectors of integers and doubles (name info in the metadata allows for keeping track of what elements represent under the assumption that all entries have identical additions).

When the *GSimpleNtpFlux* is invoked it needs to be configured by passing the method *GSimpleNtpFlux::LoadBeamSimData()* an input filename string (and a config name that is ignored). The input file name may include wildcards on the file name but not the directory path. Multiple gsimple flux files can also be combined into a larger file with the use of the ROOT *hadd* utility.

The *GSimpleNtpFlux* is in use by some NuMI experiments as a means of factorizing the computation necessary for the evaluation of the *GNuMIFlux* from the actual event generation. Unlike the *GNuMIFlux* files, entries can not be positioned for new locations (which would change the entry's weight and energy) but they also don't require the computational burden of doing so. They are meant to be simple and fast.

6.4 Built-in geometry navigation drivers

GENIE currently contains two concrete geometry drivers which are sufficient for all event generation cases encountered so far:

- *ROOTGeomAnalyzer*: A geometry driver handling detector geometries specified using ROOT. As detector geometries specified using Geant4 or *GDML* can be converted into ROOT geometries, this driver is being used in all cases where a detailed detector geometry is being passed on to GENIE.

- *PointGeomAnalyzer*: A trivial geometry corresponding to a single nuclear target or a target mix (a set of nuclear targets each with its corresponding weight fraction) at a fixed position. This driver is being used to simulate only given initial states as a means for probing the neutrino interaction physics modeling or in experimental situations where the detector is being illuminated by a spatially uniform neutrino beam and where the generated interaction vertices do not have any spatial dependence and can be generated uniformly within volumes of given nuclear targets.

6.4.1 ROOT geometry navigation driver specifics

The *ROOTGeomAnalyzer* works based on a probing a detailed ROOT geometry to evaluate the mass distribution seen along individual neutrino 'rays' (a starting position in space relative to the detector geometry and a direction). Each ray is stepped through the geometry from one volume boundary to the next; each transition to a new volume instantiates a new *PathSegment*, which are collected into a *PathSegmentList* for the ray and which also includes information about the ray itself.

A *PathSegment* object records the information about the distance from the ray origin to the entrance of the volume, the step length in the volume, information about the volume (e.g. medium, material), positions at the boundaries, and (optionally) the ROOT volume path string (the volume hierarchy in the

geometry). A neutrino ray from the flux is passed through the geometry only once. From the information recorded in the *PathSegmentList* the *GMCJDriver* can be given the density-weighted path-lengths for all the nuclear targets. If the *GMCJDriver* decides that an interaction occurred this *PathSegmentList* is then used to properly select a vertex position based on the chosen nuclear target.

Defining units

The *ROOTGeomAnalyzer* can be configured to account for differences in length and density units between the GENIE defaults and what is assumed in the ROOT geometry.

Defining a fiducial volume

For ROOT geometries that include representations of material that isn't of interest, such as the rock surrounding a cavern hall, the *ROOTGeomAnalyzer::SetTopVolName()* method allows one to consider only the material within that volume. In more sophisticated circumstances there might not be a volume in the ROOT geometry representing the region in which one wants to restrict vertices. More refined limits can be placed by configuring the *ROOTGeomAnalyzer* with a concrete implementation of the *GeomVolSelectorI* interface.

A concrete implementation of the *GeomVolSelectorI* interface must provide a method for "trimming" individual PathSegment items based on information in the segment. Trimming futher restricts the region of the step within the volume; ranges delineate sub-steps and by this means segments within a volume can be reduced, split or eliminated. The implementation must also provide methods that gets called at the start of *PathSegmentList* trimming and upon completion (these can be dummies). If the implementation needs to know the ROOT geometry volume path hierarchy then it must signal that.

Two useful examples of *GeomVolSelectorI* are provided: *GeomVolSelectorBasic* and *GeomVolSelectorFiducial*. The basic class is configurable to select or reject whole segments based on the volume name, medium, material and (optionally) volume path string. The fiducial class builds on that base and add the potential for defining a elementary shape (sphere, cylinder, box, convex polyhedron) in space that is used to trim segments. This shape does not have to correspond to anything represented in the ROOT geometry. The cut can be to require considering only material within the shape or only that outside of the shape.

6.5   Built-in specialized event generation applications

This section discusses specialized GENIE-based event generation applications included in GENIE distributions. These applications integrate the GENIE event generation modules with very specific neutrino flux and detector geometry descriptions.

- *gevgen_t2k*: A GENIE-based event generation application for T2K. It integrates GENIE with the JPARC neutrino beam-line simulation (JNUBEAM) and the geometry descriptions of nd280, 2km, INGRID and Super-K detectors. (See subsection 6.5.1.)

- *gevgen_fnal*: A GENIE-based event generation application for the Fermilab experiments (including DUNE, and the experiments in the NuMI and Booster beam-lines). It integrates GENIE with the Fermilab neutrino beam-line simulations and the geometry descriptions of MINOS, NOvA, MINERvA, SBND, MicroBooNE, DUNE and other detectors. (See subsection 6.5.2.)

- *gevgen_atmo*: A GENIE-based atmospherc neutrino event generation application. It integrates the GENIE with any of the FLUKA 3-D [128] or BGLRS [127] atmospheric neutrino flux simulations. Events can be generated for either a simple target mix or a detailed ROOT-based detector geometry (See subsection 6.5.3.)

Although the above applications have common options, each of the following subsections is entirely self-contained. Please go directly to the subsection describing the application you are interested at.

6.5.1 Event generation application for the T2K experiment

Name

gevgen_t2k – A GENIE-based event generation application for T2K. It integrates GENIE with the JPARC neutrino beam-line simulation (JNUBEAM) and the detector geometry descriptions of nd280, 2km, INGRID and Super-K.

Source and build options

The source code for this application is in '`$GENIE`/*src/support/t2k/EvGen/gT2KEvGen.cxx*'.
To enable it add '`--enable-t2k`' during the GENIE build configuration step.

Synopsis

```
$ gevgen_t2k
   -f flux [-p POT_normalization_of_flux_file] [-R]
   -g geometry [-t geometry_top_volume_name]
   [-m max_path_lengths_xml_file]
   [-P] [pre_gen_flux_prob_name]
   [-S] [output_pre_gen_flux_prob_name]
   [-L geometry_length_units] [-D geometry_density_units]
   <-n num_of_events, -c num_of_flux_ntuple_cycles, -e, -E exposure_in_POTs>
   [-o output_event_file_prefix] [-r run#]
   [-seed random_number_seed] [--cross-section xml_file] [--event-generator-list list_name]
   [--message-thresholds xml_file] [--unphysical-event-mask mask] [--event-record-print-level level]
   [--mc-job-status-refresh-rate rate] [--cache-file root_file]
   [-h]
```

where [] denotes an optional argument and <> denotes a group of arguments out of which only one can be set.

Description

The following options are available:

-f Specifies the input neutrino flux. This option can be used to specify any of:

- A JNUBEAM beam simulation output file and the detector location. The general sytax is:
  '`-f /path/flux_file.root,detector_loc(,neutrino_list)`'

  For more information on the flux ntuples see the JNUBEAM documentation. The ntuple has to be in ROOT format and can be generated from the distributed HBOOK ntuples using ROOT's *h2root* utility. The detector location can be any of 'sk' or the near detector positions 'nd1',...,'nd6' simulated by JNUBEAM. The optional *neutrino_list* is a comma separated list neutrino PDG codes. It specifies which neutrino flux species to to considered in the event generation job. If no such neutrino list is specified then, by default, GENIE will consider all neutrino species in the input flux ntuple. When a JNUBEAM ntuple is used for describing the neutrino flux, GENIE is able to

calculate the POT exposure for the generated event sample and any one of the exposure setting methods ('-e', '-E', '-c', '-n', see below) can be used.

All JNUBEAM information on the flux neutrino parent (parent PDG code, parent 4-position and 4-momentum at the production and decay points etc) is stored in a 'flux' branch of the output event tree and is associated with the corresponding generated neutrino event.

Example 1:
To use the Super-K JNUBEAM flux ntuple from the '*/t2k/flux/jnubeam001.root*' file, type:
`'-f /t2k/flux/jnubeam001.root,sk'`

Example 2:
To use the 2km flux ntuple [near detector position 'nd1' in the jnubeam flux simulation] from the '*/t2k/flux/jnubeam001.root*' file, type:
`'-f /t2k/flux/jnubeam001.root,nd1'`

Example 3:
To use the nd280 flux ntuple [near detector position 'nd5' in the jnubeam flux simulation] from the '*/t2k/flux/jnubeam001.root*' file, type:
`'-f /t2k/flux/jnubeam001.root,nd5'`

Example 4:
To the same as above but using only the ν_e and $\bar{\nu}_e$ flux ntuple entries, type:
`'-f /t2k/flux/jnubeam001.root,nd5,12,-12'`

- A set of flux histograms stored in a ROOT file. The general syntax is:
  `'-f /path/file.root,neutrino_code[histo],...'`

  where *neutrino_ code* is a standard neutrino PDG code[3] and *histo* is the corresponding ROOT histogram name.

  Multiple flux histograms can be specified for different flux neutrino species (see the example given below). The relative flux normalization for all neutrino species should be represented correctly at the input histogram normalization. The absolute flux normalization is not relevant: Unlike when using JNUBEAM ntuples to describe the flux, no POT calculations are performed when plain histogram-based flux descriptions are employed. One can only control the MC run exposure via the number of generated events ('-n', see below). In this case the POT normalization of the generated sample is calculated externally.

  Since there is no directional information in histogram-based descriptions of the flux, the generated neutrino vertex is always set to (0,0,0,0). Then it is the detector MC responsibility to rotate the interaction vectors and plant the vertex [4] Obviously no flux pass-through branch is written out in the neutrino event tree since no such information is associated with flux neutrinos selected from plain histograms.

  Example:
  To use the histogram 'h1' (representing the ν_μ flux) and the histogram 'h2' (representing the ν_e

---

[3] ν_e: 12, ν_μ: 14, ν_τ: 16, $\bar{\nu}_e$: -12, $\bar{\nu}_\mu$: -14 and $\bar{\nu}_\tau$: -16

[4] This option is used only for the Super-K simulation where vertices are distributed uniformly in volume by the detector MC (SKDETSIM). For event generation at the more complex near detectors a JNUBEAM ntuple-based flux description should be used so as the interaction vertex is properly planted within the input geometry by GENIE.

flux) from the *'/data/flux.root'* file, type:
'`-f /data/flux.root,14[h1],12[h2]`'

-p Specifies to POT normalization of the input flux file. This is an optional argument. By default, it is set to the standard JNUBEAM flux ntuple normalization of 1E+21 POT/detector (for the near detectors) or 1E+21 POT/cm2 (for the far detector). The input normalization factor will be used to interpret the flux weights and calculate the POT normalization for the generated neutrino event sample. The option is irrelevant if a simple, histogram-based description of the neutrino flux is used (see -f option)

-R Instructs the flux driver to start looping over the flux ntuples with a random offset. This is an optional argument. It may be necessary on some occassions to avoid biases when using very large input flux files.

-g Specifies the input detector geometry. This option can be used to specify any of:

- A ROOT file containing a ROOT/Geant4-based geometry description (*TGeoManager*).
  This is the standard option for generating events in the nd280, 2km and INGRID detectors.

  Example:
  To use the ROOT detector geometry description stored in the '*/data/geo/nd280.root*' file, type:
  '`-g /data/geo/nd280.root`'

  By default the entire input geometry will be used. Use the '`-t`' option to allow event generation only on specific geometry volumes.

- A mix of target materials, each with its corresponding weight.
  This is the standard option for generating events in the Super-K detector where the beam profile is uniform and distributing the event vertices uniformly in the detector volume is sufficient. The target mix is specified as a comma-separated list of nuclear PDG codes (in the PDG2006 convention: 10LZZZAAAI) followed by their corresponding weight fractions in brackets, as in:
  '`-t code1[fraction1],code2[fraction2],...`'

  Example 1:
  To use a target mix of 88.79% (weight fraction) O^{16} and 11.21% H (i.e. 'water') type:
  '`-g 1000080160[0.8879],1000010010[0.1121]`'

  Example 2:
  To use a target which is 100% C^{12}, type:
  '`-g 1000060120`'

-t Specifies the input top volume for event generation. This is an optional argument. By default, it is set to be the 'master volume' of the input geometry resulting in neutrino events being generated over the entire geometry volume. If the '`-t`' option is set, event generation will be confined in the specified detector volume. The option can be used to simulate events at specific sub-detectors.
Example:
To generate events in the P0D only, type:
'`-t P0D`'

You can use the '`-t`' option to switch generation on/off at multiple volumes
Example:

`'-t +Vol1-Vol2+Vol3-Vol4'`, or

`'-t ''+Vol1 -Vol2 +Vol3 -Vol4'''`

This instructs the GENIE geometry navigation code to switch on volumes 'Vol1' and 'Vo3' and switch off volumes 'Vol2' and 'Vol4'. If the very first character is a '+', GENIE will neglect all volumes except the ones explicitly turned on. Vice versa, if the very first character is a '-', GENIE will keep all volumes except the ones explicitly turned off.

-m Specifies an XML file with the maximum density-weighted path-lengths for each nuclear target in the input geometry. This is an optional argument. If the option is not set (and also if the options **-P** and **-S** are not set) GENIE will scan the input geometry to determine the maximum density-weighted path-lengths for all nuclear targets.then, at the MC job initialization, GENIE will scan the input geometry to determine the maximum density-weighted path-lengths for all nuclear targets. The computed information is used for calculating the neutrino interaction probability scale to be used in the MC job (the tiny neutrino interaction probabilities get normalized to a probability scale which is defined as the maximum possible total interaction probability, corresponding to a maximum energy neutrino in a worst-case trajectory maximizing its density-weighted path-length, summed up over all possible nuclear targets). That probability scale is also used to calculate the absolute, POT normalization of a generated event sample from the POT normalization of the input JNUBEAM flux ntuple.

Feeding-in pre-computed maximum density-weighted path-lengths results in faster MC job initialization and ensures that the same interaction probability scale is used across all MC jobs in a physics production job (the geometry is scanned by a MC ray-tracing method and the calculated safe maximum density-weighted path-lengths may differ between MC jobs).

The maximum density-weighted path-lengths for a Geant4/ROOT-based detector geometry can be pre-computed using GENIE's *gmxpl* utility.

-P Specifies a ROOT file with the pre-calculated interaction probability for each flux neutrino in the input flux file, for the top volume and the input geometry. This is an optional argument. This option is intended to replace the maximum density weighted path-lengths option **-m**. This option is new in v2.6.2. The pre-calculated interaction probability method is specific to the flux input (JNUBEAM flux ntuples), and so has been optimised much more than the maximum density weighted path-lengths method. The interaction probability for each flux neutrino is pre-calcuated before any events are generated. The maximum interaction probability is now exact (maximally efficient) and means that the interaction probability does not need to be recalculated, until we have decided there has been an interaction. It is especially fast for complicated geometries. This means that this method is up to 300 times faster than the **-m** option. The **-P** option can be used in one of two ways. The first is to pre-calculate the interaction probabilites in a separate job (using the **-S** option of *gevgen_ t2k*, see below). This is especially good for larger flux files with $> \mathcal{O}(100000)$ entries, as the time to pre-calculate interaction probabilities becomes comparable to the event generation time. For small flux files, the amount of bookkeeping when using pre-calculated interaction probabties means that generating interaction probabilites at the start of each job is faster (you should run **-P** with no arguments). Note that if none of **-P**, **-S** and **-m** are set, then GENIE will scan the input geometry to determine the maximum density-weighted path-lengths for all nuclear targets, during initalization of the MC job.

-S Specifies a location to save a ROOT file with the calculated interaction probability for each flux neutrino in the input flux file, for the top volume and the input geometry. This is an optional argument. It is used to create pre-calculated interaction probabilities for input into the **-P** option. You should make sure to use exactly the input flux file, input geometry, top volume name, neutrino flavours, etc... arguments in your *gevgen_t2k* submission line for your pre-calculation of interaction probabilities (**-S**), and your use of the pre-calculated interaction probabilites (**-P**). The default output name is [flux_file_name].[top_volume_name].flxprobs.root. This can be overridden by providing an argument to the **-S** option. Note that running *gevgen_ t2k* with this option will not generate any events; the only

output will be a ROOT file contaning the pre-calculated interaction probabilites.

-L Specifies the input geometry length units. This is an optional argument. By default, that option is set to 'mm', the length units used for the nd280 detector geometry description. Possible options include: 'm', 'cm', 'mm', ...

-D Specifies the input geometry density units. This is an optional argument. By default, that option is set to 'clhep_def_density_unit', the density unit used for the nd280 detector geometry description ($= \sim 1.6\text{E-}19$ x g/cm3 !). Possible options include: 'kg_m3', 'g_cm3', 'clhep_def_density_unit',...

-c Specifies how many times to cycle a JNUBEAM flux ntuple. This option provides a way to set the MC job exposure in terms of complete JNUBEAM flux ntuple cycles. On each cycle, every flux neutrino in the ntuple will be thrown towards the detector geometry.

-e Specifies how many POTs to generate. If this option is set, *gevgen_t2k* will work out how many times it has to cycle through the input flux ntuple in order to accumulate the requested statistics. The program will stop at the earliest complete flux ntuple cycle after accumulating the required statistics. The generated statistics will slightly overshoot the requested number but the calculated exposure (which is also stored at the output file) will be exact. This option is only available with JNUBEAM ntuple-based flux descriptions.

-E Specifies how many POTs to generate. This option is similar to '`-e`' but the program will stop immediately after the requested POT has been accumulated, without waiting for the current loop over the flux ntuple entries to be completed. The generated POT overshoot (with respect to the requested POT) will be negligible, but the POT calculation within a flux ntuple cycle is only approximate. This reflects the details of the JNUBEAM beam-line simulation. This option is only available with JNUBEAM ntuple-based flux descriptions.

-n Specifies how many events to generate. Note that out of the 4 possible ways of setting the exposure ('`-c`', '`-e`', '`-E`', '`-n`') this is the only available one if a plain histogram-based flux description is used.

-o Sets the prefix of the output event file. This is an optional argument. It allows you to override the output event file prefix. In GENIE, the output filename is built as:

    prefix.run_number.event_tree_format.file_format where, in *gevgen_t2k*, by default, `prefix`: 'gntp' and `event_tree_format`: 'ghep' and `file_format`: 'root'.

-r Specifies the MC run number. This is an optional argument. By default a run number of '1000' is used.

–seed Specifies the random number seed for the current job.

–cross-sections Specifies the name (incl. full path) of an input XML file with pre-computed neutrino cross-sections

–event-generator-list Specifies the list of event generators to use in the MC job. By default, GENIE is loading a list of of tuned and fully-validated generators which allow comprehensive neutrino interaction modelling the medium-energy range. Valid settings are the XML block names appearing in *$*`GENIE`*/config/EventGeneratorListAssembler.xml*'. Please, make sure you read Sec. 5.4 explaining why, almost invariantly, for physics studies you should be using a comprehensive collection of event generators.

–message-thresholds Specifies the GENIE verbosity level. The verbosity level is controlled with an XML file allowing users to customize the threshold of each message stream. The XML schema can be

seen in '*$GENIE/config/Messenger.xml*'. The '*Messenger.xml*' file contains the default thresholds used by GENIE. The '*Messenger_laconic.xml*' and '*Messenger_rambling.xml*' files define, correspondingly, less and more verbose configurations.

–unphysical-event-mask Specify a 16-bit mask to allow certain types of unphysical events to be written in the output event file. By default, all unphysical events are rejected.

–event-record-print-level Allows users to set the level of information shown when the event 94 record is printed in the screen. See GHepRecord::Print() for allowed settings.

–mc-job-status-refresh-rate Allows users to customize the refresh rate of the status file.

–cache-file Allows users to specify a ROOT file so that results of calculation cached throughout a MC job can be re-used in subsequent MC jobs.

-h Prints out the *gevgen_t2k* syntax and exits.

Examples

1. Generate events (run '1001') using the jnubeam flux ntuple in '*/data/t2k/flux/07a/jnb001.root*' and picking up the flux entries for the detector location 'nd5' (which corresponds to the 'nd280m' location). The job will load the nd280 geometry from '*/data/t2k/geom/nd280.root*' and interpret it assuming the length unit is 'mm' and the density unit is the default CLHEP one. The job will stop on the first complete flux ntuple cycle after generating 5E+17 POT. Read pre-computed cross-section splines from '*/data/t2k/xsec/xsec.xml*'. Use seed number 1982199 and, also, use the default GENIE verbosity level.

   ```
   $ gevgen_t2k -r 1001 -f /data/t2k/flux/07a/jnb001.root,nd5
   -g /data/t2k/geom/nd280.root -L mm -D clhep_def_density_unit
   --cross-sections /data/t2k/xsec/xsec.xml -e 5E+17 --seed 1982199
   ```

2. As before, but now the job will stop after 100 flux ntuple cycles, whatever POT and number of events that may correspond to.

   ```
   $ gevgen_t2k -r 1001 -f /data/t2k/flux/07a/jnb001.root,nd5
   -g /data/t2k/geom/nd280.root -L mm -D clhep_def_density_unit
   --cross-sections /data/t2k/xsec/xsec.xml -c 100 --seed 1982199
   ```

3. As before, but now the job will stop after generating 100000 events, whatever POT and number of flux ntuple cycles that may correspond to.

   ```
   $ gevgen_t2k -r 1001 -f /data/t2k/flux/07a/jnb001.root,nd5
   -g /data/t2k/geom/nd280.root -L mm -D clhep_def_density_unit
   --cross-sections /data/t2k/xsec/xsec.xml -n 100000 --seed 1982199
   ```

4. As before, but first pre-calculate interaction probilites, and then use them to generate events.

   ```
   $ gevgen_t2k -r 1001 -f /data/t2k/flux/07a/jnb001.root,nd5
   -g /data/t2k/geom/nd280.root -L mm -D clhep_def_density_unit
   --cross-sections /data/t2k/xsec/xsec.xml -n 100000 --seed 1982199
   ```

```
-S jnb001.nd280.global.flxprobs.root

$ gevgen_t2k -r 1001 -f /data/t2k/flux/07a/jnb001.root,nd5
-g /data/t2k/geom/nd280.root -L mm -D clhep_def_density_unit
--cross-sections /data/t2k/xsec/xsec.xml -n 100000 --seed 1982199
-P jnb001.nd280.global.flxprobs.root
```

5. Generate events (run '1001') using the jnubeam flux ntuple in '*/data/t2k/flux/07a/jnb001.root*' and picking up the flux entries for the Super-K detector location. This time, the job will not use any detailed detector geometry description but just (95% O^{16} + 5% H) target-mix. The job will stop after generating 50000 events. As before, read pre-computed cross-section splines from '*/data/t2k/xsec/xsec.xml*'. This time use production-mode verbosity level (set all message thresholds to 'warning').

```
$ gevgen_t2k -r 1001 -f /data/t2k/flux/07a/jnb001.root,sk
-g 1000080160[0.95],1000010010[0.05] -n 50000 --seed 1982199
--cross-sections /data/t2k/xsec/xsec.xml --message-thresholds Messenger_laconic.xml
```

6. As before, but now the flux is not described using a JNUBEAM ntuple but a set of 1-D histograms from the '*/data/flx.root*' file: The histogram named 'h1' will be used for the ν_e flux, 'h2' will will be used for the $\bar{\nu}_e$ flux, and 'h3' for the ν_μ flux.

```
$ gevgen_t2k -r 1001 -f /data/flx.root,12[h1],-12[h2],14[h3]
-g 1000080160[0.95],1000010010[0.05] -n 50000 --seed 1982199
--cross-sections /data/t2k/xsec/xsec.xml --message-thresholds Messenger_laconic.xml
```

6.5.2 Event generation application for Fermilab neutrino experiments

Name

gevgen_fnal – A GENIE-based event generation application for Fermilab neutrino experiments. It integrates the GENIE with the Fermilab neutrino beam-line simulations and the geometry descriptions of DUNE, MINOS, NOvA, MINERvA, ArgoNEUT, MicroBooNE, SBND and other experiments.

Source and build options

The source code for this application is in '`$GENIE`*/src/support/fnal/EvGen/gFNALExptEvGen.cxx*'. To enable it add '`--enable-numi`' during the GENIE build configuration step.

Synopsis

```
$ gevgen_fnal
    -f flux
    -g geometry [-t top_volume_name_at_geom]
    [-F fiducial_cut_string] [-m max_path_lengths_xml_file]
    [-L geometry_length_units] [-D geometry_density_units] [-z z_min]
    <-n number_of_events, -e exposure_in_POTs>
    [-o output_event_file_prefix] [-r run#] [-d debug_flags]
    [-seed random_number_seed] [--cross-section xml_file] [--event-generator-list list_name]
    [--message-thresholds xml_file] [--unphysical-event-mask mask] [--event-record-print-level level]
    [--mc-job-status-refresh-rate rate] [--cache-file root_file]
```

```
[-h]
```

where [] denotes an optional argument and $<>$ denotes a group of arguments out of which only one can be set.

Description

The following options are available:

-f Specifies the input neutrino flux. This option can be used to specify any of:

- A gNuMI beam simulation output file and the detector location. The general sytax is:
  `'-f /path/flux_file.root,detector_loc(,neutrino_list)'`

  For more information of the flux ntuples see the gNuMI documentation. The ntuple has to be in ROOT format and can be generated from the distributed HBOOK ntuples using ROOT's *h2root* utility. See GNuMIFlux.xml for all supported detector locations. The optional *neutrino_list* is a comma separated list neutrino PDG codes. It specifies which neutrino flux species to to considered in the event generation job. If no such neutrino list is specified then, by default, GENIE will consider all neutrino species in the input flux ntuple. When a gNuMI ntuple is used for describing the neutrino flux, GENIE is able to calculate the POT exposure for the generated event sample and any one of the exposure setting methods ('-e', '-n', see below) can be used. All gNuMI information on the flux neutrino parent (parent PDG code, parent 4-position and 4-momentum at the production and decay points etc) is stored in a 'flux' branch of the output event tree and is associated with the corresponding generated neutrino event.

  Example:
  To use the gNuMI flux ntuple flux.root at MINOS near detector location '*/data/flux.root*' file, type:
  `'-f /data/flux.root,MINOS-NearDet'`

- A set of flux histograms stored in a ROOT file. The general syntax is:
  `'-f /path/file.root,neutrino_code[histo],...'`

  where *neutrino_code* is a standard neutrino PDG code[5] and *histo* is the corresponding ROOT histogram name. Multiple flux histograms can be specified for different flux neutrino species (see the example given below). The relative flux normalization for all neutrino species should be represented correctly at the input histogram normalization. The absolute flux normalization is not relevan heret: Unlike when using gNuMI ntuples to describe the flux, no POT calculations are performed when histogram-based flux descriptions are employed. One can only control the MC run exposure via the number of generated events ('-n', see below). In this case the POT normalization of the generated sample is calculated externally.

  Since there is no directional information in plain histogram-based descriptions of the flux, the generated neutrino vertex is always set to (0,0,0,0). Then it is the detector MC responsibility to rotate the interaction vectors and plant the vertex [6] Obviously no flux pass-through branch is

---

[5] ν_e: 12, ν_μ: 14, ν_τ: 16, $\bar{\nu}_e$: -12, $\bar{\nu}_\mu$: -14 and $\bar{\nu}_\tau$: -16
[6] This option is used only for the Super-K simulation where vertices are distributed uniformly in volume by the detector MC (SKDETSIM). For event generation at the more complex near detectors a JNUBEAM ntuple-based flux description should be used so as the interaction vertex is properly planted within the input geometry by GENIE.

written out in the neutrino event tree since no such information is associated with flux neutrinos selected from plain histograms.

Example:
To use the histogram 'h1' (representing the ν_μ flux) and the histogram 'h2' (representing the ν_e flux) from the '/data/flux.root' file, type:
'-f /data/flux.root,14[h1],12[h2]'

-g Specifies the input detector geometry. This option can be used to specify any of:

- A ROOT file containing a ROOT/Geant4-based geometry description (*TGeoManager*).
  Example:
  To use the ROOT detector geometry description stored in the '/data/geo/nova.root' file, type:
  '-g /data/geo/nova.root'

  By default the entire input geometry will be used. Use the '-t' option to allow event generation only on specific geometry volumes.

- A mix of target materials, each with its corresponding weight.
  This is the standard option for generating events in the Super-K detector where the beam profile is uniform and distributing the event vertices uniformly in the detector volume is sufficient. The target mix is specified as a comma-separated list of nuclear PDG codes (in the PDG2006 convention: 10LZZZAAAI) followed by their corresponding weight fractions in brackets, as in:
  '-t code1[fraction1],code2[fraction2],...'

  Example 1:
  To use a target mix of 88.79% (weight fraction) O^{16} and 11.21% H (i.e. 'water') type:
  '-g 1000080160[0.8879],1000010010[0.1121]'

  Example 2:
  To use a target which is 100% C^{12}, type:
  '-g 1000060120'

-t Specifies the input top volume for event generation. This is an optional argument. By default, it is set to be the 'master volume' of the input geometry resulting in neutrino events being generated over the entire geometry volume. If the '-t' option is set, event generation will be confined in the specified detector volume. The option can be used to simulate events at specific sub-detectors.
Example:
To generate events in the P0D only, type:
'-t P0D'

You can use the '-t' option to switch generation on/off at multiple volumes
Example:
'-t +Vol1-Vol2+Vol3-Vol4', or
'-t "+Vol1 -Vol2 +Vol3 -Vol4"'
This instructs the GENIE geometry navigation code to switch on volumes 'Vol1' and 'Vo3' and switch off volumes 'Vol2' and 'Vol4'. If the very first character is a '+', GENIE will neglect all volumes except the ones explicitly turned on. Vice versa, if the very first character is a '-', GENIE will keep all volumes except the ones explicitly turned off.

-m Specifies an XML file with the maximum density-weighted path-lengths for each nuclear target in the input geometry. This is an optional argument. If the option is not set then, at the MC job initialization, GENIE will scan the input geometry to determine the maximum density-weighted path-lengths for all nuclear targets. The computed information is used for calculating the neutrino interaction probability scale to be used in the MC job (the tiny neutrino interaction probabilities get normalized to a probability scale which is defined as the maximum possible total interaction probability, corresponding to a maximum energy neutrino in a worst-case trajectory maximizing its density-weighted path-length, summed up over all possible nuclear targets). That probability scale is also used to calculate the absolute, POT normalization of a generated event sample from the POT normalization of the input flux ntuple.

Feeding-in pre-computed maximum density-weighted path-lengths results in faster MC job initialization and ensures that the same interaction probability scale is used across all MC jobs in a physics production job (the geometry is scanned by a MC ray-tracing method and the calculated safe maximum density-weighted path-lengths may differ between MC jobs).

The maximum density-weighted path-lengths for a Geant4/ROOT-based detector geometry can be pre-computed using GENIE's *gmxpl* utility.

-L Specifies the input geometry length units. This is an optional argument. By default it is set to 'mm'. Possible options include: 'm', 'cm', 'mm', ...

-D Specifies the input geometry density units. This is an optional argument. By default it is set to 'g_cm3'. Possible options include: 'kg_m3', 'g_cm3', 'clhep_def_density_unit' (= ∼1.6E-19 x g/cm3 !),...

-F Applies a fiducial cut. This is an optional argument. Applies a fiducial cut (for now hard-coded). Only used with ROOT-based detector geometry descriptions. If the input string starts with "-" then reverses sense (ie. anti-fiducial).

-S Number of rays to use to scan geometry for max path length. This is an optional argument. Number of rays to use to scan geometry for max path length. Only used with ROOT-based detector geometry descriptions (and the gNuMI ntuple-based flux description). If '+N' : Scan the geometry using N rays generated using flux neutrino directions pulled from the input gNuMI flux ntuple. If '-N' : Scan the geometry using N rays x N points on each face of a bounding box. Each ray has a uniformly distributed random inward direction.

-z Z from which to start flux ray in user-world coordinates. This is an optional argument. If left unset then flux originates on the flux window [No longer attempts to determine z from geometry, generally got this wrong].

-o Sets the prefix of the output event file. This is an optional argument. It allows you to override the output event file prefix. In GENIE, the output filename is built as:

`prefix.run_number.event_tree_format.file_format` where, in *gevgen_ numi*, by default, `prefix`: 'gntp' and `event_tree_format`: 'ghep' and `file_format`: 'root'.

-r Specifies the MC run number. This is an optional argument. By default a run number of '0' is used.

–seed Specifies the random number seed for the current job.

–cross-sections Specifies the name (incl. full path) of an input XML file with pre-computed neu-

trino cross-sections

–event-generator-list Specifies the list of event generators to use in the MC job. By default, GENIE is loading a list of of tuned and fully-validated generators which allow comprehensive neutrino interaction modelling the medium-energy range. Valid settings are the XML block names appearing in *$*GENIE*/config/EventGeneratorListAssembler.xml*'. Please, make sure you read Sec. 5.4 explaining why, almost invariantly, for physics studies you should be using a comprehensive collection of event generators.

–message-thresholds Specifies the GENIE verbosity level. The verbosity level is controlled with an XML file allowing users to customize the threshold of each message stream. The XML schema can be seen in '*$*GENIE*/config/Messenger.xml*'. The '*Messenger.xml*' file contains the default thresholds used by GENIE. The '*Messenger_laconic.xml*' and '*Messenger_rambling.xml*' files define, correspondingly, less and more verbose configurations.

–unphysical-event-mask Specify a 16-bit mask to allow certain types of unphysical events to be written in the output event file. By default, all unphysical events are rejected.

–event-record-print-level Allows users to set the level of information shown when the event 94 record is printed in the screen. See GHepRecord::Print() for allowed settings.

–mc-job-status-refresh-rate Allows users to customize the refresh rate of the status file.

–cache-file Allows users to specify a ROOT file so that results of calculation cached throughout a MC job can be re-used in subsequent MC jobs.

-h Prints out the *gevgen_fnal* syntax and exits.

Examples

6.5.3 Event generation application for atmospheric neutrinos

Name

gevgen_atmo – A GENIE-based atmospheric neutrino event generation application. It integrates GENIE with any of the FLUKA 3-D [128] or BGLRS [127] atmospheric neutrino flux simulations. Events can be generated for either a simple target mix or a detailed ROOT-based detector geometry.

Source and build options

The source code for this application is in '`$GENIE`*/src/support/atmo/EvGen/gAtmoEvGen.cxx*'.
To enable it add '`--enable-atmo`' during the GENIE build configuration step.

Synopsis

```
$ gevgen_atmo
   -f flux -g geometry
   [-R rotation_from_topocentric_hz_frame]
   [-t geometry_top_volume_name] [-m max_path_lengths_xml_file]
   [-L geometry_length_units] [-D geometry_density_units]
   <-n number_of_events, -e exposure_in_terms_of_kton_x_yrs>
```

```
[-E energy_range] [-o output_event_file_prefix] [-r run#]
[-seed random_number_seed] [--cross-section xml_file] [--event-generator-list list_name]
[--message-thresholds xml_file] [--unphysical-event-mask mask] [--event-record-print-level level]
[--mc-job-status-refresh-rate rate] [--cache-file root_file]
[-h]
```

where [] denotes an optional argument and <> denotes a group of arguments out of which only one can be set.

Description

The following options are available:

-f Specifies the input neutrino flux. This option can be used to specify the input flux simulation data files. The general syntax is: '`-f simulation:/path/file[neutrino_code],...`'. The '`simulation`' part of the option can be either 'FLUKA' or 'BGLRS', depending on the origin of your input data files. GENIE will use the input tag to use the appropriate input file format and to bin the input data according to the choices of the FLUKA and BGLRS flux simulation authors. See Section 6.3.3 for more details. The '`/path/file.data[neutrino_code]`' part of the option can be repeated multiple times (separated by commas), once for each flux neutrino species you wish to consider.
Example 1:
   '`-f FLUKA:/data/sdave_numu07.dat[14],/data/sdave_nue07.dat[12]`'
   This option will instruct GENIE to use the '*/data/sdave_ numu07.dat*' FLUKA flux simulation file for ν_μ and the '*/data/sdave_ nue07.dat*' file for ν_e. No other flux species will be considered in this MC job.
Example 2:
   '`-f BGLRS:/data/flux10_271003_z.kam_nue[12]`'
   This option will instruct GENIE to use the '*/data/flux10_ 271003_ z.kam_ nue*' BGLRS flux simulation file for ν_e. No other flux species will be considered in this MC job.

-g Specifies the input detector geometry. This option can be used to specify any of:

- A ROOT file containing a ROOT/Geant4-based geometry description (*TGeoManager*).

   Example:
   To use the ROOT detector geometry description stored in the '*nd280-geom.root*' file, type:
   '`-g /some/path/nd280-geom.root`'

   By default the entire input geometry will be used. Use the '`-t`' option to allow event generation only on specific geometry volumes.

- A mix of target materials, each with its corresponding weight.
   This option should only be used when the beam and/or detector are sufficiently uniform. The target mix is specified as a comma-separated list of nuclear PDG codes (in the PDG2006 convention: 10LZZZAAAI) followed by their corresponding weight fractions in brackets, as in:
   '`-t code1[fraction1],code2[fraction2],...`'

   Example 1:
   To use a target mix of 88.79% (weight fraction) O^{16} and 11.21% H (i.e. 'water') type:

```
'-g 1000080160[0.8879],1000010010[0.1121]'
```

Example 2:
To use a target which is 100% C^{12}, type:
```
'-g 1000060120'
```

-R Specifies a rotation from the default topocentric horizontal coordinate system to a user-defined frame. The rotation is specified by the 3 Euler angles φ, ϑ, ψ. The Euler angles are used for creting a ROOT TRotation object which gets applied to the flux neutrino position and momentum 4-vectors before that flux neutrino is fired towards the detector. The user has the option to select between the X and Y conventions. By default, the X-convention is used. Additionally, the user can request GENIE to invert the rotation matrix before applying it to the flux neutrino vectors. Please note the following extract from the ROOT TRotation documentation: "Euler angles usually define the rotation of the new coordinate system with respect to the original system, however, the TRotation class specifies the rotation of the object in the original system (an active rotation). To recover the usual Euler rotations (ie. rotate the system not the object), you must take the inverse of the rotation."

The Euler angles are input as a comma separated list. The general syntax for specifying the rotation is: '`-R convention:phi,theta,psi`' where '`convention`' is either X (for X-convention), Y (for Y-convention), X^-1 or Y^-1 (as previously, but using the inverse rotation matrix instead).

Example 1:
To set the Euler angles φ=3.14, ϑ=1.28, ψ=1.0 using the X-convention, type: '`-R 3.14,1.28,1.0`', or '`-R X:3.14,1.28,1.0`'.

Example 2:
To set the Euler angles φ=3.14, ϑ=1.28, ψ=1.0 using the Y-convention, type: '`-R Y:3.14,1.28,1.0`'.

Example 3:
To set the Euler angles φ=3.14, ϑ=1.28, ψ=1.0 using the Y-convention, and then use the inverse rotation matrix, type: '`-R Y^-1:3.14,1.28,1.0`'.

-t Specifies the input top volume for event generation. This is an optional argument. By default, it is set to be the 'master volume' of the input geometry resulting in neutrino events being generated over the entire geometry volume. If the '`-t`' option is set, event generation will be confined in the specified detector volume. The option can be used to simulate events at specific sub-detectors.

-m Specifies an XML file with the maximum density-weighted path-lengths for each nuclear target in the input geometry. This is an optional argument. If the option is not set then, at the MC job initialization, GENIE will scan the input geometry to determine the maximum density-weighted path-lengths for all nuclear targets. The computed information is used for calculating the neutrino interaction probability scale to be used in the MC job (the tiny neutrino interaction probabilities get normalized to a probability scale which is defined as the maximum possible total interaction probability, corresponding to a maximum energy neutrino in a worst-case trajectory maximizing its density-weighted path-length, summed up over all possible nuclear targets). That probability scale is also used to calculate the absolute normalization of generated sample in terms of kton*yrs.

Feeding-in pre-computed maximum density-weighted path-lengths results in faster MC job initialization and ensures that the same interaction probability scale is used across all MC jobs in a physics production job (the geometry is scanned by a MC ray-tracing method and the calculated safe maximum density-weighted path-lengths may differ between MC jobs).

The maximum density-weighted path-lengths for a Geant4/ROOT-based detector geometry can be pre-computed using GENIE's *gmxpl* utility.

-L Specifies the input geometry length units. This is an optional argument. By default it is set to 'm'. Possible options include: 'm', 'cm', 'mm', ...

-D Specifies the input geometry density units. This is an optional argument. By default it is set to 'kg_m3'. Possible options include: 'kg_m3', 'g_cm3', 'clhep_def_density_unit' (= ∼1.6E-19 x g/cm3 !),...

-n Specifies how many events to generate.

-e Specifies the requested exposure in terms of kton*yrs.

Not implemented yet.

-E Specifies an energy range in GeV. This is an optional argument. Must be a set of comma-separated values. By default GENIE will generate atmospheric neutrinos between 0.5 and 50 GeV.
Example: To generate events between 1 and 100 GeV type: `'-E 1,100'`

-o Sets the prefix of the output event file. This is an optional argument. It allows you to override the output event file prefix. In GENIE, the output filename is built as:
`prefix.run_number.event_tree_format.file_format` where, in *gevgen_ atmo*, by default, `prefix`: 'gntp' and `event_tree_format`: 'ghep' and `file_format`: 'root'.

-r Specifies the MC run number. This is an optional argument. By default a run number of '100000000' is used.

–seed Specifies the random number seed for the current job.

–cross-sections Specifies the name (incl. full path) of an input XML file with pre-computed neutrino cross-sections

–event-generator-list Specifies the list of event generators to use in the MC job. By default, GENIE is loading a list of of tuned and fully-validated generators which allow comprehensive neutrino interaction modelling the medium-energy range. Valid settings are the XML block names appearing in *$GENIE/config/EventGeneratorListAssembler.xml'*. Please, make sure you read Sec. 5.4 explaining why, almost invariantly, for physics studies you should be using a comprehensive collection of event generators.

–message-thresholds Specifies the GENIE verbosity level. The verbosity level is controlled with an XML file allowing users to customize the threshold of each message stream. The XML schema can be seen in '*$GENIE/config/Messenger.xml'*. The '*Messenger.xml'* file contains the default thresholds used by GENIE. The '*Messenger_ laconic.xml'* and '*Messenger_ rambling.xml'* files define, correspondingly, less and more verbose configurations.

–unphysical-event-mask Specify a 16-bit mask to allow certain types of unphysical events to be written in the output event file. By default, all unphysical events are rejected.

–event-record-print-level Allows users to set the level of information shown when the event 94 record is printed in the screen. See GHepRecord::Print() for allowed settings.

–mc-job-status-refresh-rate Allows users to customize the refresh rate of the status file.

–cache-file Allows users to specify a ROOT file so that results of calculation cached throughout a MC job can be re-used in subsequent MC jobs.

-h Prints out the *gevgen_atmo* syntax and exits.

Examples

1. Generate 100k events (run number '100000013') using the FLUKA 3-D flux simulation output files '*/data/flux/atmo/sdave_numu07.dat*', for ν_μ, and '*/data/flux/atmo/sdave_nue07.dat*', for ν_e. Do not consider any other flux neutrino species. Generate events for water (weight fraction: 88.79% O^{16} and 11.21% H) and only in the 1-15 GeV energy range. Read pre-computed cross-section splines from '*/data/xsec/xsec.xml*'. Use seed number 87218 and production mode verbosity level (all message thresholds set to warning).

```
$ gevgen_atmo -n 100000 -r 100000013 -e 1,15
-f FLUKA:/data/flux/atmo/sdave_numu07.dat[14],/data/flux/atmo/sdave_nue07.dat[12]
-g 1000080160[0.8879],1000010010[0.1121]
--cross-sections /data/xsec/xsec.xml
--seed 87218 --message-thresholds Messenger_laconic.xml
```

2. Like above but, instead of generating events in water, generate events using the detailed ROOT-based detector geometry description in file '*/data/geo/HyperKamionande.root*'. Let GENIE know that the geometry file expresses length in 'mm' and densities in 'gr/cm^3'. Don't generate events over the the entire volume but only within the volume named 'InnerDetector'.

```
$ gevgen_atmo -n 100000 -r 100000013 -e 1,15
-f FLUKA:/data/flux/atmo/sdave_numu07.dat[14],/data/flux/atmo/sdave_nue07.dat[12]
-g /data/geo/HyperKamiokande.root -t InnerDetector -L mm -D g_cm3
--cross-sections /data/xsec/xsec.xml
--seed 87218 --message-thresholds Messenger_laconic.xml
```

Chapter 7

Analyzing Output Event Samples

7.1   Introduction

[to be added in future revision]

7.2   The GHEP event structure

Events generated by GENIE are stored in a custom, *STDHEP*-like, event record called *GHEP*. Each *GHEP* event record, an instance of the *GHepRecord* class, is a ROOT *TClonesArray* container of *GHep-Particle* objects representing individual particles. Other than being a container for the generated particles, the event record holds additional information with event-, rather than particle-, scope such as the cross sections for the selected event and the differential cross section for the selected event kinematics, the event weight, a series of customizable event flags and an interaction summary. Additionally, the event record includes a host of methods for querying / setting event properties including many methods that allow querying for specific particles within the event (such as for example methods to return the target nucleus, the final state primary lepton or a list of all the stable descendants of any intermediate particle).

The event record features a 'spontaneous re-arrangement' feature which maintains the compactness of the daughter lists at any given time. This is necessary for the correct interpretation of the stored particle associations as the daughter indices correspond to a contiguous range. The particle mother and daughter indices for all particles in the event record are automatically updated as a result of any such spontaneous particle rearrangement.

The *GHEP* structure is highly compatible with the event structures used in most HEP generators. That allows us to call other generators (such as for example PYTHIA / JETSET) as part of an event generation chain and convert / append their output into the current *GHEP* event. Additionally the *GHEP* events can be converted to many other formats for facilitating the GENIE interface with experiment-specific offline software systems and cross-generator comparisons.

7.2.1   GHEP information with event-wide scope

[to be added in future revision]

7.2.2   Interaction summary

All particles generated by GENIE for each simulated event are stored into a *GHEP* record which represents the most complete description of a generated event. Certain external heavy-weight applications such

as specialized event-reweighing schemes or realistic, experiment-level MC simulation chains using the generator as the physics front-end require that detailed particle-level information.

However, many of the actual physics models employed by the generator, such as cross section, form factor or structure function models, require a much simpler event description. An event description based on simple summary information, typically including a description of the initial state, the process type and the scattering kinematics, is sufficient for driving the algorithmic objects implementing these physics models. In the interest of decoupling the physics models from event generation and the particle-level event description, GENIE uses an *Interaction* object to store summary event information. Whenever possible, algorithmic objects implementing physics models accept a single *Interaction* object as their sole source of information about an event. That enables the use of these models both within the event generation framework but also within a host of external applications such as model validation frameworks, event re-weighting tools and user physics analysis code.

An *Interaction* objects is an aggregate, hierarchical structure, containing many specialised objects holding information for the initial state (*InitialState* object), the event kinematics (*Kinematics* object), the process type (*ProcessInfo* object) and potential additional information for tagging exclusive channels (*XclsTag* object).

Users can easily instantiate *Interaction* objects and use them to drive physics models. Creating this aggregate hierarchical structure is streamlined using the 'named constructor' C++ idiom. For example, in order to define a 5 GeV QELCC $\nu_\mu + neutron$ interaction, where the neutron is bound in a Fe^{56} nucleus, (ν_μ PDG code = 14, *neutron* PDG code = 2112, Fe^{56} PDG code = 1000260560), one needs to instantiate an *Interaction* object as in:

`Interaction * qelcc = Interactions::QELCC(1000260560, 2112, 14, 5.0);`

That interaction definition can be used as is to drive a QELCC cross section algorithm.

The *Interaction* objects can serialize themselves as a unique string codes which, within the GENIE framework, play the role of the 'reaction codes' of the old procedural systems. These string codes are used extensively whenever there is a need to map information to or from interaction types (as for example, mapping interaction types to pre-computed cross section splines, or mapping interaction types to specialized event generation code)

Each generated event has an *Interaction* summary object attached to it and written out in the output event trees. Despite the implications of having a certain amount of redundancy (in the sense that this summary information can be recreated entirely from the information at the *GHEP* record) this strategy presents many advantages during both event generation and analysis of generated events.

7.2.3   GHEP particles

The basic output unit of the event generation process is a 'particle'. This is an overloaded term used to describe both particles and nuclei appearing in the initial, intermediate or final state, or generator-specific pseudo-particles used for facilitating book-keeping of the generator actions.

Each such 'particle' generated by GENIE is an instance of the *GHepParticle* class. These objects contain information with particle-scope such as its particle and status codes, its pdg mass, charge and name, the indices of its mother and daughter particles marking possible associations with other particles in the same event, its 4-momentum and 4-position (in the target nucleus coordinate system), its polarization vector, and other properties. The GHepParticle class includes methods for setting and querying these properties.

GENIE has adopted the standard PDG particle codes. For ions it has adopted a PDG extension, using the 10-digit code 10LZZZAAAI where AAA is the total baryon number, ZZZ is the total charge, L is the number of strange quarks and I is the isomer number (I=0 corresponds to the ground state). GENIE-specific pseudo-particles have PDG code $>=$ 2000000000 and can convey important information about the underlying physics model. Pseudo-particles generated by other specialized MCs that may be

called by GENIE (such as PYTHIA) are allowed to retain the codes specified in that MC.

GENIE marks each particle with a status code. This signifies the position of an particle in an event and helps navigating within the event record. Most generated particles are typically marked as one of the following:

- *'initial state'* typically the first two particles of the event record corresponding to the incoming neutrino and the nuclear target.

- *'nucleon target'*, corresponding to the hit nucleon (if any) within the nuclear target.

- *'intermediate state'*, typically referring to the remnant nucleus, fragmentation intermediates such as quarks, diquarks, some intermediate pseudo-particles etc.

- *'hadron in the nucleus'*, referring to a particle of the primary hadronic system, that is the particles emerging from the primary interaction vertex before their possible re-interactions during their intranuclear hadron transport.

- *'decayed state'*, such as for example unstable particles that have been decayed.

- *'stable final state'* for the relatively long-lived particles emerging from the nuclear targets

All particles generated by GENIE during the simulation of a single neutrino interaction are stored in a dynamic container representing an 'event'.

7.2.4 Mother / daughter particle associations

[to be added in future revision]

Idx	Name	ISt	PDG	Mom	Kids	E	px	py	...
0	nu_mu	0	14	-1	4 4	...	...	...	...
1	Fe56	0	1000260560	-1	2 3	...	...	...	...
2	neutron	11	2112	1	5 7	...	...	...	...
3	Fe55	2	1000260550	1	10 10	...	...	...	...
4	mu-	1	13	0	-1 -1	...	...	...	...
5	HadrSyst	12	2000000001	2	-1 -1	...	...	...	...
6	proton	14	211	2	-1 -1	...	...	...	...
7	pi0	14	111	2	8 9	...	...	...	...
8	proton	1	22	7	-1 -1	...	...	...	...
9	pi-	1	-211	7	-1 -1	...	...	...	...
10	**HadrBlob**	15	2000000002	3	-1 -1	...	...	...	...

Table 7.1: []

Idx	Name	ISt	PDG	Mom	Kids	E	px	py	...
0	nu_mu	0	14	-1	5 5	...	...	...	...
1	Fe56	0	1000260560	-1	2 3	...	...	...	...
2	proton	11	2212	1	4 4	...	...	...	...
3	Mn55	2	1000250550	1	12 12	...	...	...	...
4	Delta++	3	2224	2	6 7	...	...	...	...
5	mu-	1	13	0	-1 -1	...	...	...	...
6	proton	14	2112	4	8 8	...	...	...	...
7	pi+	14	211	4	11 11	...	...	...	...
8	proton	3	2212	6	9 10	...	...	...	...
9	proton	1	2212	8	-1 -1	...	...	...	...
10	proton	1	2212	8	-1 -1	...	...	...	...
11	pi+	1	211	7	-1 -1	...	...	...	...
12	HadrBlob	15	2000000002	3	-1 -1	...	...	...	...

Table 7.2:

Idx	Name	ISt	PDG	Mom	Kids	E	px	py	...
0	nu_mu	0	14	-1	4 4	...	...	...	...
1	Fe56	0	1000260560	-1	2 3	...	...	...	...
2	neutron	11	2112	1	5 5	...	...	...	...
3	Fe55	2	1000260550	1	22 22	...	...	...	...
4	mu	1	13	0	-1 -1	...	...	...	...
5	HadrSyst	12	2000000001	2	6 7	...	...	...	...
6	u	12	2	5	8 8	...	...	...	...
7	ud_1	12	2103	5	-1 -1	...	...	...	...
8	string	12	92	6	9 11	...	...	...	...
9	pi0	14	111	8	14 14	...	...	...	...
10	proton	14	2212	8	15 15	...	...	...	...
11	omega	12	223	8	12 13	...	...	...	...
12	pi-	14	-211	11	16 16	...	...	...	...
13	pi+	14	211	11	21 21	...	...	...	...
14	pi0	1	111	9	-1 -1	...	...	...	...
15	proton	1	2212	10	-1 -1	...	...	...	...
16	pi-	3	-211	12	17 20	...	...	...	...
17	neutron	1	2112	16	-1 -1	...	...	...	...
18	neutron	1	2112	16	-1 -1	...	...	...	...
19	proton	1	2212	16	-1 -1	...	...	...	...
20	proton	1	2212	16	-1 -1	...	...	...	...
21	pi+	1	211	13	-1 -1	...	...	...	...
22	HadrBlob	15	2000000002	3	-1 -1	...	...	...	...

Table 7.3:

7.3   Printing-out events

7.3.1   The *gevdump* utility

Name

gevdump - A GENIE utility printing-out GENIE GHEP event records.

Source

The source code for this utility may be found in '*$*GENIE*/src/stdapp/*gEvDump.cxx'.

Synopsis

```
$ gevdump -f filename [-n n1[,n2]]
```

where [] denotes an optional argument.

Description

The following options are available:

- **-f** Specifies a GENIE GHEP event file.

- **-n** Specifies an event number or a range of event numbers. This is an optional argument. By default all events will be printed-out.

Notes

You can fine-tune the amount of information that gets printed-out by tweaking the 'GHEPPRINTLEVEL' environmental variable (see Appendix **??**)

Examples

1. To print-out all events from '*/data/sample.ghep.root*', type:

   ```
   $ gevdump -f /data/sample.ghep.root
   ```

2. To print-out the first 500 events from '*/data/sample.ghep.root*', type:

   ```
   $ gevdump -f /data/sample.ghep.root -n 0,499
   ```

3. To print-out event 178 from '*/data/sample.ghep.root*', type:

   ```
   $ gevdump -f /data/sample.ghep.root -n 178
   ```

7.4   Event loop skeleton program

An'event loop' skeleton is given below. You may insert your event analysis code where is indicated below. Please look at the next section for information on how to extract information from the 'event' object.

```
{
  ...
  // Open the GHEP/ROOT file
  string filename = /data/sample.ghep.root;
  TFile file(filename.c_str(), READ);

  // Get the tree header & print it
  NtpMCTreeHeader * header =
    dynamic_cast<NtpMCTreeHeader*> (file.Get("header"));
  LOG(test, pINFO) << *header;

  // Get the GENIE GHEP tree and set its branch address
  TTree * tree = dynamic_cast<TTree*> (file.Get(gtree));
  NtpMCEventRecord * mcrec = 0;
  tree->SetBranchAddress(gmrec, &mcrec);

  // Event loop
  for(Long64_t i=0; i<tree->GetEntries(); i++){
    tree->GetEntry(i);

    // print-out the event
    EventRecord & event = *(mcrec->event);
    LOG(test, pINFO) << event;

    // put your event analysis code here
    ...
    ...


    mcrec->Clear();
  }
  ...
}
```

An'event loop' skeleton can be found in *'$GENIE/src/test/testEveltLoop.cxx'*. Copy this file and use it as a starting point for your event loop.

7.5 Extracting event information

The readers are instructed to spend some time browsing the GENIE doxygen documentation, especially the classes defined in the *Interaction* and *GHEP* packages, and familiarize themselves with the public methods. Some examples on how to extract information from an 'event' objects are given below.

Examples

1. Extract the interaction summary for the given event and check whether it is a QEL CC event (excluding QEL CC charm production):

```
{
 ...

 Interaction * in = event.Summary();

 const ProcessInfo & proc = in->ProcInfo();
 const XclsTag & xclsv    = in->ExclTag();

 bool qelcc = proc.IsQuasiElastic() && proc.IsWeakCC();
 bool charm = xclsv.IsCharm();

 if (qelcc && !charm)
 {
    ...
 }
 ...
}
```

2. Get the energy threshold for the given event:

```
{
 ...

 Interaction * in = event.Summary();

 double Ethr = in->PhaseSpace().Threshold();
 ...
}
```

3. Get the momentum transfer Q^2 and hadronic invariant mass W, as generated during kinematical selection, for RES CC event:

```
{
 ...
 const ProcessInfo & proc  = in->ProcInfo();
 const Kinematics  & kine = in->Kine();
```

```
    bool selected = true;

    if (proc.IsResonant() && proc.IsWeakCC())
    {
        double Q2s = kine.Q2(selected);
        double Ws  = kine.W (selected);
    }
    ...
}
```

4. Calculate the momentum transfer Q^2, the energy transfer ν, the Bjorken x variable, the inelasticity y and the hadronic invariant mass W directly from the event record:

```
{
  ...
  // get the neutrino, f/s primary lepton and hit
  // nucleon event record entries
  //
  GHepParticle * neu = event.Probe();
  GHepParticle * fsl = event.FinalStatePrimaryLepton();
  GHepParticle * nuc = event.HitNucleon();

  // the hit nucleon may not be defined
  // (eg. for coherent, or ve- events)
  //
  if(!nuc) return;

  // get their corresponding 4-momenta (@ LAB)
  //
  const TLorentzVector & k1 = *(neu->P4());
  const TLorentzVector & k2 = *(fsl->P4());
  const TLorentzVector & p1 = *(nuc->P4());

  // calculate the kinematic variables
  // (eg see Part.Phys. booklet, page 191)
  //
  double M = kNucleonMass;

  TLorentzVector q = k1 - k2;

  double Q2 = -1 * q.M2();
  double v  = q.Energy();
  double x  = Q2 / (2*M*v);
  double y  = v / k1.Energy();
  double W2 = M*M - 2*M*v - Q2;
  double W  = TMath::Sqrt(TMath::Max(0., W2));


  ...
```

```
     }
```

5. Loop over particles and count the number of final state pions:

```
   {
    ...
    int npi = 0;

    TObjArrayIter iter(&event);
    GHepParticle * p = 0;

    // loop over event particles
    for((p = dynamic_cast<GHepParticle *>(iter.Next()))) {

       int pdgc   = p->Pdg();
       int status = p->Status();

       if(status != kIStStableFinalState) continue;

       bool is_pi = (pdgc == kPdgPiP ||
                     pdgc == kPdgPi0 ||
                     pdgc == kPdgPiM);

       if(is_pi) npi++;
    }

    ...
   }
```

6. Get the corresponding NEUT reaction code for a GENIE event:

```
   {
     ...

     int neut_code = utils::ghep::NeutReactionCode(&event);
     ...
   }
```

7.6 Event tree conversions

You do not need to convert the GENIE *GHEP* trees in order to analyze the generated samples or pass them on to a detector-level Monte Carlo. But you can do so if:

- you need to pass GENIE events to legacy systems using already standardized formats,

- you want to be able to read-in GENIE events without loading any GENIE libraries (eg bare-ROOT, or XML formats),

- you want to extract just summary information and write it out in simpler ntuples.

7.6.1 The *gntpc* ntuple conversion utility

Name

gntpc – A utility tp converts the native GENIE *GHEP* event file to a host of plain text, XML or bare-ROOT formats.

Source

The source code can be found in '*$GENIE/src/stdapp/gNtpConv.cxx*' .

Synopsis

```
$ gntpc
   -i input_file_name
   -f format_of_output_file
   [-v format_version_number]
   [-c copy_MC_job_metadata?]
   [-o output_file_name]
   [-n number_of_events_to_convert]
```

where [] denotes an optional argument.

Description

The following options are available:

-i Specifies the name of the GENIE GHEP file to convert.

-f Specifies the output file format.

This can be any of the following:

- 'gst': The standard **G**ENIE **S**ummary **T**ree (gst) format (see subsection 7.6.2).

- 'gxml': The GENIE XML event format (see subsection 7.6.2).

- 'ghep_mock_data': Identical format as the input GHEP file but all information other than final state particles is hidden.

- 'rootracker': A bare-ROOT STDHEP-like event tree. Very similar to the native GHEP tree but with no dependency on GENIE classes (see subsection 7.6.2).

- 'rootracker_mock_data': Like 'rootracker' but with all information other than final state particles hidden.

- 't2k_rootracker': A variation of the 'rootracker' format used by some T2K detector MC chains (nd280). Includes, in addition, tree branches storing JNUBEAM flux simulation 'pass-through info'[1] (see subsection 7.6.2).

- 'numi_rootracker': A variation of the 'rootracker' format used by some NuMI beamline experiments. Includes, in addition, tree branches storing gNuMI flux simulation pass-through info (see subsection 7.6.2).

- 't2k_tracker': A tracker-type format with tweaks required by the SuperK MC (SKDETSIM) (see subsection 7.6.2).

- 'nuance_tracker': [Depreciated] The original tracker format (see subsection 7.6.2).

- 'ghad': [Depreciated] NEUGEN-style text-based format for hadronization studies.

- 'ginuke': A summary ntuple for intranuclear-rescattering studies using simulated hadron-nucleus samples.

-v Specifies the output file format version number.

This is an optional arument. It defaults to the latest version of each specified format. The option exists to maintain ability to generate old versions of certain formats.

-o Specifies the output file name.

This is an optional argument. By default, the output file name is constructed from the input *GHEP* file name by removing the '.ghep.root' (or just the '.root' one if '.ghep' is not present) extension and by appending:

- 'gst' format files: '*.gst.root*'

- 'gxml' format files: '*.gxml*'

- 'ghep_mock_data' format files: '*.mockd.ghep.root*'

- 'rootracker' format files: '*.gtrac.root*'

- 'rootracker_mock_data' format files: '*.mockd.gtrac.root*'

- 't2k_rootracker' format files: '*.gtrac.root*'

- 'numi_rootracker' format files: '*.gtrac.root*'

- 't2k_tracker' format files: '*.gtrac.dat*'

- 'nuance_tracker' format files: '*.gtrac_legacy.dat*'

- 'ghad' format files: '*.ghad.dat*'

- 'ginuke' format files: '*.ginuke.root*'

-n Specifies the number of events to convert.

This is an optional argument. By default, gntpc will convert all events in the input file.

---

[1]This refers to parent meson information for every flux neutrino for which GENIE generated an interaction.

Examples:

1. To convert all events in the *input GHEP* file '*myfile.ghep.root*' into the
   't2k_rootracker' format, type:

   ```
   $ gntpc -i myfile.ghep.root -f t2k_rootracker
   ```

   The output file is automatically named '*myfile.gtrac.root*'

2. To convert the first 20,000 events in the GHEP file '*myfile.ghep.root*' into the 'gst' format and name
   the output file '*out.root*', type:

   ```
   $ gntpc -i myfile.ghep.root -f gst -n 20000 -o out.root
   ```

7.6.2   Formats supported by *gntpc*

The 'gst' format

The 'gst' is a GENIE summary ntuple format. It is a simple, plain ntuple that can be easily used for
plotting in interactive ROOT sessions. The stored ROOT *TTree* contains the following branches:

- **iev** (*int*): Event number.

- **neu** (*int*): Neutrino PDG code.

- **tgt** (*int*): Nuclear target PDG code (10LZZZAAAI).

- **Z** (*int*): Nuclear target Z.

- **A** (*int*): Nuclear target A.

- **hitnuc** (*int*): Hit nucleon PDG code (not set for coherent, inverse muon decay and ve- elastic
  events).

- **hitqrk** (*int*): Hit quark PDG code (set for deep-inelastic scattering events only).

- **sea** (*bool*): Hit quark is from sea (set for deep-inelastic scattering events only).

- **resid** (*bool*): Produced baryon resonance id (set for resonance events only).

- **qel** (*bool*): Is it a quasi-elastic scattering event?

- **res** (*bool*): Is it a resonanec neutrino-production event?

- **dis** (*bool*): Is it a deep-inelastic scattering event?

- **coh** (*bool*): Is it a coherent meson production event?

- **dfr** (*bool*): Is it a diffractive meson production event?

- **imd** (*bool*): Is it an invese muon decay event?

- **nuel** (*bool*): Is it a ve- elastic event?

- **cc** (*bool*): Is it a CC event?

- **nc** (*bool*): Is it a NC event?

- **charm** (*bool*): Produces charm?

- **neut_code** (*int*): The equivalent NEUT reaction code (if any).

- **nuance_code** (*int*): The equivalent NUANCE reaction code (if any).

- **wght** (*double*): Event weight.

- **xs** (*double*): Bjorken x (as was generated during the kinematical selection / off-shell kinematics).

- **ys** (*double*): Inelasticity y (as was generated during the kinematical selection / off-shell kinematics).

- **ts** (*double*): Energy transfer to nucleus (nucleon) at coherent (diffractive) production events (as was generated during the kinematical selection).

- **Q2s** (*double*): Momentum transfer Q^2 (as was generated during the kinematical selection / off-shell kinematics) (in GeV^2).

- **Ws** (*double*): Hadronic invariant mass W (as was generated during the kinematical selection / off-shell kinematics).

- **x** (*double*): Bjorken x (as computed from the event record).

- **y** (*double*): Inelasticity y (as computed from the event record).

- **t** (*double*): Energy transfer to nucleus (nucleon) at coherent (diffractive) production events (as computed from the event record).

- **Q2** (*double*): Momentum transfer Q^2 (as computed from the event record) (in GeV^2).

- **W** (*double*): Hadronic invariant mass W (as computed from the event record).

- **Ev** (*double*): Incoming neutrino energy (in GeV).

- **pxv** (*double*): Incoming neutrino px (in GeV).

- **pyv** (*double*): Incoming neutrino py (in GeV).

- **pzv** (*double*): Incoming neutrino pz (in GeV).

- **En** (*double*): Initial state hit nucleon energy (in GeV).

- **pxn** (*double*): Initial state hit nucleon px (in GeV).

- **pyn** (*double*): Initial state hit nucleon py (in GeV).

- **pzn** (*double*): Initial state hit nucleon pz (in GeV).

- **El** (*double*): Final state primary lepton energy (in GeV).

- **pxl** (*double*): Final state primary lepton px (in GeV).

- **pyl** (*double*): Final state primary lepton py (in GeV).

- **pzl** (*double*): Final state primary lepton pz (in GeV).

- **nfp** (*int*): Number of final state p and $\bar{p}$ (after intranuclear rescattering).

- **nfn** (*int*): Number of final state n and $\bar{n}$.

- **nfpip** (*int*): Number of final state π^+.

- **nfpim** (*int*): Number of final state π^-.

- **nfpi0** (*int*): Number of final state π^0.

- **nfkp** (*int*): Number of final state K^+.

- **nfkm** (*int*): Number of final state K^-.

- **nfk0** (*int*): Number of final state K^0 and $\bar{K}^0$.

- **nfem** (*int*): Number of final state γ, e^- and e^+.

- **nfother** (*int*): Number of heavier final state hadrons (D+/-,D0,Ds+/-,Lamda,Sigma,Lamda_c,Sigma_c,...).

- **nip** (*int*): Number of 'primary' ('primary' : before intranuclear rescattering) p and $\bar{p}$.

- **nin** (int): Number of 'primary' n and $\bar{n}$.

- **nipip** (*int*): Number of 'primary' π^+.

- **nipim** (*int*): Number of 'primary' π^-.

- **nipi0** (*int*): Number of 'primary' π^0.

- **nikp** (*int*): Number of 'primary' K^+.

- **nikm** (*int*): Number of 'primary' K^-.

- **nik0** (*int*): Number of 'primary' K^0 and $\bar{K}^0$.

- **niem** (*int*): Number of 'primary' γ, e^- and e^+ (eg from nuclear de-excitations or from pre-intranuked resonance decays).

- **niother** (*int*): Number of other 'primary' hadron shower particles.

- **nf** (*int*): Number of final state particles in hadronic system.

- **pdgf** (*int[kNPmax]*): PDG code of kth final state particle in hadronic system.

- **Ef** (*double[kNPmax]*): Energy of kth final state particle in hadronic system (in GeV).

- **pxf** (*double[kNPmax]*): Px of kth final state particle in hadronic system (in GeV).

- **pyf** (*double[kNPmax]*): Py of kth final state particle in hadronic system (in GeV).

- **pzf** (*double[kNPmax]*): Pz of kth final state particle in hadronic system (in GeV).

- **ni** (*int*): Number of particles in the 'primary' hadronic system ('primary' : before intranuclear rescattering).

- **pdgi** (*int[kNPmax]*): PDG code of kth particle in 'primary' hadronic system.

- **Ei** (*double[kNPmax]*): Energy of kth particle in 'primary' hadronic system (in GeV).

- **pxi** (*double[kNPmax]*): Px of kth particle in 'primary' hadronic system (in GeV).

- **pyi** (*double[kNPmax]*): Py of kth particle in 'primary' hadronic system (in GeV).

- **pzi** (*double[kNPmax]*): Pz of kth particle in 'primary' hadronic system (in GeV).

- **vtxx** (*double*): Vertex x in detector coord system (in SI units).

- **vtxy** (*double*): Vertex y in detector coord system (in SI units).

- **vtxx** (*double*): Vertex z in detector coord system (in SI units).

- **vtxt** (*double*): Vertex t in detector coord system (in SI units).

- **calresp0** (*double*): An approximate calorimetric response to the generated hadronic vertex actibity, calculated by summing up: the kinetic energy for generated $\{\pi^+, \pi^-, p, n\}$, the energy+mass for generated $\{\bar{p}, \bar{n}\}$, the (e/h)*energy for generated $\{\pi^0, \gamma, e^-, e^+\}$ (with an e/h = 1.3) and the kinetic energy for any other generated particle.

Using ROOT to plot quantities stored in a 'gst' ntuple   The 'gst' summary ntuples make it especially easy to plot GENIE information in a ROOT/CINT session. Some examples are given below:

1. To draw a histogram of the final state primary lepton energy for all ν_μ CC DIS interactions with an invariant mass $W > 3$ GeV, then type:

   ```
   root[0] gst->Draw('‘El'',''dis&&cc&&neu==14&&Ws>3'');
   ```

2. To draw a histogram of all final state π^+ energies in CC RES interactions, then type:

   ```
   root[0] gst->Draw('‘Ef'',''pdgf==211&&res&&cc'');
   ```

The 'gxml' format

The 'gxml' format is a GENIE XML-based event format[2].

Each event is included within <ghep> </ghep> tags as in:

```
<ghep np       = "{number of particles; int}"
     unphysical = "{is it physical?; boolean (T/F)}">

</ghep>
```

Both information with event-wide scope such as:

```
<wght>      {event weight; double}        </wght>
<xsec_evnt> {event cross section; double} </xsec_evnt>
<xsec_kine> {cross section for event kinematics; double} </xsec_kine>

<vx> {vertex x in detector coord system (SI); double} </vx>
<vy> {vertex y in detector coord system (SI); double} </vy>
<vz> {vertex z in detector coord system (SI); double} </vz>
<vt> {vertex t (SI); double} </vt>
```

---

[2]In the format description that follows, the curly braces within tags are to be 'viewed' as a single value of the specified type with the specified semantics. For example '{number of particles; int}' is to be thought of as an integer value describing a number particles.

and a full list of the generated particles is included between the <ghep> tags. The information for each generated particle is expressed as:

```
<p idx  = "{particle index in event record; int}"
   type = "{particle type; char (F[ake]/P[article]/N[uleus])}">

   <pdg> {pdg code; int}    </pdg>
   <ist> {status code; int} </ist>

   <mother>
           <fst> {first mother index; int} </fst>
           <lst> {last  mother index; int} </lst>
   </mother>
   <daughter>
           <fst> {first daughter index; int} </fst>
           <lst> {last  daughter index; int} </lst>
   </daughter>

   <px> {Px in GeV; double}  </px>
   <py> {Py in GeV; double} </py>
   <pz> {Pz in GeV; double}  </pz>
   <E>  {E  in GeV; double}  </E>
   <x>  {x  in fm;  double}  </x>
   <y>  {y  in fm;  double}  </y>
   <z>  {z  in fm;  double}  </z>
   <t>  {t; always set to 0} </t>

   <ppolar> {polarization, polar angle;      in rad} </ppolar>
   <pazmth> {polarization, azimuthal angle; in rad} </pazmth>
</p>
```

The 'rootracker' formats

The 'rootracker' format is a standardized bare-ROOT GENIE event tree format evolved from work on integrating the GENIE simulations with the nd280, INGRID and 2km detector-level simulations. In recent versions of GENIE that format was renamed to 't2k_rootracker', with 'rootracker' now being a more generic, stripped-down (excudes pass-through JPARC flux info etc.) version of the T2K variance.

The 'rootracker' tree branch names, leaf types and a short description is given below. For the JNUBEAM branches please consult the corresponding documentation:

- **EvtNum** (*int*): Event number

- **EvtFlags** (*TBits**): [GENIE] Event flags.

- **EvtCode** (*TObjString**): [GENIE] A string event code.

- **EvtXSec** (*double*): [GENIE] Event cross section (in $10^{38}cm^2$).

- **EvtDXSec** (*double*): [GENIE] Differential cross section for the selected kinematics in the K^n space (in $10^{38}cm^2/$ $[K^n]$). Typically, K^n is: $\{Q^2\}$ for QEL, $\{Q^2, W\}$ for RES, $\{x, y\}$ for DIS and COH, $\{y\}$ for ve^- etc.

- **EvtWght** (*double*): [GENIE] Event weight.

- **EvtProb** (*double*): [GENIE] Event probability (given cross section, density-weighted path-length, etc).

- **EvtVtx** (*double[4]*): [GENIE] Event vertex position (x, y, z, t) in the detector coordinate system (in SI).

- **StdHepN** (*int*): [GENIE] Number of entries in the particle array.

- **StdHepPdg** (*int*): [GENIE] k^{th} particle PDG code.

- **StdHepStatus** (*int*): [GENIE] k^{th} particle status code (Generator-specific: For GENIE see *GHepStatus_t*).

- **StdHepRescat** (*int*): [GENIE] k^{th} particle intranuclear rescattering code (Hadron-transport model specific: For INTRANUKE/hA see INukeFateHA_t).

- **StdHepX4** (*double [kNPmax][4]*): [GENIE] k^{th} particle 4-position (x, y, z, t) in the hit nucleus rest frame (in fm)

- **StdHepP4** (*double [kNPmax][4]*): [GENIE] k^{th} particle 4-momentum (px, py, pz, E) in the LAB frame (in GeV).

- **StdHepPolz** (*double [kNPmax][3]*): [GENIE] k^{th} particle polarization vector.

- **StdHepFd** (*int [kNPmax]*): [GENIE] k^{th} particle first-daughter index.

- **StdHepLd** (*int [kNPmax]*): [GENIE] k^{th} particle last-daughter index.

- **StdHepFm** (*int [kNPmax]*): [GENIE] k^{th} particle first-mother index.

- **StdHepLm** (*int [kNPmax]*): [GENIE] k^{th} particle last-mother index.

The following branches exist only in the 't2k_rootracker' variance:

- **NuParentPdg** (*int*): [JNUBEAM] Parent PDG code.

- **NuParentDecMode** (*int*): [JNUBEAM] Parent decay mode.

- **NuParentDecP4** (*double [4]*): [JNUBEAM] Parent 4-momentum at decay.

- **NuParentDecX4** (*double [4]*): [JNUBEAM] Parent 4-position at decay.

- **NuParentProP4** (*double [4]*): [JNUBEAM] Parent 4-momentum at production.

- **NuParentProX4** (*double [4]*): [JNUBEAM] Parent 4-position at production.

- **NuParentProNVtx** (*int*): [JNUBEAM] Parent vertex id.

- **G2NeutEvtCode** (*int*): corresponding NEUT reaction code for the GENIE event.

  The following branches exist only in the 'numi_rootracker' variance[3]:

- **NumiFluxRun** (*int*): [GNUMI] Run number.

- **NumiFluxEvtno** (*int*): [GNUMI] Event number (proton on target).

- **NumiFluxNdxdz** (*double*): [GNUMI] Neutrino direction slope (dx/dz) for a random decay.

- **NumiFluxNdydz** (*double*): [GNUMI] Neutrino direction slope (dy/dz) for a random decay.

- **NumiFluxNpz** (*double*): [GNUMI] Neutrino momentum (GeV/c) along z direction (beam axis).

- **NumiFluxNenergy** (*double*): [GNUMI] Neutrino energy (GeV/c) for a random decay.

- **NumiFluxNdxdznea** (*double*): [GNUMI] Neutrino direction slope (dx/dz) for a decay forced at center of near detector.

- **NumiFluxNdydznea** (*double*): [GNUMI] Neutrino direction slope (dy/dz) for a decay forced at center of near detector.

- **NumiFluxNenergyn** (*double*): [GNUMI] Neutrino energy for a decay forced at center of near detector.

- **NumiFluxNwtnear** (*double*): [GNUMI] Neutrino weight for a decay forced at center of near detector.

- **NumiFluxNdxdzfar** (*double*): [GNUMI] Neutrino direction slope (dx/dz) for a decay forced at center of far detector.

- **NumiFluxNdydzfar** (*double*): [GNUMI] Neutrino direction slope (dy/dz) for a decay forced at center of far detector.

- **NumiFluxNenergyf** (*double*): [GNUMI] Neutrino energy for a decay forced at center of far detector.

- **NumiFluxNwtfar** (*double*): [GNUMI] Neutrino weight for a decay forced at center of far detector.

- **NumiFluxNorig** (*int*): [GNUMI] Obsolete

- **NumiFluxNdecay** (*int*): [GNUMI] Decay mode that produced neutrino[4]

---

[3] More details on the GNuMI beam simulation outputs can be found at
http://www.hep.utexas.edu/~zarko/wwwgnumi/v19/

[4]

  - 1: K0L -> nue pi- e+
  - 2: K0L -> nuebar pi+ e-
  - 3: K0L -> numu pi- mu+
  - 4: K0L -> numubar pi+ mu-
  - 5: K+ -> numu mu+
  - 6: K+ -> nue pi0 e+
  - 7: K+ -> numu pi0 mu+
  - 8: K- -> numubar mu-

- **NumiFluxNtype** (*int*): [GNUMI] Neutrino flavor.

- **NumiFluxVx** (*double*): [GNUMI] Position of hadron/muon decay, X coordinate.

- **NumiFluxVy** (*double*): [GNUMI] Position of hadron/muon decay, Y coordinate.

- **NumiFluxVz** (*double*): [GNUMI] Position of hadron/muon decay, Z coordinate.

- **NumiFluxPdpx** (*double*): [GNUMI] Parent momentum at decay point, X - component.

- **NumiFluxPdpy** (*double*): [GNUMI] Parent momentum at decay point, Y - component.

- **NumiFluxPdpz** (*double*): [GNUMI] Parent momentum at decay point, Z - component.

- **NumiFluxPpdxdz** (*double*): [GNUMI] Parent dx/dz direction at production.

- **NumiFluxPpdydz** (*double*): [GNUMI] Parent dy/dz direction at production.

- **NumiFluxPppz** (*double*): [GNUMI] Parent Z momentum at production.

- **NumiFluxPpenergy** (*double*): [GNUMI] Parent energy at production.

- **NumiFluxPpmedium** (*int*): [GNUMI] Tracking medium number where parent was produced.

- **NumiFluxPtype** (*int*): [GNUMI] Parent particle ID (PDG)

- **NumiFluxPpvx** (*double*): [GNUMI] Parent production vertex, X coordinate (cm).

- **NumiFluxPpvy** (*double*): [GNUMI] Parent production vertex, Y coordinate (cm).

- **NumiFluxPpvz** (*double*): [GNUMI] Parent production vertex, Z coordinate (cm).

- **NumiFluxMuparpx** (*double*): [GNUMI] Repeat of information above, but for muon neutrino parents.

- **NumiFluxMuparpy** (*double*): [GNUMI] -//-.

- **NumiFluxMuparpz** (*double*): [GNUMI] -//-.

- **NumiFluxMupare** (*double*): [GNUMI] -//-.

- **NumiFluxNecm** (*double*): [GNUMI] Neutrino energy in COM frame.

- **NumiFluxNimpwt** (*double*): [GNUMI] Weight of neutrino parent.

- **NumiFluxXpoint** (*double*): [GNUMI] Unused.

- **NumiFluxYpoint** (*double*): [GNUMI] Unused.

- **NumiFluxZpoint** (*double*): [GNUMI] Unused.

- 9: K- -> nuebar pi0 e-
- 10: K- -> numubar pi0 mu-
- 11: mu+ -> numubar nue e+
- 12: mu- -> numu nuebar e-
- 13: pi+ -> numu mu+
- 14: pi- -> numubar mu-

- **NumiFluxTvx** (*double*): [GNUMI] Exit point of parent particle at the target, X coordinate.

- **NumiFluxTvy** (*double*): [GNUMI] Exit point of parent particle at the target, Y coordinate.

- **NumiFluxTvz** (*double*): [GNUMI] Exit point of parent particle at the target, Z coordinate.

- **NumiFluxTpx** (*double*): [GNUMI] Parent momentum exiting the target, X - component.

- **NumiFluxTpy** (*double*): [GNUMI] Parent momentum exiting the target, Y- component.

- **NumiFluxTpz** (*double*): [GNUMI] Parent momentum exiting the target, Z - component.

- **NumiFluxTptype** (*double*): [GNUMI] Parent particle ID exiting the target.

- **NumiFluxTgen** (*double*): [GNUMI] Parent generation in cascade[5].

- **NumiFluxTgptype** (*double*): [GNUMI] Type of particle that created a particle flying of the target.

- **NumiFluxTgppx** (*double*): [GNUMI] Momentum of a particle, that created a particle that flies off the target (at the interaction point), X - component.

- **NumiFluxTgppy** (*double*): [GNUMI] Momentum of a particle, that created a particle that flies off the target (at the interaction point), Y - component.

- **NumiFluxTgppz** (*double*): [GNUMI] Momentum of a particle, that created a particle that flies off the target (at the interaction point), Z - component.

- **NumiFluxTprivx** (*double*): [GNUMI] Primary particle interaction vertex, X coordinate.

- **NumiFluxTprivy** (*double*): [GNUMI] Primary particle interaction vertex, Ycoordinate.

- **NumiFluxTprivz** (*double*): [GNUMI] Primary particle interaction vertex, Z coordinate.

- **NumiFluxBeamx** (*double*): [GNUMI] Primary proton origin, X coordinate.

- **NumiFluxBeamy** (*double*): [GNUMI] Primary proton origin, Y coordinate.

- **NumiFluxBeamz** (*double*): [GNUMI] Primary proton origin, Z coordinate.

- **NumiFluxBeampx** (*double*): [GNUMI] Primary proton momentum, X - component.

- **NumiFluxBeampy** (*double*): [GNUMI] Primary proton momentum, Y - component.

- **NumiFluxBeampz** (*double*): [GNUMI] Primary proton momentum, Z - component.

The 'tracker' formats

The 'tracker'-type format is a legacy event format used by some fortran-based event generators (eg. NUANCE) and detector-level simulations (eg. SuperK's Geant3-based SKDETSIM). GENIE includes a number of 'tracker' format variations:

---

[5]

- 1: Primary proton,
- 2: Particles produced by proton interaction,
- 3: Particles produced by interactions of the 2's,
- ...

* 't2k_tracker':

This is tracker-type format with all the tweaks required for passing GENIE events into the Geant3-based SuperK detector MC. In the 't2k_tracker' files:

- The begging of event file is marked with a `$begin` line, while the end of the file is marked by an `$end` line.

- Each new event is marked with a `$genie` line. What follows is a reaction code. Since GENIE doesn't use integer reaction codes, it is writting-out the corresponding NEUT reaction code for the generated GENIE event. This simplifies comparisons between the GENIE and NEUT samples in SuperK physics analyses.

- The `$vertex` line is being used to pass the interaction vertex position in the detector coordinate system in SI units

- The `$track` lines are being used to pass minimal information on (some) initial / intermediate state particles (as expected by SKDETSIM) and all final state particles to be tracked by the detector simulation. Each `$track` line includes the particle PDG code, its energy, its direction cosines and a 'status code' (Not to be confused with GENIE's status code. The 'tracker' file status code expected by SKDETSIM is '-1' for initial state particles, '0' for stable final states and '-2' for intermediate particles).

Some further clarifications are in order here:

- K^0, $\bar{K}^0$ generated by GENIE are converted to K^0_L, K^0_S (as expected by SKDETSIM)

- Since no mother / daughter associations are stored in `$track` lines only one level of intermediates can exist (the 'primary' hadronic system). Any intermediate particles corresponding to states evolved from the 'primary' hadronic state but before reaching the 'final state' are neglected.

- The `$track` line ordering is the one expected by SKDETSIM with all the primaries, intermediates and final states grouped together.

The 't2k_tracker' format includes a set of `$info` lines. They include the exact same information as the one stored 't2k_rootracker' format files (complete event information generated by GENIE and JPARC / JNUBEAM flux pass-through information). This is partially redundant information (some of it was included in the minimal `$track` lines) that is not intended for pushing particles through the detector simulation. The `$info` lines are read-in by SKDETSIM and are passed-through to the DST stage so that the identical, full MC information is available for events simulated on both SuperK and the near detector complex (thus enabling global systematic studies).

A complete event in 't2k_tracker' format looks-like:

```
$begin

$genie  {neut_like_event_type}
$vertex {vtxx} {vtxy} {vtxz} {vtxt}

$track  {pdg code} {E} {dcosx} {dcosy} {dcosz} {status}
$track  {pdg code} {E} {dcosx} {dcosy} {dcosz} {status}
...
$track  {pdg code} {E} {dcosx} {dcosy} {dcosz} {status}
```

```
$info {event_num} {error_code} {genie_event_type}
$info {event_xsec} {event_kinematics_xsec} {event_weight} {event_probability}
$info {vtxx} {vtxy} {vtxz} {vtxt}

$info {nparticles}
$info {idx}{pdg}{status}{fd}{ld}{fm}{lm}{px}{py}{pz}{E}{x}{y}{z}{t}{polx}{poly}{polz}
$info {idx}{pdg}{status}{fd}{ld}{fm}{lm}{px}{py}{pz}{E}{x}{y}{z}{t}{polx}{poly}{polz}
...
$info {idx}{pdg}{status}{fd}{ld}{fm}{lm}{px}{py}{pz}{E}{x}{y}{z}{t}{polx}{poly}{polz}

$info (jnubeam_parent_pdg) (jnubeam_parent_decay_mode)
$info (jnubeam_dec_px) (jnubeam_dec_py) (jnubeam_dec_pz) (jnubeam_dec_E)
$info (jnubeam_dec_x)  (jnubeam_dec_y)  (jnubeam_dec_z)  (jnubeam_dec_t)
$info (jnubeam_pro_px) (jnubeam_pro_py) (jnubeam_pro_pz) (jnubeam_pro_E)
$info (jnubeam_pro_x)  (jnubeam_pro_y)  (jnubeam_pro_z)  (jnubeam_pro_t)
$info (jnubeam_nvtx)

$end
```

7.7   Units

GENIE is using the natural system of units ($\hbar = c = 1$) so (almost) everything is expressed in $[GeV]^n$. Notable exceptions are the event vertex (in SI units, in the detector coordinate system) and particle positions (in fm, in the hit nucleus coordinate system). Additionally, although internally all cross sections are expressed in the natural system units, values copied to certain files (eg 'rootracker'- or 'tracker'-format files) are converted to $10^{38}cm^2$ (See the corresponding documentation for these file formats).

GENIE provides an easy way for converting back and forth between its internal, natural system of units and other units. The conversion factors are included in '*$GENIE/src/Conventions/Units.h*'.

For exampe, in order to convert a cross section value returned by '*a_ function()*' from the natural system of units to $10^{38}cm^2$, type:

```
double xsec = a_function() / (1E-38 * units::cm2);
```

Chapter 8

Non-Neutrino Event Generation Modes

8.1 Introduction

[to be added in future revision]

8.2 Hadron (and Photon) - Nucleus scattering

[to be added in future revision]

8.2.1 The *gevgen_ hadron* event generation application

Name

gevgen_ hadron - A GENIE hadron+nucleus event generation application.

Source

The source code for this utility may be found in '*$GENIE/src/stdapp/*gEvGenHadronNucleus.cxx'.

```
Synopsis

$ gevgen_hadron
   [-n number_of_events] -p probe_pdg_code -t target_pdg_code
   -k kinetic_energy [-m mode]
   [-f flux] [-o output_file_prefix][-r run#|
   [-seed random_number_seed] [--message-thresholds xml_file]
   [--event-record-print-level level] [--mc-job-status-refresh-rate rate]
```

Description

The following options are available:

-n Specifies the number of events to generate. This is an optional argument. By default it is set to '10000'.

-p Specifies the incoming hadron PDG code.   The choice is limited to the hadrons that can be handled by the intranuclear cascade code that is invoked by the application (choice made via the -m option).

-t Specifies the nuclear target PDG code. As usual the PDG2006 convention is used (10LZZZA-AAI). So, for example, O^{16} code = 1000080160, Fe^{56} code = 1000260560. For more details see Appendix D.

-k Specifies the incoming hadron's kinetic energy (range). This option can be use to specify either a single kinetic energy value (eg `-k 0.5`) or a kinetic energy range as a comma-separated set of numbers (eg `-k 0.1,1.2`). The input values are taken to be in GeV. If no flux is specified then hadrons will be fired towards the nucleus with a uniform kinetic energy distribution within the specified range. If a kinetic energy spectrum is supplied then the hadron kinetic energies will be generated using the input spectrum within the specified range.

-f Specifies the incoming hadron's kinetic energy spectrum.   This is an optional argument. It can be either: a) a function, eg `x*x+4*exp(-x)`, or b) a text file containing 2 columns corresponding to (kinetic energy {GeV}, 'flux'). If you do specify a flux then you need to specify a kinetic energy range (not kust a single value).

-o Specifies the output filename prefix. This is an optional argument. It allows you to override the output event file prefix. In GENIE, the output filename is built as:
    `prefix.run_number.event_tree_format.file_format` where, in *gevgen_hadro*, by default, `prefix`: 'gntp' and `event_tree_format`: 'ghep' and `file_format`: 'root'.

-m Specifies which intranuclear cascade model to use.   This is an optional argument. Possible options are 'hA' (for the INTRANUKE hA model), 'hN' (for the INTRANUKE hN model). By default it is set to 'hA'.

-r Specifies the run number. This is an optional argument. By default it is set to '0'.

–seed Specifies the random number seed for the current job.

–message-thresholds Specifies the GENIE verbosity level. The verbosity level is controlled with an XML file allowing users to customize the threshold of each message stream. The XML schema can be seen in '*\$GENIE/config/Messenger.xml*'. The '*Messenger.xml*' file contains the default thresholds used by GENIE. The '*Messenger_laconic.xml*' and '*Messenger_rambling.xml*' files define, correspondingly, less and more verbose configurations.

–event-record-print-level Allows users to set the level of information shown when the event 94 record is printed in the screen. See GHepRecord::Print() for allowed settings.

–mc-job-status-refresh-rate Allows users to customize the refresh rate of the status file.

Examples

  1. Generate 100k $\pi^+ + Fe^{56}$ events with a π^+ kinetic energy of 165 MeV. Use seed number 10010.

     ```
     $ gevgen_hadron -n 100000 -p 211 -t 1000260560 -k 0.165 --seed 10010
     ```

ID	Decay channel	Current limit ($\times 10^{34}$ yrs)
0	$p \to e^+\pi^0$	1.3
1	$p \to \mu^+\pi^0$	1.1
2	$p \to e^+\eta^0$	0.42
3	$p \to \mu^+\eta^0$	0.13
4	$p \to e^+\rho^0$	0.07
5	$p \to \mu^+\rho^0$	0.02
6	$p \to e^+\omega^0$	0.03
7	$p \to \mu^+\omega^0$	0.08
8	$n \to e^+\pi^-$	0.2
9	$n \to \mu^+\pi^-$	0.1
10	$p \to \bar{\nu}K^+$	0.4

Table 8.1: Nucleon decay modes simulated in GENIE.

2. Generate 100k $\pi^+ + Fe^{56}$ events with the π^+ kinetic energy distributed uniformly in the [165 MeV, 1200 MeV] range. Use default seed number.

```
$ gevgen_hadron -n 100000 -p 211 -t 1000260560 -k 0.165,1.200
```

3. Generate 100k $\pi^+ + Fe^{56}$ events with the π^+ kinetic energy distributed as f(KE) = 1/KE in the [165 MeV, 1200 MeV] range. Use seed number 10010 and production-mode verbosity level (all message thresholds set to warning).

```
$ gevgen_hadron -n 100000 -p 211 -t 1000260560 -k 0.165,1.200 -f '1/x'
--seed 10010 --message-thresholds Messenger_laconic.xml
```

8.3 Charged Lepton - Nucleus scattering

[to be added in future revision]

8.4 Nucleon decay

The simulated nucleon decay modes are given in Tab.8.1. The primary decay is simulated using a phase-space-decay generator. For bound nucleons, the nuclear environment is simulated as in neutrino scattering. The nucleon is assigned a Fermi momentum and removal energy and it is off the mass shell. The propagation of decay products is simulated using an intranuclear cascade Monte Carlo.

[expand]

8.4.1 The *gevgen_ndcy* event generation application

Name

gevgen_ndcy - A GENIE-based nucleon decay event generation application.

Source and build options

The source code for *gevgen_ndcy* may be found in
'*$*GENIE/*src/support/ndcy/*EvGen/gNucleonDecayEvGen.cxx'.
To enable it add '`--enable-nucleon-decay`' during the GENIE build configuration.


```
Synopsis
```

```
$ gevgen_ndcy
    -n number_of_events -m nucleon_decay_mode
    -g geometry [-t geometry_top_volume_name]
    [-L geometry_length_units] [-D geometry_density_units]
    [-o output_event_file_prefix] [-r run#]
    [-seed random_number_seed] [--message-thresholds xml_file]
    [--event-record-print-level level] [--mc-job-status-refresh-rate rate]
    [-h]
```

where [] denotes an optional argument.

Description

The following options are available:

-n Specifies the number of events to generate.

-m Specifies the nucleon decay channel ID. The list of decay channels and the corresponding ID is given in Tab. 8.1

-g Specifies the input detector geometry. This option can be used to specify any of:

- A ROOT file containing a ROOT / Geant4-based geometry description (*TGeoManager*).

  Example:
  To use the ROOT detector geometry description stored in the '*/data/geo/laguna.root*' file, type:
  '`-g /data/geo/laguna.root`'

  By default the entire input geometry will be used. Use the '`-t`' option to allow event generation only on specific geometry volumes.

- A mix of target materials, each with its corresponding weight.
  The target mix is specified as a comma-separated list of nuclear PDG codes (in the PDG2006 convention: 10LZZZAAAI) followed by their corresponding weight fractions in brackets, as in:
  '`-t code1[fraction1],code2[fraction2],...`'

  Example 1:
  To use a target mix of 88.79% (weight fraction) O^{16} and 11.21% H (aka 'water') type:
  '`-g 1000080160[0.8879],1000010010[0.1121]`'

  Example 2:
  To use a target which is 100% C^{12}, type:
  '`-g 1000060120`'

-t Specifies the input top volume for event generation. This is an optional argument, relevant only for ROOT-based detector geometry descriptions. By default, it is set to be the 'master volume' of the input geometry resulting in neutrino events being generated over the entire geometry volume. If the '`-t`' option is set, event generation will be confined in the specified detector volume. The option can be used to simulate events at specific sub-detectors.

You can use the '`-t`' option to switch generation on / off at multiple volumes. For further details, see similar discussion in the description of other event generation applications (eg. *gevgen_t2k*).

-L Specifies the input geometry length units. This is an optional argument, relevant only for ROOT-based detector geometry descriptions. By default, that option is set to 'mm'. Possible options include: 'm', 'cm', 'mm', ...

-D Specifies the input geometry density units. This is an optional argument, relevant only for ROOT-based detector geometry descriptions. By default, that option is set to 'g_cm3'. Possible options include: 'kg_m3', 'g_cm3', 'clhep_def_density_unit',...

-o Specifies the output filename prefix. This is an optional argument. It allows you to override the output event file prefix. In GENIE, the output filename is built as:

'`prefix.run_number.event_tree_format.file_format`' where, in *gevgen_hadro*, by default, `prefix`: 'gntp' and `event_tree_format`: 'ghep' and `file_format`: 'root'.

-r Specifies the run number. This is an optional argument. By default it is set to '0'.

–seed Specifies the random number seed for the current job.

–message-thresholds Specifies the GENIE verbosity level. The verbosity level is controlled with an XML file allowing users to customize the threshold of each message stream. The XML schema can be seen in '*$GENIE/config/Messenger.xml*'. The '*Messenger.xml*' file contains the default thresholds used by GENIE. The '*Messenger_laconic.xml*' and '*Messenger_rambling.xml*' files define, correspondingly, less and more verbose configurations.

–event-record-print-level Allows users to set the level of information shown when the event 94 record is printed in the screen. See GHepRecord::Print() for allowed settings.

–mc-job-status-refresh-rate Allows users to customize the refresh rate of the status file.

Chapter 9

Event Reweighting

9.1 Introduction

This chapter describes strategies for propagating neutrino interaction uncertainties. The reweighting schemes described here are tied to the default physics choices made in GENIE and they have been implemented in the GENIE ReWeight package. As GENIE evolves, by including better-motivated theoretical models and integrating new data in its effective models[1], the reweighting schemes need to be updated. This evolution can not always be transparent but, to aid the user, we strive to keep this part of the user and physics manual up to date. Also, as our understanding and the systematic analysis of the GENIE model improves, new reweighting schemes are added.

For each neutrino-generator input physics quantity P, whose uncertainty is taken into account in this work, we introduce a systematic parameter[2] x_P. Tweaking this systematic parameter modifies the corresponding physics parameter P as follows:

$$P \rightarrow P' = P(1 + x_P * \frac{\delta P}{P})$$ (9.1)

where δP is the estimated standard deviation of P. Setting the systematic parameter to zero corresponds to using the nominal value of the physics parameter. Tweaking the systematic parameter by ± 1 modifies the corresponding physics quantity P by $\pm \delta P$. In this section, we provide a summary of all the systematic parameters included in this work, and a brief description of the corresponding tweaked physics quantities P and, wherever possible, the assumed fractional errors $\delta P/P$. The quantity P may be a single configurable parameter (eg. $CCQE$ axial mass), or it may be a simple function of a kinematical parameter (eg. a hadron-nucleus cross-section as a function of the hadron energy), or, more generally, it may be any nominal MC prediction, which can not be easily expressed analytically or tabulated. For that reason, it is always preferable to formulate the problem (eg. oscillation fits in presence of neutrino-interaction nuisance parameters) in terms of x_P.

A number of neutrino cross section systematics are considered in this chapter, and a complete list of these is given in Tab. 9.1. The dominant systematics, for neutrino interactions in the few-GeV energy range, include the axial mass for charged-current quasi-elastic scattering and the axial and vector masses for both charged-current and neutral-current resonance neutrino production. Uncertainties in nuclear effects (Pauli supression) in charged-current quasi-elastic reactions are taken into account by modifying the Fermi momentum level k_F. Uncertainties in the choice of vector form factors (dipole vs BBA2005) for

---

[1] The GENIE development roadmap is outlined at: http://releases.genie-mc.org

[2] The terms 'systematic parameter', 'nuisance parameter', 'tweaking dial' may be used interchangeably in this paper and our presentations/discussions of this work.

charged-current quasi-elastic reactions are also taken into account. Charged-current and neutral-current coherent pion production uncertainties are taken into account by modifying the corresponding axial mass and the nuclear size parameter R_0, which controls the pion absorption factor in the Rein-Sehgal (RS) model. Uncertainties in the level of the non-resonance background are considered for all neutrino charged-current and neutral-current 1π- and 2π-production channels. Finally, in order to consider uncertainties in charged-current and neutral-current deep inelastic scattering, the most important parameters of the Bodek-Yang (BY) model are taken into account. These BY uncertainties are considered only for events in the 'safe' deep-inelastic kinematic regime ($Q^2 > 1\ GeV^2/c^2$ and $W > 2\ GeV/c^2$) to avoid double counting uncertainties in the resonance / transition region that have already been taken into account.

We consider a number of uncertainties in neutrino-induced hadronization and resonance decays. We include uncertainties in the assignment of pion kinematics in $N\pi$ hadronic states generated by the Andreopoulos-Gallagher-Kehayias-Yang (AGKY) GENIE hadronization model, as well as uncertainties in the in-medium modifications of the hadronization process. Uncertainties in the pion angular distribution in $\Delta \to \pi N$ decays and uncertainties in certain resonance-decay branching ratios are also taken into account. The complete list is given in Tab. 9.2.

Finally, we consider two kinds of uncertainties affecting the INTRANUKE (hA) intranuclear hadron transport model: Uncertainties in the total rescattering probability (mean free path) for hadrons within the target nucleus and uncertainties in the conditional probability of each hadron rescattering mode (elastic, inelastic, charge exchange, pion production and absorption / multi-nucleon knock-out), given that a rescattering did occur. These physics uncertainties are considered separately for nucleons and pions. The complete list of systematic parameters is given in Tab. 9.3.

x_P	Description of P	$\delta P/P$
$x_{M_A^{NCEL}}$	Axial mass for NC elastic	$\pm 25\%$
$x_{\eta^{NCEL}}$	Strange axial form factor η for NC elastic	$\pm 30\%$
$x_{M_A^{CCQE}}$	Axial mass for CC quasi-elastic	-15% +25%
$x_{CCQE-Norm}$	Normalization factor for CCQE	
$x_{CCQE-PauliSup}$	CCQE Pauli suppression (via changes in Fermi level k_F)	$\pm 35\%$
$x_{CCQE-VecFF}$	Choice of CCQE vector form factors (BBA05 $\leftrightarrow$ Dipole)	-
$x_{CCRES-Norm}$	Normalization factor for CC resonance neutrino production	
$x_{NCRES-Norm}$	Normalization factor for NC resonance neutrino production	
$x_{M_A^{CCRES}}$	Axial mass for CC resonance neutrino production	$\pm 20\%$
$x_{M_V^{CCRES}}$	Vector mass for CC resonance neutrino production	$\pm 10\%$
$x_{M_A^{NCRES}}$	Axial mass for NC resonance neutrino production	$\pm 20\%$
$x_{M_V^{NCRES}}$	Vector mass for NC resonance neutrino production	$\pm 10\%$
$x_{M_A^{COHpi}}$	Axial mass for CC and NC coherent pion production	$\pm 50\%$
$x_{R_0^{COHpi}}$	Nuclear size param. controlling π absorption in RS model	$\pm 10\%$
$x_{R_{bkg}^{\nu p,CC1\pi}}$	Non-resonance bkg in νp $CC1\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu p,CC2\pi}}$	Non-resonance bkg in νp $CC2\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu n,CC1\pi}}$	Non-resonance bkg in νn $CC1\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu n,CC2\pi}}$	Non-resonance bkg in νn $CC2\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu p,NC1\pi}}$	Non-resonance bkg in νp $NC1\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu p,NC2\pi}}$	Non-resonance bkg in νp $NC2\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu n,NC1\pi}}$	Non-resonance bkg in νn $NC1\pi$ reactions	$\pm 50\%$
$x_{R_{bkg}^{\nu n,NC2\pi}}$	Non-resonance bkg in νn $NC2\pi$ reactions	$\pm 50\%$
$x_{A_{HT}^{BY}}$	A_{HT} higher-twist param in BY model scaling variable ξ_w	$\pm 25\%$
$x_{B_{HT}^{BY}}$	B_{HT} higher-twist param in BY model scaling variable ξ_w	$\pm 25\%$
$x_{C_{V1u}^{BY}}$	C_{V1u} u valence GRV98 PDF correction param in BY model	$\pm 30\%$
$x_{C_{V2u}^{BY}}$	C_{V2u} u valence GRV98 PDF correction param in BY model	$\pm 40\%$
x_{CCDIS}	Inclusive CC cross-section normalization factor	
$x_{CC\bar{\nu}/\nu}$	$\bar{\nu}/\nu$ CC ratio	
$x_{DIS-NuclMod}$	DIS nuclear modification (shadowing, anti-shadowing, EMC)	

Table 9.1: Neutrino interaction cross-section systematic parameters considered in GENIE. For some of the above parameters there are two reweighting implementations: One which includes the full effect of the systematic (shape + normalization) and one which includes only its effect on the shape of observable distributions (maintains normalization). Note that some systematics have overlapping effects so care is needed to avoid double counting.

x_P	Description of P	$\delta P/P$
$x_{AGKY}^{pT1\pi}$	Pion transverse momentum (p_T) for $N\pi$ states in AGKY	-
$x_{AGKY}^{xF1\pi}$	Pion Feynman x (x_F) for $N\pi$ states in AGKY	-
x_{fz}	Hadron formation zone	$\pm 50\%$
$x_{\theta_\pi}^{\Delta \to \pi N}$	Pion angular distribution in $\Delta \to \pi N$ (isotropic $\leftrightarrow$ RS)	-
$x_{BR}^{R \to X+1\gamma}$	Branching ratio for radiative resonance decays	$\pm 50\%$
$x_{BR}^{R \to X+1\eta}$	Branching ratio for single-η resonance decays	$\pm 50\%$

Table 9.2:   Neutrino-induced hadronization and resonance-decay systematic parameters considered in this work.

x_P	Description of P	$\delta P/P$
x_{mfp}^{N}	Nucleon mean free path (total rescattering probability)	$\pm 20\%$
x_{cex}^{N}	Nucleon charge exchange probability	$\pm 50\%$
x_{el}^{N}	Nucleon elastic reaction probability	$\pm 30\%$
x_{inel}^{N}	Nucleon inelastic reaction probability	$\pm 40\%$
x_{abs}^{N}	Nucleon absorption probability	$\pm 20\%$
x_{π}^{N}	Nucleon π-production probability	$\pm 20\%$
x_{mfp}^{π}	π mean free path (total rescattering probability)	$\pm 20\%$
x_{cex}^{π}	π charge exchange probability	$\pm 50\%$
x_{el}^{π}	π elastic reaction probability	$\pm 10\%$
x_{inel}^{π}	π inelastic reaction probability	$\pm 40\%$
x_{abs}^{π}	π absorption probability	$\pm 20\%$
x_{π}^{π}	π π-production probability	$\pm 20\%$

Table 9.3:   Intranuclear hadron transport systematic parameters considered in this work.

9.2 Propagating neutrino-cross section uncertainties

Unlike the propagation of hadronic simulation uncertainties (to be discussed later), which is challenging as the probability for a generated multi-particle configuration is difficult to calculate analytically, the propagation of neutrino interaction cross-section modelling uncertainties is relatively straightforward using a generic reweighing scheme less strongly tied to the details of the physics modeling. Cross section reweighing is modifying the neutrino interaction probability directly and, therefore the considerations on unitarity conservation developed in the hadron transport reweighing section are not relevant here.

The neutrino event weight, w_σ^{evt}, to account for changes in physics parameters controlling neutrino cross sections is calculated as

$$w_\sigma^{evt} = (d^n \sigma_\nu'/dK^n)/(d^n \sigma_\nu/dK^n) \tag{9.2}$$

where $d^n\sigma/dK^n$ is the nominal differential cross section for the process at hand, $d^n\sigma'/dK^n$ is the differential cross section computed using the modified input physics parameters. The differential cross section is evaluated at the kinematical phase space $\{K^n\}^3$. A critical point in implementing the cross section reweighing scheme for scattering off nuclear targets, is that the correct off-shell kinematics, as used in the original simulation, must be recreated before evaluating the differential cross sections. This is trivial as long as detailed information for the bound nucleon target has been maintained by the simulation.

Figure 9.1: JPARC neutrino beam kinematic coverage at the nd280. Cross section uncertainties of different magnitude are appropriate for different parts of the kinematic phase space.

---

[3] In GENIE, typically, the K^n kinematical phase space is $\{Q^2\}$ for CC quasi-elastic and NC elastic, $\{Q^2, W\}$ for resonance neutrino production, $\{x, y\}$ for deep inelastic scattering and coherent or diffractive meson production, $\{y\}$ for νe^- elastic scattering or inverse muon decay where Q^2 is the momentum transfer, W the hadronic invariant mass, x is Bjorken scaling variable and y the inelasticity. The choice is not significant. The differential cross section calculation can be mapped from the K^n to the $K^{n\prime}$ kinematic phase space through the Jacobian for the $K^n \to K^{n\prime}$ transformation.

Figure 9.2:   True hadronic invariant mass, W, distribution for inelastic events in nd280 (shown with the black solid line). The red hatched area shows the resonance contributions while the blue hatched area shows the contributions from the type of inelastic events dubbed in GENIE as 'transition-DIS'. The remaining contributions are coming from the 'safe-DIS' and 'low Q^2 DIS' components. Different uncertainties are associated with each component: The resonance uncertainty is of the order of 20%, while the 'transition-DIS' uncertainty is of the order of 50%. The uncertainty associated with the remaining DIS component at higher invariant masses is significantly lower (of the order of 5% at an energy of 5 GeV and lower at higher energies) and have not been included at this first iteration of deploying the reweighing tools.

9.3   Propagating hadronization and resonance decay uncertainties

Significant uncertainties exist in the modelling of neutrino-induced hadronization for neutrinos in the few-GeV energy range. In the energy range of T2K, possibly the most important hadronization uncertainty is that in the assignme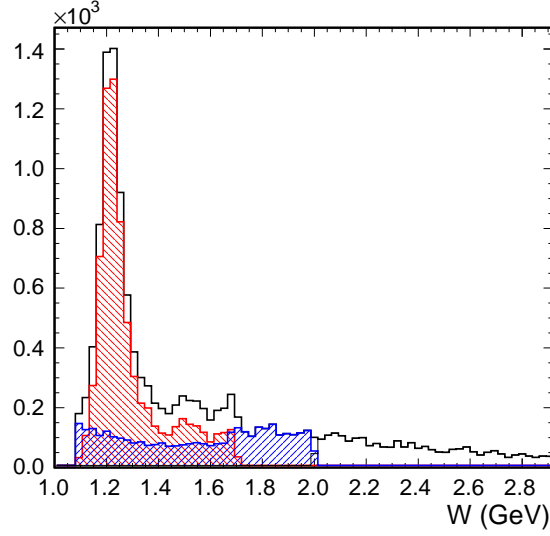nt of pion kinematics for $N\pi$ hadronic states. In GENIE, low invariant-mass hadronization is handled exclusively by the KNO-based model included in AGKY. This model uses target-fragment Feynman x (x_F) and transverse momentum (p_T^2) pdfs extracted from bubble chamber data. The pdf used for x_F has a particularly large effect on the characteristics of the generated hadronic system since a preferentially backward-going (in the hadronic CM frame) heavy target-fragment (nucleon) leads to a preferentially forward-going fast current-fragment (pion). This allows GENIE to reproduce the experimental data on the backward/forward x_F asymmetry. There is, however, experimental ambiguity on whether this backward/forward asymmetry also exists for lower-multiplicity events. The x_F and p_T^2 pdfs used in GENIE (v2.6.0) are shown in Figs. 9.3 and 9.4 respectively. They are parametrized as

$$f(x_F) = Ae^{0.5(x_F - <x_F>)^2/\sigma_{x_F}^2} \tag{9.3}$$

and

$$f(p_T^2) = Be^{-p_T^2/<p_T^2>} \tag{9.4}$$

In the reweighting scheme employed in this work, the systematic parameter $x_{AGKY}^{xF1\pi}$ ($x_{AGKY}^{pT1\pi}$) is used to tweak $<x_F>$ ($<p_T^2>$) in Eqs. 9.3 (9.4). This modifies the x_F and p_T^2 pdfs as shown in Figs. 9.5

and 9.6. Our reweighting code identifies events with a $N\pi$ hadronic state produced by the AGKY model and extracts the pion x_F, p_T^2 and the hadronic invariant mass W. For each such event, 2×10^4 $N\pi$ hadronic decays, with invariant mass W, are performed for both the default and tweaked values of the $x_{AGKY}^{xF1\pi}$ and $x_{AGKY}^{pT1\pi}$ systematic parameters. The generated decays are analysed to obtain the default and tweaked 2-dimensional pion-kinematics pdfs $f_\pi^{def}(x_F, p_T^2; W)$ and $f_\pi^{twk}(x_F, p_T^2; W)$. The event weight is computed as

$$w = f_\pi^{twk}(x_F, p_T^2; W)/f_\pi^{def}(x_F, p_T^2; W) \tag{9.5}$$

Figure 9.3: Nucleon Feynman x (x_F) pdf used in the GENIE AGKY model for generating the kinematics of 2-body $N + \pi$ primary hadronic systems.

Figure 9.4: Nucleon transverse momentum (p_T^2) pdf used in the GENIE AGKY model for generating the kinematics of 2-body $N + \pi$ primary hadronic systems.

Formation-zone uncertainties

It is well established that hadrons produced in the nuclear environment do not immediately reinteract with their full cross section. Initially quarks propagate through the nucleus with a dramatically reduced probability of interaction as they have not yet materialized as hadrons. This is implemented in GENIE as a 'free step' for all hadrons produced in deep-inelastic reactions. The 'free step', f_z, which comes from a formation time of $\tau_0 = 0.342$ fm/c, is calculated as

$$f_z = pc\tau_0/m \tag{9.6}$$

where p is the hadron momentum, m is the hadron mass and c is the speed of light.

In the reweighting scheme employed in this work, the original formation zone assigned to each hadron during event generation is recovered from the distance between the intranuclear event vertex and the hadron position as recorded at the beginning of the intranuclear cascade step. As usual, the systematic parameter x_{fz} modifies the formation zone:

$$f_z \rightarrow f_z' = f_z(1 + x_{fz} * \delta f_z/f_z) \tag{9.7}$$

Weights are calculated in a way similar to that used when modifying the hadron mean free path (see section 9.4). When the formation zone is tweaked, it alters the amount of nuclear matter through which the hadron must propagate before it exits the target nucleus. The nominal and tweaked survival probabilities are calculated as in Eq. 9.12 and a hadron weight is assigned as in Eq. 9.15. An event weight is calculated as the product of particle weights for all particles in the primary hadronic system.

Figure 9.5:   Default x_F pdf (solid line) and tweaked pdfs (dotted lines) resulting from modifying the $x_{AGKY}^{xF1\pi}$ systematic parameter by ± 1.

Figure 9.6:   Default p_T^2 pdf (solid line) and tweaked pdfs (dotted lines) resulting from modifying the $x_{AGKY}^{pT1\pi}$ systematic parameter by ± 1.

Pion angular distribution uncertainties in $\Delta \to N\pi$ decay

In general, the pion angular distribution $W_\pi(cos\theta)$ in $\Delta \to N\pi$ decay can be expressed as

$$W_\pi(cos\theta) = 1 - p(\frac{3}{2})P2(cos\theta) + p(\frac{1}{2})P2(cos\theta) \tag{9.8}$$

where θ is the pion production angle in the Δ center of mass frame with respect to the Δ angular momentum quantization axis, $P2$ is the 2nd ord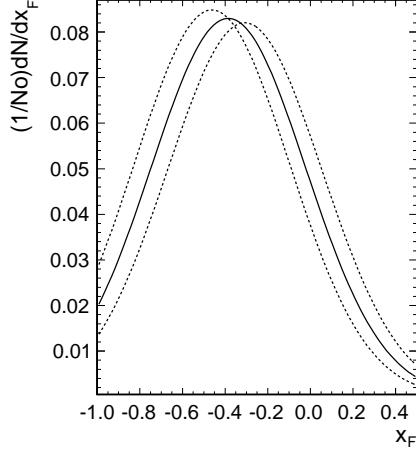er Legendre polynomial and $p(\frac{3}{2})$, $p(\frac{1}{2})$ are coefficients for each state of Δ angular momentum projection $(\frac{3}{2}, \frac{1}{2})$.

For simplicity, GENIE decays baryon resonances isotropically during event generation. Isotropy requires $p(\frac{3}{2}) = p(\frac{1}{2}) = 0.5$ but the Rein-Sehgal (RS) model predicts $p(\frac{3}{2}) = 0.75$ and $p(\frac{1}{2}) = 0.25$. In this work, we employ a reweighting scheme to quantify the uncertainty over the π angular momentum distribution. A measure of this uncertainty is taken to be the difference between the isotropic and RS predictions. The reweighting code identifies events with a $\Delta + +(1232)$ decaying to a $N\pi$ state. In a nuclear environment, where hadronic rescattering is possible, the $N\pi$ state produced may not necessarily be the final hadronic state. Once the event is identified, the daughter π and parent Δ 4-momenta in the LAB frame are used to calculate the π 4-momentum in the Δ center-of-mass frame. Then the π production angle θ is calculated with respect to an arbitrarily-defined angular momentum quantization axis ($+z$). Let $W_\pi^{iso}(cos\theta)$ and $W_\pi^{RS}(cos\theta)$ be, respectively, the π production-angle probability density for the isotropic and RS cases, computed from Eq. 9.8. An event weight is constructed as follows:

$$w = \left(x_{\theta_\pi}^{\Delta \to \pi N} W_\pi^{RS}(cos\theta) + (1 - x_{\theta_\pi}^{\Delta \to \pi N})W_\pi^{iso}(cos\theta) \right)/W_\pi^{iso}(cos\theta) \tag{9.9}$$

where $x_{\theta_\pi}^{\Delta \to \pi N}$ is the corresponding nuisance parameter. For $x_{\theta_\pi}^{\Delta \to \pi N} = 0$, all weights are equal to 1, i.e. this setting corresponds to the default case of isotropic Δ decays. For $x_{\theta_\pi}^{\Delta \to \pi N} = 1$, the calculated weight is equal to $W_\pi^{RS}(cos\theta)/W_\pi^{iso}(cos\theta)$; this reweights the isotropic pion angular distributions to those predicted by RS. For values of $x_{\theta_\pi}^{\Delta \to \pi N}$ between 0 and 1, there is a linear transition between the isotropic and RS angular distributions.

Figure 9.7: Angular distributions for pions from $\Delta \to \pi N$ decays for various values of the $x_{\theta_\pi}^{\Delta \to \pi N}$ nuisance parameter between -1 and 1. The isotropic distribution (GENIE simulation default) is obtained for $x_{\theta_\pi}^{\Delta \to \pi N} = 0$. The RS model prediction is obtained for $x_{\theta_\pi}^{\Delta \to \pi N} = 1$.

Branching ratio uncertainties

Reweighting events to account for changes in decay branching ratios is straightforward. It is important to ensure that the sum of all branching ratios for each unstable particle remains unchanged.

Let x_d^p be a nuisance parameter which affects the branching ratio f_d^p for the decay channel d which is available to particle p. As usual in this work, the nuisance parameter modifies the corresponding physics parameter (branching ratio) as $f_d^p \to f_d'^p = f_d^p(1 + x_d^p * \sigma_{f_d^p}/f_d^p)$, where $\sigma_{f_d^p}$ is the uncertainty in the branching ratio. In the reweighting scheme employed in this work, if *any* branching ratio of a given particle is tweaked, then *all* decays of that particle are reweighted so that the sum of all branching ratios remains unchanged. If x_d^p is tweaked, the weight for decay d is computed as follows:

$$w_d^p = \frac{f_d'^p}{f_d^p} \tag{9.10}$$

For every other decay $d\prime \neq d$ of that particle a weight is computed as:

$$w_{d\prime}^p = \frac{1 - f_d'^p}{1 - f_d^p} \tag{9.11}$$

The above weight is assigned to a single unstable particle for which the branching ratio of any decay channel has been altered. The event weight is the product of weights for all such particles.

9.4 Propagating intranuclear hadron transport uncertainties

Hadrons produced in the nuclear environment may rescatter on their way out of the nucleus, and these reinteractions significantly modify the observable distributions. The simulated effect of hadronic reinteractions is illustrated in Tab. 9.4 and Fig. 9.8. The sensitivity of a particular experiment to intranuclear rescattering depends strongly on the detector technology, the energy range of the neutrinos, and the physics measurement being made.

Figure 9.8: Kinetic energy spectrum of final state and primary (before rescattering) π^+ produced in $\nu_\mu Fe^{56}$ interactions at 1 GeV.

Neutrino generators typically use intranuclear cascade simulations to handle the propagation of hadronic multi-particle states. At each simulation step a large number of outcomes is accessible with the probabilities of those outcomes being conditional upon the hadron transport history up to that point. The complexity of intranuclear hadron transport makes it difficult to evaluate the probability for a generated multi-particle final state, given a primary hadronic multi-particle system, without resorting to a Monte Carlo method. Subsequently, is not possible to evaluate how that probability ought to be modified in response to changes in the fundamental physics inputs. As a result it is generally not possible to build comprehensive reweighing schemes for intranuclear hadron-transport simulations.

In this regard GENIE's INTRANUKE/hA model is unique by virtue of the simplicity of the simulation while, at the same time, it exhibiting very reliable aspects by being anchored to key hadron-nucleon and hadron-nucleus data. Its simplicity allows a rather straightforward probability estimate for the generated final state making it amenable to reweighing. A full systematic analysis of the model is therefore possible making it a unique tool in the analysis of neutrino data. The event reweighing strategy to be presented here is *specific* to GENIE's INTRANUKE/hA model. The current reweighing implementation has been tied to the physics choices made in the GENIE v2.4.0[4].

---

[4] The validity of the current reweighing implementation in future versions of GENIE is dependent upon the IN-TRANUKE/hA changes that may be installed. The T2KReWeight package will always be updated and kept in sync with GENIE. In case of important updates a follow-up internal note will be posted.

Final-	Primary Hadronic System									
State	$0\pi X$	$1\pi^0 X$	$1\pi^+ X$	$1\pi^- X$	$2\pi^0 X$	$2\pi^+ X$	$2\pi^- X$	$\pi^0\pi^+ X$	$\pi^0\pi^- X$	$\pi^+\pi^- X$
$0\pi X$	**293446**	12710	22033	3038	113	51	5	350	57	193
$1\pi^0 X$	1744	**44643**	3836	491	1002	25	1	1622	307	59
$1\pi^+ X$	2590	1065	**82459**	23	14	660	0	1746	5	997
$1\pi^- X$	298	1127	1	**12090**	16	0	46	34	318	1001
$2\pi^0 X$	0	0	0	0	**2761**	2	0	260	40	7
$2\pi^+ X$	57	5	411	0	1	**1999**	0	136	0	12
$2\pi^- X$	0	0	0	1	0	0	**134**	0	31	0
$\pi^0\pi^+ X$	412	869	1128	232	109	106	0	**9837**	15	183
$\pi^0\pi^- X$	0	0	1	0	73	0	8	5	**1808**	154
$\pi^+\pi^- X$	799	7	10	65	0	0	0	139	20	**5643**

Table 9.4: Occupancy of primary and final state hadronic systems for interactions off O^{16} computed with GENIE v2.4.0. The off-diagonal elements illustrate and quantify the topology changing effect of intranuclear rescattering.

Any intranuclear hadron-transport reweighing strategy should, by *virtue of construction*, have no effect on the inclusive leptonic distributions of the reweighted sample, as illustrated in Fig. 9.9. In this paper we will be referring to that probability conservation condition as the *'unitarity constraint'*. We emphasize the fact that the constraint needs to hold only for unselected samples. It does not need to hold for selected samples, where the normalization is expected to vary due to the effect of the cut acceptance.

The unitarity constraint is obviously very difficult to satisfy by virtue of construction and has had a significant role in determining the reweighing strategy. Additionally, the constraint played an important role in validating the reweighing scheme and in matching exactly all physics assumptions of the original simulation. The most profound effect of weighting artifacts is to cause the unitarity constraint to be violated. We will revisit the issue of the unitarity constraint in later sections and, particularly, on the discussion of the reweighing validation.

Figure 9.9:   Consider the effect of modifying the intranuclear hadron-transport physics (affecting the particles within the *box*) from the perspective of an observer who is blind to the hadronic system emerging from the nucleus and measures only the primary lepton. One can easily assert that, from the perspective of that observer, the hadron-transport reweighing scheme should have no effect on the leptonic system characteristics of samples that have not been selected for hadronic system characteristics. The event weights must cancel each other so as the sum of weights is conserved, therefore maintaining the sample normalization. We will be referring to that condition as the *'unitarity constraint'*. As we will see in the reweighing validation section, the scheme discussed in this note satisfies the unitarity constraint, by *virtue of construction*, to better than 1 part in 5000.

In the reweighing strategy developed here we consider 2 kinds of physics uncertainties:

- Uncertainties in the total rescattering probability for hadrons within the target nucleus.

- Uncertainties in the relative probability of rescattering modes available to each hadron once it interacts.

These physics uncertainties are considered separately for nucleons and pions. The determination of simulation parameters linked with these physics uncertainties and the prescription for calculating event weights to account for variations in these parameters is discussed next.

Reweighting the rescattering rate

During event generation, for each hadron being propagated within the nuclear environment its rescattering probability, P_{rescat}^h (or, equivalently the survival probability, P_{surv}^h) is calculated as

$$P_{rescat}^h = 1 - P_{surv}^h = 1 - \int e^{-r/\lambda^h(\vec{r}, h, E_h)} dr \tag{9.12}$$

where λ^h is the mean free path and the integral is evaluated along the hadron trajectory. The mean free path is a function of the hadron type, h, the hadron energy, E_h, and its position, $\vec{r}$, within the target nucleus and is computed as

$$\lambda^h = 1/(\rho_{nucl}(r) * \sigma^{hN}(E_h)) \tag{9.13}$$

where $\rho_{nucl}(r)$ is the nuclear density profile and $\sigma^{hN}(E_h)$ the corresponding hadron-nucleon total cross section.

During the reweighing procedure, using the positions and 4-momenta of the simulated primary hadronic system particles (that is the hadrons emerging from the primary interaction vertex before any intranuclear rescattering ever took place) we calculate the exact same hadron survival probabilities as in the original simulation. In doing so we match exactly the physics choices of the hadron transport simulation code so as to avoid weighting artifacts. More importantly:

- The reweighing code accesses the same hadron-nucleon cross section and nuclear density profile functions as the simulation code. The nuclear density profiles for ^{12}C, ^{16}O and ^{56}Fe and the nucleon-nucleon and pion-nucleon cross sections used by INTRANUKE/hA in GENIE v2.4.0 are shown in Figs. 9.10 and 9.11 respectively.

- The hadrons are being transported in steps of 0.05 fm as in the original simulation.

- Each hadron is traced till it reaches a distance of $r = N * R_{nucl} = N * R_0 * A^{1/3}$, where $R_0 = 1.4$ fm and $N = 3.0$. This allows taking into account the effect the nuclear density distribution tail has on the hadron survival probability. (For example, the nuclear radius, R_{nucl} for C^{12}, O^{16} and Fe^{56} is 3.20 fm, 3.53 fm and 5.36 fm respectively. The reweighing, as the actual simulation code, integrates Eq. 9.12 for distances up to 9.62 fm, 10.58 fm and 16.07 fm respectively. Compare these values with the nuclear density profiles shown in Fig. 9.10.)

- The nuclear density distribution through which the hadron is tracked is increased by $n * \lambda_B$, where λ_B is the de Broglie wave-length of the hadron and n is a tunable parameter (in GENIE v2.4.0, INTRANUKE/hA uses $n = 1$ for nucleons and $n = 0.5$ for pions). As explained earlier, this empirical approach is an important feature of the INTRANUKE/hA mean free path tuning, accounting for the effects of wave-like processes to the hadron survival probability which are typically not well described within the context of an INC model. The reweighing code matches that feature so as to emulate the hadron survival probabilities calculated during event generation. The effect on the nuclear density profile is shown in Fig. 9.12.

The reweighing scheme allows the mean free path, λ^h, for a hadron type h to be modified in terms of its corresponding error, $\delta\lambda^h$:

$$\lambda^h \rightarrow \lambda^{h\prime} = \lambda^h(1 + x^h_{mfp} * \delta\lambda^h/\lambda^h) \tag{9.14}$$

where $\lambda^{h\prime}$ is the modified mean free path and x_{mfp} is a tweaking knob. Then, by re-evaluating the integral in Eq. 9.12, we are able to compute the hadron survival probabilities that the simulation code would have computed, had it been using the modified mean free path. The nominal, P^h_{surv}, and tweaked, $P^{h\prime}_{surv}$, survival probabilities can be used to calculate the weight that accounts for that change in the hadron mean free path. The choice of how to weight each hadron depends critically on its intranuclear transport history. Consider the case illustrated in Fig. 9.13 where a neutrino event has 2 primary hadrons, h_1 and h_2, one of which (h_1) re-interacts while the other (h_2) escapes. Had the mean free path been larger than the one used in the simulation (and therefore, had the the interaction probability been lower) then h_1's history would have been more unlikely while, on the other hand, h_2's history would have been more likely. Therefore, in order to account for an increase in mean free path, h_1 has to be weighted down while h_2 has to be weighted up (and vice versa for a mean free path decrease). The desired qualitative behavior of single-hadron weights in response to mean free path changes is summarized in Tab. 9.5. The following weighting function exhibits the desired qualitative characteristics:

$$w^h_{mfp} = \begin{cases} \dfrac{1-P^{h\prime}_{surv}}{1-P^h_{surv}} & \text{if } h\text{re-interacts} \\[3ex] \dfrac{P^{h\prime}_{surv}}{P^h_{surv}} & \text{if } h\text{escapes} \end{cases} \tag{9.15}$$

where P^h_{surv} is the hadron survival probability corresponding to mean free path λ^h and $P^{h\prime}_{surv}$ is the hadron survival probability corresponding to the tweaked mean free path $\lambda^{h\prime}$.

Figure 9.10: Nuclear density profiles for C^{12}, O^{16} and Fe^{56}.

Figure 9.11: The nucleon-nucleon (dashed line) and pion-nucleon (solid line) cross sections used in IN-TRANUKE/hA (GENIE v2.4.0) for determining the hadron mean free path.

Figure 9.12: Nuclear density profiles for Fe^{56} 'stretched' by the de-Broglie wave-length corresponding to hadrons with a momentum of 0.1 GeV, 0.5 GeV and 1.0 GeV. The default nuclear density distribution is also shown.

Figure 9.13: An example event with two primary hadrons, h_1 and h_2, one of which (h_1) re-interacts within the target nucleus while the other escapes (h_2). See text for a description of the weights to be assigned to each hadron if the mean free path has been tweaked.

λ^h change	P^h_{rescat} change	Weight (hadrons interacting)	Weight (hadrons escaping)
$\Uparrow$	$\Downarrow$	$\Downarrow$	$\Uparrow$
$\Downarrow$	$\Uparrow$	$\Uparrow$	$\Downarrow$

Table 9.5: The intended qualitative behavior of hadron weights in response to mean free path, λ^h, changes depending on whether the simulated hadron had been rescattered or escaped. Had the mean free path been larger in reality than the one used in the simulation (and therefore, had the the interaction probability, P^h_{rescat}, been lower) then rescattered hadrons would have been over-represented in the generated sample and they would need to be weighted-down to match reality, while escaping hadrons would have been under-represented and they would need to be weighted-up. Vice versa for a mean free path decrease. See text for description of the hadron weighting functions.

Reweighting the rescattering fates

Once INTRANUKE/hA determines that a particular hadron is to be rescattered, then a host of scattering modes are available to it. We will be referring to these scattering modes as the *hadron fates*. Many fates are considered for both pions and nucleons. The fates considered here are: elastic, inelastic, charge exchange[5], absorption[6], and pion production. Each such fate may include many actual rescattering channels [7].

In order to calculate the probability of each fate INTRANUKE/hA, being an effective data-driven hadron transport MC, switches to a more macroscopic description of hadron rescattering: Rather than building everything up from hadron-nucleon cross sections, at this point in event simulation, INTRANUKE/hA determines the probability for each fate using built-in hadron-nucleus cross sections coming primarily from data. The probability for a hadron fate f is

$$P^h_f = \sigma^{hA}_f / \sigma^{hA}_{total} \tag{9.16}$$

where σ^{hA}_f is the hadron-nucleus cross section for that particular fate and σ^{hA}_{total} is the total hadron-nucleus cross section. The calculated probabilities are conditional upon a hadron being rescattered and the sum of these probabilities over all possible fates should always add up to 1. The default probability fractions for pions and nucleons in INTRANUKE/hA (GENIE v2.4.0) are shown in Fig. 9.14 and 9.15.

The generation strategy leads to a conceptually simple and technically straight-forward fate reweighing strategy: The hadron-nucleus cross section for a particular fate may be modified in terms of its corresponding error, $\delta\sigma^{hA}_f$ as in:

$$\sigma^{hA}_f \rightarrow \sigma'^{hA}_f = \sigma^{hA}_f (1 + x^h_f * \delta\sigma^{hA}_f / \sigma^{hA}_f) \tag{9.17}$$

where x_f is a fate tweaking knob.

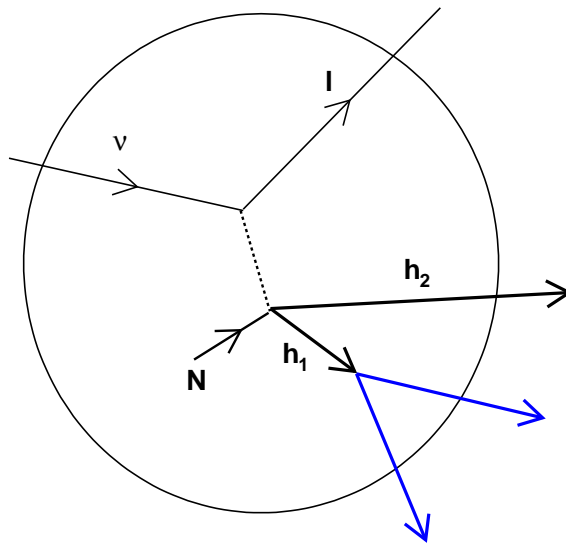It follows that the single-hadron fate weight is

$$w^h_{fate} = \sum_f \delta_{f;f'} * x^h_f * \delta\sigma^{hA}_f / \sigma^{hA}_f \tag{9.18}$$

---

[5] Only single charge exchange is considered

[6] Followed by emission of 2 or more nucleons with no pions in the final state. The term *'absorption'* is usually used for pions while the term *'multi-nucleon knock-out'* is used for nucleons. Here, for simplicity and in the interest of having common fate names for both pions and nucleons we will be using the term *'absorption'* for both.

[7] For example, the *'pion absorption'* fate includes rescattering modes with any of the np, pp, npp, nnp, nnpp final states

where f runs over all possible fates {elastic, inelastic, charge exchange, absorption, pion production}, f' is the actual fate for that hadron as it was determined during the simulation and $\delta_{f;f'}$ is a factor which is 1 if $f = f'$ and 0 otherwise.

Not all 5 hadron fates may be tweaked simultaneously. Since the sum of all fractions should add up to 1 then, at most, at most 4 out of the 5 fates may be tweaked directly. The fates not tweaked directly (*cushion* terms) are adjusted automatically to conserve the sum. The choice of which fates act as a cushion terms is configurable.

In Fig. 9.16 we show the tweaked pion fate fraction (dashed lines) obtained by simultaneously increasing the pion production, absorption, charge exchange and inelastic cross sections by 10%. In this example the elastic component is being used as a cushion term absorbing the changes in all other terms so as to maintain the total probability. The default pion fate fractions (solid lines) are superimposed for reference.

Figure 9.14: The default fate fractions for rescattered pions in INTRANUKE/hA (GENIE v2.4.0). The area that corresponds to each pion fate represents the probability for that fate as a function of the pion kinetic energy. The probabilities shown here conditional upon the pion interacting so they always add up to 1.

Computing event weights

The scheme outlined above, provides a detailed prescription for calculating single-hadron weights so as to take into account the effect that modified hadron-nucleon and hadron-nucleus cross sections would have had on that hadron (w^h_{mfp} and w^h_{fate} respectively). The total single-hadron weight is

$$w^h = w^h_{mfp} * w^h_{fate} \tag{9.19}$$

Figure 9.15: The default fate fractions for rescattered nucleons in INTRANUKE/hA (GENIE v2.4.0). The area that corresponds to each nucleon fate represents the probability for that fate as a function of the nucleon kinetic energy. The probabilities shown here conditional upon the nucleon interacting so they always add up to 1.

The corresponding hadron transport (HT) related weight for a neutrino interaction event, w_{HT}^{evt}, is, obviously, the product of single-hadron weights

$$w_{HT}^{evt} = \prod_j w_j^h \tag{9.20}$$

where the index j runs over all the primary hadronic system particles in the event.

Computing penalty terms

A penalty term can easily be calculated from the physics tweaking knobs which can be included as nuisance parameters in physics fits. The penalty has components, penalizing deviations from the default total rescattering rate and from the default fractions of rescattering modes. It can be written as

$$\chi_{penalty}^2 = \sum_{h=\pi,N} \{(x_{mfp}^h)^2 + \sum_{f \neq fc} (x_f^h)^2 + (\widehat{x_{fc}^h})^2\} \tag{9.21}$$

where the $x's$ correspond to mean free path and fate tweaking knobs for pions and nucleons The sum over fates, f, excludes the cushion term, fc, which is added separately. The reason is technical: All directly tweaked hadron-nucleus cross sections are tweaked in units of their own (typically hadron energy-dependent) uncertainty, therefore having a corresponding contribution to penalty term which is energy independent. The change in the cushion term, being forced to absorb the other changes, is not well

Figure 9.16: The default (solid lines) and tweaked (dashed lines) pion fate fractions. The tweaked pion fate fractions are shown for a case where the pion production, absorption, charge exchange and inelastic cross sections have been increased by 10%. Here it is the elastic cross section term that is being used as the cushion term. See text for details.

defined in terms of its own uncertainty. Therefore, its contribution in the penalty term, $\widehat{x_{fc}^{h}}^{2}$, is averaged over the hadron energy range.

Unitarity expectations

This section demonstrates why both the intranuclear reweighting schemes presented earlier are expected to maintain unitarity. In general, when reweighting an event, we multiply by a weight w

$$ w = \frac{P'}{P}. \tag{9.22} $$

where P and P' are the probabilities for getting that event[8], for the nominal and tweaked cases respectively, and they depend on the particular event being reweighted.

When describing processes where multiple discrete outcomes are possible then the analytical form of the above probabilities will change depending on the outcome. An example of this is the case of mean free path (rescattering rate) reweighting where the fate of an event can be divided into two categories: Those that rescattered and those that escaped the nucleus. The two forms of P in this case are,

$$ P_{rescat} = 1 - e^{\frac{-x}{\lambda}} \tag{9.23} $$

and

---

[8]In this section an event is defined as the transport of a single hadron.

$$P_{surv} = e^{\frac{-x}{\lambda}}. \tag{9.24}$$

Thus a hadron that rescattered will receive a weight, reflecting a change in mean free path of $\lambda \to \lambda'$, of

$$w_{rescat} = \frac{1 - e^{\frac{-x}{\lambda'}}}{1 - e^{\frac{-x}{\lambda}}} \tag{9.25}$$

whereas one that escaped the nucleus will get a weight

$$w_{surv} = \frac{e^{\frac{-x}{\lambda'}}}{e^{\frac{-x}{\lambda}}} \tag{9.26}$$

Take the general case where there are n possible outcomes and where the i'th outcome occurs with a probability P_i. For a set of N_{tot} events one expects

$$N_i = N_{tot} \times \frac{P_i}{\sum_{j=1}^{n} P_j} \tag{9.27}$$

events corresponding to the i'th outcome.

Now consider reweighting all N_{tot} events. Events corresponding to the i'th outcome get weighted by w_i so that the after reweighting the number of events for the i'th outcome is given by

$$N_i' = w_i \times N_i. \tag{9.28}$$

Note that Eq. 9.28 holds only if we consider just the functional dependance of the weights on the weighting parameters[9]. The number of events in the new reweighted sample is given by

$$
\begin{aligned}
N_{tot}' &= \sum_{j=1}^{N_{tot}} w_j^{evt} \\
&= \sum_{i=1}^{n} w_i^{outcome} \times N_i \\
&= \sum_{i=1}^{n} \frac{P_i'}{P_i} \times N_i.
\end{aligned}
$$

Substituting Eq. 9.27 we get,

$$N_{tot}' = N_{tot} \times \frac{\sum_{i=1}^{n} P_i'}{\sum_{j=1}^{n} P_j}.$$

So if

$$\sum_{i=1}^{n} P_i = \sum_{i=1}^{n} P_i' \tag{9.29}$$

then $N_{tot}' = N_{tot}$ and unitarity is conserved.

In the case of rescattering,

---

[9]We neglect any functional dependance on kinematical quantities. This is a valid assumption if the density of events, defined as the number in a given volume of kinematical phase space, is high enough such that a statistically significant number of neighboring events cover a small enough volume in the kinematical phase space over which the effect of the variation in kinematical quantities is negligible.

$$
\begin{aligned}
\sum_{i=1}^{n} P_i &= P_{rescat} + P_{surv} \\
&= 1 - e^{\frac{-x}{\lambda}} + e^{\frac{-x}{\lambda}} \\
&= 1 - e^{\frac{-x}{\lambda'}} + e^{\frac{-x}{\lambda'}} \\
&= P'_{rescat} + P'_{surv} \\
&= \sum_{i=1}^{n} P'_i
\end{aligned}
$$

So for the rescattering scheme we expect unitarity to be a built in feature. This is also true for the fate reweighting where the cushion term ensures Eq. 9.29 is satisfied. It is worth highlighting that the unitarity constraint is sensitive to any differences between the generator and the reweighting scheme. This is also why a particular implementation of a reweighting scheme is not generator agnostic.

9.5   Event reweighting applications

9.5.1   Built-in applications

The *grwght1scan* utility

Name

grwght1scan - Generates weights given an input GHEP event file and for a given systematic parameter (supported by the ReWeight package). It outputs a ROOT file containing a tree with an entry for every input event. Each such tree entry contains a *TArrayF* of all computed weights and a *TArrayF* of all used tweak dial values.

Source and build options

The source code for this application is in '`$GENIE`/*src/support/rwght/gRwght1Scan.cxx*'.
To enable this application (and, also, to build the ReWeight package library) add '`--enable-rwght`' during the GENIE build configuration step.

Synopsis

```
grwght1scan
   -f input_filename
   [-n number_of_events]
   -s systematic_name
   -t number_of_tweaking_diall_values
   [-p neutrino_codes]
```

where [] is an optional argument.

Description

The following options are available:

-f Specifies an input GHEP event file.

-n Specifies the number of events to process.

This is an optional argument. By default GENIE will process all events.

-s Specifies the name of the systematic param to tweak.

-t Specifies the number of the systematic parameter tweaking dial values between -1 and 1.

Note: This must be an odd number so as to include al; -1, 0 and 1. If it is an even number then it will be incremented by 1.

-p If set, specifies which neutrino species to reweight.

This is an optional argument. By default GENIE will reweight all neutrino species. The expected input is a comma separated list of PDG codes.

Examples

9.5.2 Writing a new reweighting application

Writing a new reweighting application is relatively trivial. The built-in applications described above can be used as a template and be modified accordingly. A *GReWeight* object provides an interface between the user and the GENIE event reweighting objects (weight calculators). *GReWeight* holds both a list of weight calculators (*GReWeightI* subclasses), each one referred-to by a user-specified name, and a set of tweaked systematic parameters (*GSystSet* object).

Typically, in an event reweighting application one would have to include at least the following steps:

- Instantiate a *GReWeight* object and add to it a set of concrete weight calculators. For example (modify accordingly by adding / removing weight calculators from this list):

```
GReWeight rw;

rw.AdoptWghtCalc( "xsec_ccqe",       new GReWeightNuXSecCCQE      );
rw.AdoptWghtCalc( "xsec_ccqe_vec",   new GReWeightNuXSecCCQEvec   );
rw.AdoptWghtCalc( "xsec_ccres",      new GReWeightNuXSecCCRES     );
rw.AdoptWghtCalc( "xsec_ncres",      new GReWeightNuXSecNCRES     );
rw.AdoptWghtCalc( "xsec_nonresbkg",  new GReWeightNonResonanceBkg );
rw.AdoptWghtCalc( "xsec_dis",        new GReWeightNuXSecDIS       );
rw.AdoptWghtCalc( "xsec_coh",        new GReWeightNuXSecCOH       );
rw.AdoptWghtCalc( "nuclear_qe",      new GReWeightFGM             );
rw.AdoptWghtCalc( "nuclear_dis",     new GReWeightDISNuclMod      );
rw.AdoptWghtCalc( "hadro_res_decay", new GReWeightResonanceDecay  );
rw.AdoptWghtCalc( "hadro_fzone",     new GReWeightFZone           );
rw.AdoptWghtCalc( "hadro_intranuke", new GReWeightINuke           );
rw.AdoptWghtCalc( "hadro_agky",      new GReWeightAGKY            );
```

- Retrieve and fine-tune weight calculators. This is an optional step. Each calculator is retrieved from *GReWeight* using the user-defined name specified in the previous step. Fine-tuning methods

are specific to each weight calculator, so please refer to the documentation for each individual calculator. For example, to disable ν_e, $\bar{\nu}_e$ and $\bar{\nu}_\mu$ reweighting in *GReWeightNuXSecCCQE* stored with the "xsec_calc" name, type:

```
GReWeightNuXSecCCQE * rwccqe =
    dynamic_cast<GReWeightNuXSecCCQE *> (
      rw.WghtCalc("xsec_ccqe"));
rwccqe -> RewNue    (false);
rwccqe -> RewNuebar (false);
rwccqe -> RewNumubar(false);
```

- Get the *GSystSet* object held by *GReWeight* and tweak all systematic params you wish to consider (complete list to be found in '$GENIE/src/ReWeight/GSyst.h'). What you are actually setting is the value d of a tweaking dial (default value: 0) which modifies a corresponding physics parameter p as $p \to p' = p \times (1 + d \times (dp/p))$. Setting a tweaking dial to $+/-1$ modifies a physics quantity by $+/- 1 \sigma$ respectivelly. The default fractional errors dp/p are defined in *GSystUncertainty* and can be overriden. The following example sets non-default values to a series of systematics parameters handled by the weight calculators included in the previous step. After all parameters have been tweaked, invoke *GReWeight::Reconfigure()* so that tweaked parameters can be propagated across GENIE. You probably need to be setting these parameters and reconfiguring GENIE inside a 'parameter loop' or a 'minimization function'.

```
GSystSet & syst = rw.Systematics();

syst.Set(kXSecTwkDial_NormCCQE,        +1.0);
syst.Set(kXSecTwkDial_MaCCQEshape,     +1.0);
syst.Set(kXSecTwkDial_NormCCRES,       -1.0);
syst.Set(kXSecTwkDial_VecFFCCQEshape,  -1.0);
syst.Set(kXSecTwkDial_MaCCRESshape,    -1.0);
syst.Set(kXSecTwkDial_MvCCRESshape,    +0.5);
syst.Set(kXSecTwkDial_NormNCRES,       +1.0);
syst.Set(kXSecTwkDial_MaNCRESshape,    -0.7);
syst.Set(kXSecTwkDial_MvNCRESshape,    +0.3);
syst.Set(kXSecTwkDial_RvpCC1pi,        +0.5);
syst.Set(kXSecTwkDial_RvnCC1pi,        +0.5);
syst.Set(kXSecTwkDial_MaCOHpi,         -0.5);
syst.Set(kINukeTwkDial_MFP_pi,         +1.0);
syst.Set(kINukeTwkDial_MFP_N,          -1.0);
syst.Set(kINukeTwkDial_FrPiProd_pi,    -0.7);
syst.Set(kHadrAGKYTwkDial_xF1pi,       -1.0);
syst.Set(kHadrAGKYTwkDial_pT1pi,       +1.0);
syst.Set(kHadrNuclTwkDial_FormZone,    +1.0);
syst.Set(kRDcyTwkDial_Theta_Delta2Npi, +1.0);

rw.Reconfigure();
```

- Calculate an event weight by invoking *GReWeight::CalcWeight()*. The function expects an *EventRecord* object as input. The return value is the calculated weight and is computed as the product of the weights computed by all included weight calculators for the current set of systematics / tweaking dial values stored in *GSystSet*. You can also calculate a penalty factor, $\chi^2_{penalty}$, for the current set of systematic tweaking dial values by invoking *GReWeight::CalcChisq()*.

Important notes The reweighting package includes a large number of weight calculators handling a large numbers of systematic parameters. Alternative reweighting schemes may exist for the same systematic parameter. It is the user's responsibilty to make sure that all parameters tweaked in *GSystSet* are handled by exactly one weight calculator added via *GReWeight::AdoptWeightCalc()*. Additionally, certain systematic parameters should not be combined together. For example, you should tweak either kXSecTwkDial_MaCCQE (tweakes the axial mass used in the $CCQE$ cross section model and allows it to change both the shape and the normalization of the output $d\sigma/dQ^2$ distribution at fixed energy), OR kXSecTwkDial_NormCCQE and kXSecTwkDial_MaCCQEshape (where the normalization and shape-effects have been separated) and you should never mix them all together. All in all, a good understanding of the effect of each included systematic parameter and weight calculator (see this Chapter) is imperative in order to get meaningfull results.

9.6 Adding a new event reweighting class

A large number of event reweighting classes (weight calculators) exist within GENIE and can serve as examples. One can easily add a new concrete weight calculator which can be integrated with the existing reweighting framework. This new calculator should subclass *GReWeightI* and implement, at least, the following methods:

- '*bool IsHandled(genie::GSyst_t syst)*' :
  Declare whether the weight calculator handles the input systematic parameter.

- '*void SetSystematic(genie::GSyst_t syst, double val)*' :
  Update the current value for the specified systematic parameter.

- '*void Reset(void)*' :
  Set all handled systematic parameters to default values.

- '*void Reconfigure(void)*' :
  Propagate updated systematic parameter values to actual GENIE MC code, if needed.

- '*double CalcWeight(const genie::EventRecord & event)*' :
  Calculate a weight for the input event using the current values of all handled systematic parameters.

- '*double CalcChisq(void)*' :
  Calculate a penalty factor for the current deviation of all handled systematic params from their default values.

This is the minimum set of methods required by GENIE itself. More methods, specific to each weight calculator, can be added and used in the user's event reweighting application so as to fine-tune the behaviour of each calculator.

Note that if you are adding a weight calculator to quantify the effect of a new systematic parameter, one which is not already included in '$GENIE/*src/ReWeight/GSyst.h*', then also you need to:

- add the new parameter in '$GENIE/*src/ReWeight/GSyst.h*', and

- define a default $1\,\sigma$ error in '$GENIE/*src/ReWeight/GSystUncertainty.cxx*'.

Appendix A

Copyright Notice and Citation Guidelines

A.1 Guidelines for Fair Academic Use

The authors of GENIE endorse the MCNET guidelines[1] for fair academic use. In particular, users are invited to consider which GENIE components are important for a particular analysis and cire them, in addition to the main references.

A.2 Main references

All derivative works should cite:

C.Andreopoulos et al., 'The GENIE Neutrino Monte Carlo Generator', Nucl.Instrum.Meth. A614:87-104,2010.

---

[1]Full text may be found at *http://www.montecarlonet.org/GUIDELINES*

Corresponding BibTeX entry:

```
@Article{Andreopoulos:2009rq,
   author    = "Andreopoulos, C. and others",
   title     = "{The GENIE Neutrino Monte Carlo Generator}",
   journal   = "Nucl. Instrum. Meth.",
   volume    = "A614",
   year      = "2010",
   pages     = "87-104",
   eprint    = "0905.2517",
   archivePrefix = "arXiv",
   primaryClass  =  "hep-ph",
   doi       = "10.1016/j.nima.2009.12.009",
   SLACcitation  = "%%CITATION = 0905.2517;%%"
}
```

Appendix B

Special Topics, FAQs and Troubleshooting

B.1   Installation / Versioning

B.1.1   Making user-code conditional on the GENIE version

User-code can be made conditional upon the GENIE version number, in similar way as with ROOT, by including '$GENIE/*src/Conventions/GVersion.h*'. This header file is automatically generated during the GENIE installation. If, for example, one wishes to do something different before / after version 2.16.22, then simply type:

```
#if __GENIE_RELEASE_CODE__ >= GRELCODE(2,16,22)
...
<your code here>
...
#else
...
<your code here>
...
#endif
```

B.2   Software framework

B.2.1   Calling GENIE algorithms directly

GENIE provides a host of event generation applications and utilities and most users will only ever interact with these. It is only for the most advanced GENIE uses-cases that one may need to access and run algorithms directly. This is typically a 4-step process, as outlined below:

1. Get an algorithm factory (*AlgFactory*) instance. The algorithm factory provides access to configured instances of all GENIE algorithms.

   ```
   AlgFactory * algf = AlgFactory::Instance();
   ```

2. Request a concrete algorithm from the factory. Each algorithm is uniquely specified by its name and the name of its configuration parameter set.

```
const Algorithm * alg_base = algf->GetAlgorithm(``name'', ``config'');
```

3. Type-cast *Algorithm* to the specific algorithmic interface (*XyzI*) being implemented. For example, for cross section algorithms type-cast to *XSecAlgorithmI*, for hadronization models to *HadronizationModelI*, for strucrure function models to *DISStructureFuncModelI*, for event generation modules to *EventRecordVisitorI* etc (please consult the GENIE doxygen code reference for a full list of possibilities).

```
const XzyI * alg = dynamic_cast<const XyzI *>(alg_base);
```

4. Prepare the algorithm inputs and run it (please consult GENIE doxygen code reference for documentation on each algorithmic interface).

Example 1

The following example shows how to get the Rein-Sehgal resonance neutrino-production model, calculate the differential cross section $d^2\sigma/dW\,dQ^2$ for $\nu_\mu + n$ (bound in Fe^{56}) $\to \mu^- + \mathrm{P}11(1440)$ at $E_\nu = 2.4\ GeV$, $W = 1.35\ GeV$, $Q^2 = 1.1\ GeV^2$ and then calculate the integrated cross section at the same energy:

```
{
 ...

 // get the algorithm factory
 AlgFactory * algf = AlgFactory::Instance();

 // get the cross section algorithm
 const Algorithm * algbase =
    algf->GetAlgorithm("genie::ReinSeghalRESPXSec", "Default"));
 const XSecAlgorithmI * xsec_model =
    dynamic_cast<const XSecAlgorithmI *> (algbase);

 // prepare the cross section algorithm inputs
 Interaction * interaction
        = Interaction::RESCC(kPdgTgtFe56,kPdgNeutron,kPdgNuMu);
 interaction->InitStatePtr()->SetProbeE(2.4);
 interaction->KinePtr()->SetW(1.35);
 interaction->KinePtr()->SetQ2(1.1);
 interaction->ExclTagPtr()->SetResonance(kP11_1440);

 // calculate d2sigma/dWdQ2 differential cross section
 // (in 1E-38 cm^2 / GeV^3)
 double diff_xsec = xsec_model->XSec(
        interaction, kPSWQ2fE) / (1E-38 * units::cm2);

 // get the integrated cross section
 // (in 1E-38 cm^2)
```

```
double intg_xsec = xsec_model->Integral(
        interaction) / (1E-38 * units::cm2);


...
}
```

B.2.2 Plugging-in to the message logging system

The message logging system is based on the *log4cpp* library. GENIE provides the *Messenger* class which enforces common formatting for messages emitted by GENIE classes and provides an easier interface to the log4cpp library. Messages are sent using one of the

- LOG(stream, priority),
- LOG_FATAL(stream),
- LOG_ALERT(stream),
- LOG_CRIT(stream),
- LOG_ERROR(stream),
- LOG_WARN(stream),
- LOG_NOTICE(stream),
- LOG_INFO(stream)
- LOG_DEBUG(stream)

Messenger macros as shown in B.1. Each message is assigned a priority level (see Table B.1) that can be used for message filtering using the

    void genie::Messenger::SetPriorityLevel(const char * stream log4cpp::Priority::Value priority)

method as shown in B.1. Each message is 'decorated' with its time stamp, its priority level, its stream name and the name space / class name / method name / line of code from where it was emitted

    time priority stream name : <method signature (line of code)> : actual message

For example:

    10891167 ERROR Config:<bool genie::ConfigPool::LoadXMLConfig() (100)>: Parsing failed

B.3 Particle decays

B.3.1 Deciding which particles to decay

GENIE attempts to simulate the complex physics within the nuclear environment and, by default, it considers that every particle which escapes the target nucleus has left its realm. It is the responsibility of the detector simulation to handle particles that propagate more than a few fermis before decaying.

Message Priority Levels
pFATAL
pALERT
pCRIT
pERROR
pWARN
pNOTICE
pINFO
pDEBUG

Table B.1: Priority levels in GENIE / log4cpp shown in decreasing importance.

Algorithm B.1 Example use of the GENIE / log4cpp message logging.

```
{
    ...
    LOG("stream-name", pFATAL) << " a fatal message";
    LOG("stream-name", pERROR) << " an error message";
    LOG("stream-name", pWARN) << " a warning";

    // alternative ways to send messages
    LOG_ERROR("stream-name") << " another error message";
    LOG_WARN("stream-name") << " another warning";
    ...
    Messenger * msg = Messenger::Instance(); // get a messenger instance
    ...
    msg->SetPriorityLevel("stream-name",pERROR); // set message threshold to 'ERROR'
    ...
    LOG("stream-name", pALERT) << " an alert – passes the message thershold";
    LOG("stream-name", pDEBUG) << " a debug message – filtered / not shown";
    ...
}
```

GENIE, for example, in its default mode, will not decay charmed hadrons. If, like many others, you think that these are "short-lived" particles GENIE ought to decay then consider this: If a C^{12} nucleus was as big as the Earth, then these particles would decay more than a light year away $(c\tau_0(\Lambda_c^+)/(C^{12}radius) \sim 2 \times 10^{10}$, $c\tau_0(D_s)/(C^{12}radius) \sim 5 \times 10^{10}$, etc). Similarly, GENIE won't decay τ leptons. The default GENIE settings are appropriate as we do not want to be making any assumption regarding the user's detector technology and its ability to detect these short tracks. (Decaying τ leptons is obviously not desirable for an emulsion detector.) By default, GENIE does not inhibit any kinematically allowed channel. Users can modify these options (see next chapter).

B.3.2 Setting particle decay flags

The default particle decay flag choices were described in the previous chapter. One can easily override the default GENIE choices by setting a series of "`DecayParticleWithCode=i`" flags at the '`$GENIE`/*config/UserPhysicsOptions.xml*' configuration file, where i is the particle's PDG code.

For example, to enable decays of τ^- leptons (PDG code = 15), one needs to change:
```
<param type=''bool'' name=''DecayParticleWithCode=15''> false </param>
```

to:
```
<param type=''bool'' name=''DecayParticleWithCode=15''> true </param>
```

B.3.3 Inhibiting decay channels

By default, GENIE does not inhibit any kinematically allowed channel. However, for certain studies, a user may wish to inhibit certain uninteresting decay channels in order to speed up event generation. This can be done by setting a series of "`InhibitDecay/Particle=i,Channel=j`" configuration options at the '`$GENIE`/*config/UserPhysicsOptions.xml*' file, where i is the particle's PDG code and j the decay channel ID. To figure out the decay channel code numbers use the *print_decay_channels.C* script in '`$GENIE`/*src/contrib/misc/*' (GENIE uses the ROOT '*TDecayChannel*' IDs).

For example, to inibit the τ^- lepton (PDG code = 15) $\tau^- \to \nu_\tau e^- \bar{\nu}_e$ decay channel (decay channel ID = 0), one needs to type:

```
<param type=''bool'' name=InhibitDecay/Particle=15,Channel=0''> true </param>
```

B.4 Numerical algorithms

B.4.1 Random number periodicity

GENIE is using ROOT's Mersenne Twistor random number generator with periodicity of 10^{6000}. See the ROOT *TRandom3* class for details. In addition GENIE is structured to use several random number generator objects each with its own "independent" random number sequence (see discussion in ROOT *TRandom* class description). GENIE provides different random number generators for different types of GENIE modules: As an example, *RandomGen::RndHadro()* returns the generator to by used in hadronization models, *RandomGen::RndDec()* returns the generator to be used by decayers, *RandomGen::RndKine()* returns the generator to be used by kinematics generators, *RandomGen::RndFsi()* returns the generator to be used by intranuclear rescattering MCs and so on... (see *RandomGen* for the list of all generators). This is an option reserved for the future as currently all modules are passed the same random number generator (no problems with the generator periodicity have been found or reported so far).

B.4.2 Setting required numerical accurancy

...

Appendix C

Summary of Important Physics Parameters

Physics Parameter	Default value	GENIE parameter name
CKM element V_{ud}	0.97377	`CKM-Vud`
CKM element V_{us}	0.2257	`CKM-Vus`
CKM element V_{cd}	0.230	`CKM-Vcd`
CKM element V_{cs}	0.957	`CKM-Vcs`
Cabbibo angle, θ_c	0.22853207	`CabbiboAngle`
Fermi coupling constant, G_F	1.16639E-5 GeV^{-2}	
Fine structure constant, α_{em}		
Weinberg angle, θ_w	0.49744211	`WeinbergAngle`
Charm mass, m_{charm}	1.430 GeV	
Anomalous magnetic moment of the proton, μ_p	2.7930	`AnomMagnMoment-P`
Anomalous magnetic moment of the neutron, μ_n	-1.913042	`AnomMagnMoment-N`
Nucleon e/m f/f, BBA2005 - $G_{ep}(a0)$	1.	
Nucleon e/m f/f, BBA2005 - $G_{ep}(a1)$	-0.0578	
Nucleon e/m f/f, BBA2005 - $G_{ep}(a2)$	0.	
Nucleon e/m f/f, BBA2005 - $G_{ep}(b1)$	11.100	
Nucleon e/m f/f, BBA2005 - $G_{ep}(b2)$	13.60	

Physics Parameter	Default value	GENIE parameter name
Nucleon e/m f/f, BBA2005 - $G_{ep}(b3)$	33.00	
Nucleon e/m f/f, BBA2005 - $G_{ep}(b4)$	0.	
Nucleon e/m f/f, BBA2005 - $G_{\mu p}(a0)$	1.	
Nucleon e/m f/f, BBA2005 - $G_{\mu p}(a1)$	0.1500	
Nucleon e/m f/f, BBA2005 - $G_{\mu p}(a2)$	0.	
Nucleon e/m f/f, BBA2005 - $G_{\mu p}(b1)$	11.100	
Nucleon e/m f/f, BBA2005 - $G_{\mu p}(b2)$	19.600	
Nucleon e/m f/f, BBA2005 - $G_{\mu p}(b3)$	7.540	
Nucleon e/m f/f, BBA2005 - $G_{\mu p}(b4)$	0.	
Nucleon e/m f/f, BBA2005 - $G_{en}(a0)$	0.	
Nucleon e/m f/f, BBA2005 - $G_{en}(a1)$	1.250	
Nucleon e/m f/f, BBA2005 - $G_{en}(a2)$	1.30	
Nucleon e/m f/f, BBA2005 - $G_{en}(b1)$	-9.86	
Nucleon e/m f/f, BBA2005 - $G_{en}(b2)$	305.0	
Nucleon e/m f/f, BBA2005 - $G_{en}(b3)$	-758.0	
Nucleon e/m f/f, BBA2005 - $G_{en}(b4)$	802.0	
Nucleon e/m f/f, BBA2005 - $G_{\mu n}(a0)$	1.	
Nucleon e/m f/f, BBA2005 - $G_{\mu n}(a1)$	1.810	
Nucleon e/m f/f, BBA2005 - $G_{\mu n}(a2)$	0.	
Nucleon e/m f/f, BBA2005 - $G_{\mu n}(b1)$	14.100	
Nucleon e/m f/f, BBA2005 - $G_{\mu n}(b2)$	20.70	
Nucleon e/m f/f, BBA2005 - $G_{\mu n}(b3)$	68.7	
Nucleon e/m f/f, BBA2005 - $G_{\mu n}(b4)$	0.	
$P_{33}(1232)$ resonance mass	1.232 GeV	
$S_{11}(1535)$ resonance mass	1.535 GeV	
$D_{13}(1520)$ resonance mass	1.520 GeV	

Physics Parameter	Default value	GENIE parameter name
$S_{11}(1650)$ resonance mass	1.650 GeV	
$D_{13}(1700)$ resonance mass	1.700 GeV	
$D_{15}(1675)$ resonance mass	1.675 GeV	
$S_{31}(1620)$ resonance mass	1.620 GeV	
$D_{33}(1700)$ resonance mass	1.700 GeV	
$P_{11}(1440)$ resonance mass	1.440 GeV	
$P_{13}(1720)$ resonance mass	1.720 GeV	
$F_{15}(1680)$ resonance mass	1.680 GeV	
$P_{31}(1910)$ resonance mass	1.910 GeV	
$P_{33}(1920)$ resonance mass	1.920 GeV	
$F_{35}(1905)$ resonance mass	1.905 GeV	
$F_{37}(1950)$ resonance mass	1.950 GeV	
$P_{11}(1710)$ resonance mass	1.710 GeV	
$P_{33}(1232)$ resonance width	0.120 GeV	
$S_{11}(1535)$ resonance width	0.150 GeV	
$D_{13}(1520)$ resonance width	0.120 GeV	
$S_{11}(1650)$ resonance width	0.150 GeV	
$D_{13}(1700)$ resonance width	0.100 GeV	
$D_{15}(1675)$ resonance width	0.150 GeV	
$S_{31}(1620)$ resonance width	0.150 GeV	
$D_{33}(1700)$ resonance width	0.300 GeV	
$P_{11}(1440)$ resonance width	0.350 GeV	
$P_{13}(1720)$ resonance width	0.150 GeV	
$F_{15}(1680)$ resonance width	0.130 GeV	
$P_{31}(1910)$ resonance width	0.250 GeV	
$P_{33}(1920)$ resonance width	0.200 GeV	

Physics Parameter	Default value	GENIE parameter name
$F_{35}(1905)$ resonance width	0.350 GeV	
$F_{37}(1950)$ resonance width	0.300 GeV	
$P_{11}(1710)$ resonance width	0.100 GeV	
NCEL axial mass,M_A	$0.990\ GeV$	`EL-Ma`
NCEL vector mass,M_V	$0.840\ GeV$	`EL-Mv`
NCEL strange axial form factor,η_{axial}	0.12	`EL-Axial-Eta`
CCQE axial mass,M_A	$0.990\ GeV$	`QEL-Ma`
CCQE vector mass,M_V	$0.840\ GeV$	`QEL-Mv`
$F_A(Q^2=0)$	-1.2670	`QEL-FA0`
CC/NC resonance (Rein-Sehgal), Ω	1.05	
CC/NC resonance (Rein-Sehgal), Z	0.762	
CC/NC resonance axial mass (Rein-Sehgal), M_A	$1.120\ GeV$	
CC/NC resonance vector mass (Rein-Sehgal),M_V	$0.840\ GeV$	
CC/NC DIS (Bodek-Yang), A	0.538	
CC/NC DIS (Bodek-Yang), B	0.305	
CC/NC DIS (Bodek-Yang), C_s^u	0.363	
CC/NC DIS (Bodek-Yang), C_s^d	0.621	
CC/NC DIS (Bodek-Yang), C_{v1}^u	0.291	
CC/NC DIS (Bodek-Yang), C_{v2}^u	0.189	
CC/NC DIS (Bodek-Yang), C_{v1}^d	0.202	
CC/NC DIS (Bodek-Yang), C_{v2}^d	0.255	
CC/NC DIS (Bodek-Yang), X_0	-0.00817	
CC/NC DIS (Bodek-Yang), X_1	0.0506	
CC/NC DIS (Bodek-Yang), X_2	0.0798	
CC/NC DIS (Bodek-Yang), PDF Q_{min}^2	$0.800\ GeV^2$	
CC/NC DIS (Bodek-Yang), Uncorr. PDF set	GRV98LO	

Physics Parameter	Default value	GENIE parameter name
CC/NC DIS (Bodek-Yang), Nuclear mod.?	true	
CC/NC DIS (Bodek-Yang), Whitlow R (F_L)?	true	
CC/NC coherent π (Rein-Sehgal), M_A	1.000 GeV	
CC/NC coherent π (Rein-Sehgal), R_0	1.000 fm	
CC/NC coherent π (Rein-Sehgal), $Re/ImAmpl$	0.300	
CC/NC coherent π (Rein-Sehgal), Mod. PCAC?	true	
Transition region modeling (neugen3), $R_{bkg}^{\nu p CC1\pi}$	0.100	
Transition region modeling (neugen3), $R_{bkg}^{\nu p CC2\pi}$	1.000	
Transition region modeling (neugen3), $R_{bkg}^{\nu p NC1\pi}$	0.100	
Transition region modeling (neugen3), $R_{bkg}^{\nu p NC2\pi}$	1.000	
Transition region modeling (neugen3), $R_{bkg}^{\nu n CC1\pi}$	0.300	
Transition region modeling (neugen3), $R_{bkg}^{\nu n CC2\pi}$	1.000	
Transition region modeling (neugen3), $R_{bkg}^{\nu n NC1\pi}$	0.300	
Transition region modeling (neugen3), $R_{bkg}^{\nu n NC2\pi}$	1.000	
Transition region modeling (neugen3), $R_{bkg}^{\bar{\nu} p CC1\pi}$	0.300	
Transition region modeling (neugen3), $R_{bkg}^{\bar{\nu} p CC2\pi}$	1.000	
Transition region modeling (neugen3), $R_{bkg}^{\bar{\nu} p NC1\pi}$	0.300	
Transition region modeling (neugen3), $R_{bkg}^{\bar{\nu} p NC2\pi}$	1.000	
Transition region modeling (neugen3), $R_{bkg}^{\bar{\nu} n CC1\pi}$	0.100	
Transition region modeling (neugen3), $R_{bkg}^{\bar{\nu} n CC2\pi}$	1.000	
Transition region modeling (neugen3), $R_{bkg}^{\bar{\nu} n NC1\pi}$	0.100	
Transition region modeling (neugen3), $R_{bkg}^{\bar{\nu} n NC2\pi}$	1.000	
Transition region modeling (neugen3), W_{cut}	1.7 GeV	
Average chg. hadron multiplicity factor (KNO), $a_{\nu p}$	0.4	
Average chg. hadron multiplicity factor (KNO), $a_{\nu n}$	-0.2	
Average chg. hadron multiplicity factor (KNO, $a_{\bar{\nu} p}$	0.02	

Physics Parameter	Default value	GENIE parameter name
Average chg. hadron multiplicity factor (KNO), $a_{\bar{\nu}n}$	0.80	
Average chg. hadron multiplicity factor (KNO), $b_{\nu p}$	1.42	
Average chg. hadron multiplicity factor (KNO), $b_{\nu n}$	1.42	
Average chg. hadron multiplicity factor (KNO), $b_{\bar{\nu}p}$	1.28	
Average chg. hadron multiplicity factor (KNO), $b_{\bar{\nu}n}$	0.95	
Average strange baryon multiplicity factor (KNO), a_{hyp}	0.021951447	
Average strange baryon multiplicity factor (KNO), b_{hyp}	0.041969985	
KNO parameterization (Levy), $c_{\nu p}$	7.93	
KNO parameterization (Levy), $c_{\nu n}$	5.22	
KNO parameterization (Levy), $c_{\bar{\nu}p}$	5.22	
KNO parameterization (Levy), $c_{\bar{\nu}n}$	7.93	
$\pi^0\pi^0$ probability (AGKY/KNO)	0.3133	
$\pi^+\pi^-$ probability (AGKY/KNO)	0.6267	
$K^0\bar{K}^0$ probability (AGKY/KNO)	0.03	
K^+K^- probability (AGKY/KNO)	0.03	
Phase-space p_T reweighting factor (AGKY/KNO)	3.5	
$s\bar{s}$ suppression factor (PYTHIA6)	0.30	
Gaussian p_T^2 (PYTHIA6)	0.44	
Non-gaussian p_T^2 tail (PYTHIA6)	0.01	
Remaining energy cut-off (PYTHIA6)	0.20	
Formation time, $c\tau_0$	0.342 fm	
Binding energy (RFG), $E_b(\mathrm{Li}^6)$	0.017 GeV	
Binding energy (RFG), $E_b(\mathrm{C}^{12})$	0.025 GeV	
Binding energy (RFG), $E_b(\mathrm{O}^{16})$	0.027 GeV	
Binding energy (RFG), $E_b(\mathrm{Mg}^{24})$	0.032 GeV	
Binding energy (RFG), $E_b(\mathrm{Ca}^{40})$	0.028 GeV	

Physics Parameter	Default value	GENIE parameter name
Binding energy (RFG), $E_b(\text{Fe}^{56})$	0.036 GeV	
Binding energy (RFG), $E_b(\text{Ni}^{58})$	0.036 GeV	
Binding energy (RFG), $E_b(\text{Pb}^{208})$	0.044 GeV	
Fermi momentum (RFG), $k_F(\text{Li}^{6}, \text{p})$	0.169 GeV	
Fermi momentum (RFG), $k_F(\text{Li}^{6}, \text{n})$	0.169 GeV	
Fermi momentum (RFG), $k_F(\text{C}^{12}, \text{p})$	0.221 GeV	
Fermi momentum (RFG), $k_F(\text{C}^{12}, \text{n})$	0.221 GeV	
Fermi momentum (RFG), $k_F(\text{O}^{16}, \text{p})$	0.225 GeV	
Fermi momentum (RFG), $k_F(\text{O}^{16}, \text{n})$	0.225 GeV	
Fermi momentum (RFG), $k_F(\text{Mg}^{24}, \text{p})$	0.235 GeV	
Fermi momentum (RFG), $k_F(\text{Mg}^{24}, \text{n})$	0.235 GeV	
Fermi momentum (RFG), $k_F(\text{Si}^{28}, \text{p})$	0.239 GeV	
Fermi momentum (RFG), $k_F(\text{Si}^{28}, \text{n})$	0.239 GeV	
Fermi momentum (RFG), $k_F(\text{Ar}^{40}, \text{p})$	0.242 GeV	

Appendix D

Common Status and Particle Codes

D.1 Status codes

Description	GHepStatus_t	As *int*
Undefined	*kIStUndefined*	-1
Initial state	*kIStInitialState*	0
Stable final state	*kIstStableFinalState*	1
Intermediate state	*kIStIntermediateState*	2
Decayed state	*kIStDecayedState*	3
Nucleon target	*kIStNucleonTarget*	11
DIS pre-fragm. hadronic state	*kIStDISPreFragmHadronicState*	12
Resonant pre-decayed state	*kIStPreDecayResonantState*	13
Hadron in the nucleus	*kIStHadronInTheNucleus*	14
Final state nuclear remnant	*kIStFinalStateNuclearRemnant*	15
Nucleon cluster target	*kIStNucleonClusterTarget*	16

D.2 Particle codes

See PDG 'Monte Carlo Particle Numbering Scheme' for a complete list.http://pdg.lbl.gov/2008/mcdata/mc_particle_id_contents.shtml

ν_e $(\bar{\nu_e})$	12 (-12)	p	2212	π^0	111	uu $(s=1)$	2203	g	21
ν_μ $(\bar{\nu_\mu})$	14 (-14)	n	2112	π^+ (π^-)	211 (-211)	ud $(s=0)$	2101	γ	22
ν_τ $(\bar{\nu_\tau})$	16 (-16)	Λ^0	3122	ρ^0	113	ud $(s=1)$	2103	Z^0	23
e^- (e^+)	11 (-11)	Σ^+	3222	ρ^+ (ρ^-)	213 (-213)	su $(s=0)$	3201	W^+ (W^-)	24 (-24)
μ^- (μ^+)	13 (-13)	Σ^0	3212	η	221	su $(s=1)$	3203		
τ^- (τ^+)	15 (-15)	Σ^-	3112	η'	331	sd $(s=0)$	3101		
d $(\bar{d})$	1 (-1)	Ξ^0	3322	ω	223	sd $(s=1)$	3103		
u $(\bar{u})$	2 (-2)	Ξ^-	3312	ϕ	333	ss $(s=1)$	3303		
s $(\bar{s})$	3 (-3)	Ω^-	3332	η_c	441				
c $(\bar{c})$	4 (-4)	Λ_c^+	4122	J/ψ	443				
b $(\bar{b})$	5 (-5)	Σ_c^0	4112	K^0 $(\bar{K}^0)$	311 (-311)				
t $(\bar{t})$	6 (-6)	Σ_c^+	4212	K^+ (K^-)	321 (-321)				
		Σ_c^{++}	4222	K_L^0	130				
		Ξ_c^0	4132	K_S^0	310				
		Ξ_c^+	4232	D^0 $(\bar{D}^0)$	421 (-421)				
		Ω_c^0	4332	D^+ (D^-)	411 (-411)				
				D_s^+ (D_s^-)	431 (-431)				

D.3   Baryon resonance codes

$P_{33}(1232);\ \Delta^-$	1114	$S_{11}(1650);\ N^0$	32112	$D_{13}(1700);\ N^0$	21214	$P_{31}(1910);\ \Delta^-$	21112
$P_{33}(1232);\ \Delta^0$	2114	$S_{11}(1650);\ N^+$	32212	$D_{13}(1700);\ N^+$	22124	$P_{31}(1910);\ \Delta^0$	21212
$P_{33}(1232);\ \Delta^+$	2214	$D_{15}(1675);\ N^0$	2116	$P_{11}(1710);\ N^0$	42112	$P_{31}(1910);\ \Delta^+$	22122
$P_{33}(1232);\ \Delta^{++}$	2224	$D_{15}(1675);\ N^+$	2216	$P_{11}(1710);\ N^+$	42212	$P_{31}(1910);\ \Delta^{++}$	22222
$P_{11}(1440);\ N^0$	12112	$F_{15}(1680);\ N^0$	12116	$P_{13}(1720);\ N^0$	31214	$P_{33}(1920);\ \Delta^-$	21114
$P_{11}(1440);\ N^+$	12212	$F_{15}(1680);\ N^+$	12216	$P_{13}(1720);\ N^+$	32124	$P_{33}(1920);\ \Delta^0$	22114
$D_{13}(1520);\ N^0$	1214	$D_{33}(1700);\ \Delta^-$	11114	$F_{35}(1905);\ \Delta^-$	1116	$P_{33}(1920);\ \Delta^+$	22214
$D_{13}(1520);\ N^+$	2124	$D_{33}(1700);\ \Delta^0$	12114	$F_{35}(1905);\ \Delta^0$	1216	$P_{33}(1920);\ \Delta^{++}$	22224
$S_{11}(1535);\ N^0$	22112	$D_{33}(1700);\ \Delta^+$	12214	$F_{35}(1905);\ \Delta^+$	2126	$F_{37}(1950);\ \Delta^-$	1118
$S_{11}(1535);\ N^+$	22212	$D_{33}(1700);\ \Delta^{++}$	12224	$F_{35}(1905);\ \Delta^{++}$	2226	$F_{37}(1950);\ \Delta^0$	2118
$S_{31}(1620);\ \Delta^-$	11112					$F_{37}(1950);\ \Delta^+$	2218
$S_{31}(1620);\ \Delta^0$	1212					$F_{37}(1950);\ \Delta^{++}$	2228
$S_{31}(1620);\ \Delta^+$	2122						
$S_{31}(1620);\ \Delta^{++}$	2222						

D.4   Ion codes

GENIE has adopted the standard PDG (2006) particle codes. For ions it has adopted a PDG extension, using the 10-digit code 10LZZZAAAI where AAA is the total baryon number, ZZZ is the total charge, L is the number of strange quarks and I is the isomer number (I=0 corresponds to the ground state).

So, for example:

$1000010010 \rightarrow H^1$
$1000060120 \rightarrow C12 :$
$1000080160 \rightarrow O^{16} :$
$1000260560 \rightarrow Fe^{56} :$

and so on.

D.5   GENIE pseudo-particle codes

GENIE-specific pseudo-particles have PDG codes $>= 2000000000$.

Appendix E

3rd Party Softw. Installation Instructions

The following dependencies need to be installed, in the following order.

LOG4CPP

Before installing log4cpp

Check whether log4cpp is already installed at your system. The library filename contains liblog4cpp, so if you cannot find a file with a filename containing liblog4cpp, then you probably do not have the software installed.

Getting the source code

Download the source code from the sourceforge anonymous CVS repository (when prompted for a password, simply hit enter):
```
$ cd /dir/for/external/src/code
$ cvs -d :pserver:anonymous@log4cpp.cvs.sourceforge.net:/cvsroot/log4cpp login
$ cvs -d :pserver:anonymous@log4cpp.cvs.sourceforge.net:/cvsroot/log4cpp -z3 co log4cpp
```

Configuring and building

Enter the log4cpp directory and run '`autogen`' and '`configure`'. Replace [location] with the installation directory of your choice; you cannot install it in the same directory as the source (where you are now). You can choose not to use the '`--prefix`' tag, in which case the default install directory is '*/usr/local*'.
```
$ cd log4cpp
$ ./autogen.sh
$ ./configure --prefix=[location]
```

What's left is to run '`make`' and '`make install`'. If make install gives you an error while copying or moving files stating that the files are identical, then you probably choose the source folder as your install folder in the above configure step. Rerun configure with a different location (or simply leave the '`--prefix`' option out for the default).
```
$ make
$ make install
```

Notes:

- Alternatively, you may install pre-compiled binaries. For example, if you are using 'yum' on LINUX then just type:
  ```
  $ yum install log4cpp
  ```
  On MAC OS X you can do the same using 'DarwinPorts':
  ```
  $ sudo port install log4cpp
  ```

LIBXML2

Before installing libxml2

Check whether libxml2 is already installed at your system - most likely it is. Look for a libxml2.* library (typically in '*/usr/lib*') and for a libxml2 include folder (typically in '*/usr/include*').

Getting the source code

Download the source code from the GNOME subversion repository:
```
$ cd /dir/for/external/src/code
$ svn co https://svn.gnome.org/svn/libxml2/trunk libxml2
```

Alternatively, you download the code as a gzipped tarball from:
http://xmlsoft.org/downloads.html.

Configuring and building

```
$ cd libxml2
$ ./autogen.sh --prefix=[location]
$ make
$ make install
```

Notes:

- Alternatively, you may install pre-compiled binaries. For example, if you are using 'yum' on LINUX then just type:
  ```
  $ yum install libxml2
  ```
  On MAC OS X you can do the same using 'DarwinPorts':
  ```
  $ sudo port install libxml2
  ```

LHAPDF

Getting the source code

Get the LHAPDF code (and PDF data files) from *http://projects.hepforge.org/lhapdf/*. The tarball corrsponding to version '*x.y.z*' is named '*lhapdf-x.y.z.tar.gz*'.
```
$ mv lhapdf-x.y.z.tar.gz /dir/for/external/src/code
$ cd /directory/to/download/external/code
$ tar xzvf lhapdf-x.y.z.tar.gz
```

Configuring and building

```
$ cd lhapdf-x.y.z/
$ ./configure --prefix=[location]
$ make
$ make install
```

PYTHIA6

Installation of PYTHIA6 is simplified by using a script provided by Robert Hatcher ('*build_pythia6.sh*'). The file is included in the GENIE source tree (see '`$GENIE`*/src/scripts/build/ext/build_pythia6.sh*'). You can also get a copy from the web[1]:

You can run the script (please, also read its documentation) as:
```
$ source build_pythia6.sh [version]
```

For example, in order to download and install version 6.4.12, type:
```
$ source build_pythia6.sh 6412
```

ROOT

Getting the source code

Get the source code from the ROOT subversion repository. To get the development version, type:
```
$ cvs co http://root.cern.ch/svn/root/trunk root
```
To get a specific version '*x.y.z*', type:
```
$ cvs co http://root.cern.ch/svn/root/tags/vx-y-z root
$ cvs co http://root.cern.ch/svn/root/tags/v5-22-00 root
```

See *http://root.cern.ch/drupal/content/downloading-root/*

Configuring and building

```
$ export ROOTSYS=/path/to/install_root
$ cd $ROOTSYS
$ ./configure [arch] [other options] --enable-pythia6 --with-pythia6-libdir=$PYTHIA6 --enable-mathmore
$ make
```

Testing

Accessing root is an easy test to see if it has installed correctly. If you are not familiar with root, use ".q" in root prompt to quit.

```
$ root -l
```

---

[1]Visit: *http://projects.hepforge.org/genie/trac/browser/trunk/src/scripts/build/ext/*
Click on the file and then download it by clicking on 'Download in other formats / Original format' towards the end of the page.

```
root [0] .q
```

See *http://root.cern.ch/root/Install.html* for more information on installing ROOT from source.

Appendix F

Finding More Information

F.1 The GENIE web page

The GENIE web page, hosted at HepForge is the exclusive official source of information on GENIE. The page can be reached at `http://www.genie-mc.org`

F.2 Subscribing at the GENIE mailing lists

The GENIE mailing lists are hosted at JISCmail, UK's National Academic Mailing List Service. We currently maintain two mailing lists

- `neutrino-mc-support@jiscmail.ac.uk` : This is the GENIE support mailing list and is open to all users.

- `neutrino-mc-core@jiscmail.ac.uk` : This is the GENIE developers mailing list and is open only to members of the GENIE collaboration.

To register at the GENIE support mailing list go to
https://www.jiscmail.ac.uk/cgi-bin/webadmin?A0=NEUTRINO-MC-SUPPORT
(or follow the link the GENIE web page) and click on 'Join or Leave NEUTRINO-MC-SUPPORT'. In the registration page specify your name, preferred e-mail address and subscription type and click on 'Join NEUTRINO-MC-SUPPORT'. This will generate a request that has to be approved by a member of the GENIE collaboration. Upon approval a notification and the JISCmail Data Protection policy will be forwarded at your nominated e-mail address.

F.3 The GENIE document database (DocDB)

The GENIE internal note repository is hosted at Fermilab at the Projects Document Database. Most documents are internal to the GENIE collaboration. However certain documents are made publicly available. The Fermilab Projects Documents Database can be reached at:
http://projects-docdb.fnal.gov:8080/cgi-bin/ListBy?groupid=30

F.4   The GENIE issue tracker

The issue tracker hosted at HepForge is a useful tool for monitoring tasks and milestones, for submitting bug reports and getting information about their resolution. It is available at:
http://projects.hepforge.org/genie/trac/report/

Non-developers can also submit tickets. A general 'guest' account has been setup (the password is available upon request).

F.5   The GENIE Generator repository browser

http://projects.hepforge.org/genie/trac/browser/generator

F.6   The GENIE doxygen documentation

http://doxygen.genie-mc.org/

Appendix G

Glossary

- **A**

  – AGKY: A home-grown neutrino-induced hadronic multiplarticle production model developed by C.Andreopoulos, H.Gallagher, P.Kehayias and T.Yang.

- **B**

  – BGLRS: An atmospheric neutrino simulation developed by G. Barr, T.K. Gaisser, P. Lipari, S. Robbins and T. Stanev.

  – BY: Bodek-Yang.

- **C**

  – COH:

  – CVS:

- **D**

  – DIS: Deep Inelastic Scattering.

- **E**

- **F**

  – FGM: Fermi Gas Model.

  – FLUKA:

- **G**

  – GEF: Geocentric Earth-Fixed Coordinate System (+z: Points to North Pole / xy: Equatorial plane / +x: Points to the Prime Meridian / +y: As needed to make a right-handed coordinate system).

  – Geant4:

  – GDML:

- GENEVE: A legacy, fortran77-based neutrino generator by F.Cavanna et al.
- GENIE: **G**enerates **E**vents for **N**eutrino **I**nteraction **E**xperiments.
- GiBUU: A fortran2003-based state-of-the-art particle transport simulation using the Boltzmann-Uehling-Uhlenbeck (BUU) framework. Developed primarily by the theory group at Giessen University (U.Mosel et al.)
- GNuMI: Geant3- and Geant4-based NuMI beamline simulation software.
- GSL: GNU Scientific Library
- gevdump: A GENIE application for printing-out event records.
- gevpick: A GENIE event topology cherry-picking application.
- gevgen: A simple, generic GENIE event generation application.
- gevgen_hadron: A GENIE hadron+nucleus event generation application.
- gevgen_atmo: A GENIE event generation application for atmospheric neutrinos.
- gevgen_ndcy: A GENIE nucleon decay event generation application.
- gevgen_t2k: A GENIE event generation application customized for T2K.
- gevgen_fnal: A GENIE event generation application customized for the NuMI beamline experiments.
- gmkspl: A GENIE application for generating cross section spline files (evet generation inputs).
- gntpc: A GENIE ntuple conversion application.
- gspladd: A GENIE XML cross section spline file merging application.
- gspl2root: A GENIE XML to ROOT cross section spline file conversion utility.
- gevgen_numi: Alias for gevgen_fnal maintained for historical reasons.

- **H**

  - hA: See INTRANUKE.
  - hN: See INTRANUKE.

- **I**

  - IMD: Inverse Muon Decay
  - INTRANUKE: A home-grown intranuclear hadron transport MC. Intranuke was initially developed within NEUGEN for the Soudan-2 experiment by W.A.Mann, R.Merenyi, R.Edgecock, H.Gallagher, G.F.Pearce and others. Since then it was significantly improved and is now extensively used by MINOS and other experiments. Current INTRANUKE development is led by S.Dytman. INTRANUKE, in fact, contains two independent models (called 'hN' and 'hA').

- **J**

  - JNUBEAM: Geant3-based JPARC neutrino beamline simulation software.
  - JPARC: Japan Proton Accelerator Research Complex. Home of T2K neutrino beamline.

- **K**

  - KNO: Koba, Nielsen and Olesen scaling law.

- **L**

  - LHAPDF: Les Houches Accord PDF Interface.
  - libxml2: The XML C parser and toolkit of Gnome (see http://xmlsoft.org).
  - log4cpp: A library of C++ classes for fleible loggingto files, syslog, IDSA and other destinations (see http://log4cpp.sourceforge.net).

- **M**

  - MacPorts: An open-source community initiative to design an easy-to-use system for compiling, installing, and upgrading either command-line, X11 or Aqua based open-source software on the Mac OS X operating system.
  - Mersenne Twistor: The default random number generator in GENIE (via ROOT TRandom3 whose implementation is based on M. Matsumoto and T. Nishimura, Mersenne Twistor: A 623-diminsionally equidistributed uniform pseudorandom number generator ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3–30.)

- **N**

  - NeuGEN: A legacy, fortran77-based neutrino generator by H.Gallagher et al.
  - NEUT: A legacy, fortran77-based neutrino generator by Y.Hayato et al.
  - NUANCE: A legacy, fortran77-based neutrino generator by D.Casper et al.
  - NUX: A legacy, fortran77-based neutrino generator by A.Rubbia et al.
  - NuMI: Neutrinos at the Main Injector. A neutrino beamline at Fermilab.

- **O**

- **P**

  - PREM: Preliminary Earth Model, The Encyclopedia of Solid Earth Geophysics, David E. James, ed., Van Nostrand Reinhold, New York, 1989, p.331
  - PYTHIA:

- **Q**

  - QEL: Quasi-Elastic.

- **R**

  - RES: Resonance.
  - Registry:
  - ROOT:
  - RooTracker: A ROOT-only STDHEP-like event format (very similar to GHEP event format but with no GENIE class dependencies) developed in GENIE as an evolution of the Tracker format. See also Tracker.
  - RS: Rein-Sehgal
  - RSB: Rein-Sehgal-Berger

  – RSD: Remote Software Deployment Tools. A system for automated software installation developed by Nick West (Oxford).

- **S**

  – SF:
  – SVN: See Subversion.
  – SKDETSIM: The fortran77-based Super-Kamiokande detector simulation.
  – Subversion:

- **T**

  – THZ: Topocentric Horizontal Coordinate System (+z: Points towards the Local Zenith / +x: On same plane as Local Meridian, pointing South / +y: as needed to make a right-handed coordinate system / Origin: Detector centre).
  – Tracker:

- **U**

- **V**

- **W**

- **X**

  – XML: Extensible Markup Language.

- **Y**:

  – Yum: Yellowdog Updater, Modified (YUM). An open-source command-line package-management utility for RPM-compatible Linux operating systems.

- **Z**

Bibliography

[1] C. Andreopoulos *et al. http://www.genie-mc.org.*

[2] S. Agostinelli *et al.*, "GEANT4: A simulation toolkit," *Nucl. Instrum. Meth.*, vol. A506, pp. 250–303, 2003.

[3] M. Bahr *et al.*, "Herwig++ Physics and Manual," *Eur. Phys. J.*, vol. C58, pp. 639–707, 2008.

[4] T. Sjostrand, S. Mrenna, and P. Skands, "A Brief Introduction to PYTHIA 8.1," *Comput. Phys. Commun.*, vol. 178, pp. 852–867, 2008.

[5] R. Brun and F. Rademakers, "ROOT: An object oriented data analysis framework," *Nucl. Instrum. Meth.*, vol. A389, pp. 81–86, 1997.

[6] H. Gallagher, "The neugen neutrino event generator," *Nucl. Phys. Proc. Suppl.*, vol. 112, pp. 188–194, 2002.

[7] F. Cavanna and O. Palamara, "Geneve: A monte carlo generator for neutrino interactions in the intermediate-energy range," *Nucl. Phys. Proc. Suppl.*, vol. 112, pp. 183–187, 2002.

[8] Y. Hayato, "Neut," *Nucl. Phys. Proc. Suppl.*, vol. 112, pp. 171–176, 2002.

[9] D. Casper, "The nuance neutrino physics simulation, and the future," *Nucl. Phys. Proc. Suppl.*, vol. 112, pp. 161–170, 2002.

[10] A. Rubbia, "Nux-neutrino generator," *Talk at the 1st Workshop on Neutrino-Nucleus Interactions in the Few-GeV Region (NuINT01)*, vol. http://neutrino.kek.jp/nuint01/slide/Rubbia.1.pdf, p. 29, 2001.

[11] C. Juszczak, J. A. Nowak, and J. T. Sobczyk, "Simulations from a new neutrino event generator," *Nucl. Phys. Proc. Suppl.*, vol. 159, pp. 211–216, 2006.

[12] T. Leitner, L. Alvarez-Ruso, and U. Mosel, "Charged current neutrino nucleus interactions at intermediate energies," *Phys. Rev.*, vol. C73, p. 065502, 2006.

[13] A. Fasso *et al.*, "The physics models of FLUKA: Status and recent development," 2003.

[14] T. Ishida, "Charged-current inclusive distributions from K2K near detectors," 2002. ,Prepared for 1st Workshop on Neutrino - Nucleus Interactions in the Few GeV Region (NuInt01), Tsukuba, Japan, 13-16 Dec 2001.

[15] A. A. Aguilar-Arevalo *et al.*, "Measurement of muon neutrino quasi-elastic scattering on carbon," *Phys. Rev. Lett.*, vol. 100, p. 032301, 2008.

[16] F. Sanchez, "Search for neutrino-induced charged current coherent pion production with carbon in a 1.3-gev wide band muon neutrino beam," *Nucl. Phys. Proc. Suppl.*, vol. 155, pp. 239–241, 2006.

[17] A. A. Aguilar-Arevalo *et al.*, "Unexplained Excess of Electron-Like Events From a 1-GeV Neutrino Beam," *Phys. Rev. Lett.*, vol. 102, p. 101802, 2009.

[18] E. A. Paschos, A. Kartavtsev, and G. J. Gounaris, "Coherent pion production by neutrinos on nuclei," *Phys. Rev.*, vol. D74, p. 054007, 2006.

[19] S. K. Singh, M. Sajjad Athar, and S. Ahmad, "Nuclear effects in neutrino induced coherent pion production at k2k and miniboone," *Phys. Rev. Lett.*, vol. 96, p. 241801, 2006.

[20] L. Alvarez-Ruso, L. S. Geng, S. Hirenzaki, and M. J. Vicente Vacas, "Charged current neutrino induced coherent pion production," *Phys. Rev.*, vol. C75, p. 055501, 2007.

[21] J. A. Harvey, C. T. Hill, and R. J. Hill, "Anomaly mediated neutrino-photon interactions at finite baryon density," *Phys. Rev. Lett.*, vol. 99, p. 261601, 2007.

[22] O. Buss, T. Leitner, U. Mosel, and L. Alvarez-Ruso, "The influence of the nuclear medium on inclusive electron and neutrino scattering off nuclei," *Phys. Rev.*, vol. C76, p. 035502, 2007.

[23] O. Benhar, N. Farina, H. Nakamura, M. Sakuda, and R. Seki, "Electron and neutrino nucleus scattering in the impulse approximation regime," *Phys. Rev.*, vol. D72, p. 053005, 2005.

[24] J. E. Amaro, M. B. Barbaro, J. A. Caballero, T. W. Donnelly, and J. M. Udias, "Final-state interactions and superscaling in the semi- relativistic approach to quasielastic electron and neutrino scattering," *Phys. Rev.*, vol. C75, p. 034613, 2007.

[25] A. Bodek and J. L. Ritchie, "Further studies of fermi motion effects in lepton scattering from nuclear targets," *Phys. Rev.*, vol. D24, p. 1400, 1981.

[26] E. J. Moniz *et al.*, "Nuclear fermi momenta from quasielastic electron scattering," *Phys. Rev. Lett.*, vol. 26, pp. 445–448, 1971.

[27] H. De Vries, C. W. De Jager, and C. De Vries, "Nuclear charge and magnetization density distribution parameters from elastic electron scattering," *Atom. Data Nucl. Data Tabl.*, vol. 36, pp. 495–536, 1987.

[28] A. Bodek and U. K. Yang, "Higher twist, ξ_w scaling, and effective LO PDFs for lepton scattering in the few GeV region," *J. Phys.*, vol. G29, pp. 1899–1906, 2003.

[29] H. Ejiri, "Nuclear deexcitations of nucleon holes associated with nucleon decays in nuclei," *Phys. Rev.*, vol. C48, pp. 1442–1444, 1993.

[30] K. Kobayashi *et al.*, "Detection of nuclear de-excitation gamma-rays in water Cherenkov detector," *Nucl. Phys. Proc. Suppl.*, vol. 139, pp. 72–76, 2005.

[31] C. H. Llewellyn Smith, "Neutrino reactions at accelerator energies," *Phys. Rept.*, vol. 3, p. 261, 1972.

[32] R. G. Sachs, "High-Energy Behavior of Nucleon Electromagnetic Form Factors," *Phys. Rev.*, vol. 126, pp. 2256–2260, 1962.

[33] H. Budd, A. Bodek, and J. Arrington, "Modeling quasi-elastic form factors for electron and neutrino scattering, hep-ex/0308005," 2003.

[34] R. Bradford, A. Bodek, H. S. Budd, and J. Arrington, "A new parameterization of the nucleon elastic form factors," *Nucl. Phys. Proc. Suppl.*, vol. 159, pp. 127–132, 2006.

[35] L. A. Ahrens *et al.*, "Measurement of Neutrino - Proton and anti-neutrino - Proton Elastic Scattering," *Phys. Rev.*, vol. D35, p. 785, 1987.

[36] D. Rein and L. M. Sehgal, "Neutrino excitation of baryon resonances and single pion production," *Ann. Phys.*, vol. 133, p. 79, 1981.

[37] R. P. Feynman, M. Kislinger, and F. Ravndal, "Current matrix elements from a relativistic quark model," *Phys. Rev.*, vol. D3, pp. 2706–2732, 1971.

[38] K. S. Kuzmin, V. V. Lyubushkin, and V. A. Naumov, "Axial masses in quasielastic neutrino scattering and single-pion neutrinoproduction on nucleons and nuclei," *Acta Phys. Polon.*, vol. B37, pp. 2337–2348, 2006.

[39] D. Rein and L. M. Sehgal, "Coherent π^0 production in neutrino reactions," *Nucl. Phys.*, vol. B223, p. 29, 1983.

[40] W. M. Yao *et al.*, "Review of particle physics," *J. Phys.*, vol. G33, pp. 1–1232, 2006.

[41] D. Rein and L. M. Sehgal, "PCAC and the deficit of forward muons in π^+ production by neutrinos," *Phys. Lett.*, vol. B657, pp. 207–209, 2007.

[42] L. W. Whitlow, S. Rock, A. Bodek, E. M. Riordan, and S. Dasu, "A Precise extraction of R = σ_L/σ_T from a global analysis of the SLAC deep inelastic e p and e d scattering cross-sections," *Phys. Lett.*, vol. B250, pp. 193–198, 1990.

[43] M. Gluck, E. Reya, and A. Vogt, "Dynamical parton distributions revisited," *Eur. Phys. J.*, vol. C5, pp. 461–470, 1998.

[44] S. G. Kovalenko, "Quasielastic neutrino production of charmed baryons from the point of view of local duality," *Sov. J. Nucl. Phys.*, vol. 52, pp. 934–936, 1990.

[45] M. Bischofberger, "Quasi elastic charm production in neutrino nucleon scattering," *UMI-C821490*.

[46] M. A. G. Aivazis, F. I. Olness, and W.-K. Tung, "Leptoproduction of heavy quarks. 1. General formalism and kinematics of charged current and neutral current production processes," *Phys. Rev.*, vol. D50, pp. 3085–3101, 1994.

[47] G. De Lellis, F. Di Capua, and P. Migliozzi, "Prediction of charm-production fractions in neutrino interactions. ((U)) ((V))," *Phys. Lett.*, vol. B550, pp. 16–23, 2002.

[48] C. Peterson, D. Schlatter, I. Schmitt, and P. M. Zerwas, "Scaling violations in inclusive e+ e- annihilation spectra," *Phys. Rev.*, vol. D27, p. 105, 1983.

[49] P. D. B. Collins and T. P. Spiller, "The fragmentation of heavy quarks," *J. Phys.*, vol. G11, p. 1289, 1985.

[50] D. Y. Bardin and V. A. Dokuchaeva, "Muon energy spectrum in inverse muon decay," *Nucl. Phys.*, vol. B287, p. 839, 1987.

[51] W. J. Marciano and Z. Parsa, "Neutrino-electron scattering theory," *J. Phys.*, vol. G29, pp. 2629–2645, 2003.

[52] K. S. Kuzmin, V. V. Lyubushkin, and V. A. Naumov, "How to sum contributions into the total charged-current neutrino nucleon cross section, hep-ph/0511308," 2005.

[53] D. MacFarlane *et al.*, "Nucleon Structure Functions from High-Energy Neutrino Interactions with Iron and QCD Results," *Z. Phys.*, vol. C26, pp. 1–12, 1984.

[54] J. P. Berge *et al.*, "Total neutrino and anti-neutrino charged current cross section measurements in 100 GeV, 160 GeV and 200 GeV narrow band beams," *Z. Phys.*, vol. C35, p. 443, 1987.

[55] S. Ciampolillo *et al.*, "Total cross section for neutrino charged current interactions at 3 GeV and 9 GeV," *Phys. Lett.*, vol. B84, p. 281, 1979.

[56] D. C. Colley *et al.*, "Cross sections for charged current neutrino and anti-neutrino interactions in the energy range 10 GeV to 50 GeV," *Zeit. Phys.*, vol. C2, p. 187, 1979.

[57] P. Bosetti *et al.*, "Total cross sections for muon-neutrino and anti-muon- neutrino charged current interactions between 20 GeV and 200 GeV," *Phys. Lett.*, vol. B110, p. 167, 1982.

[58] A. I. Mukhin *et al.*, "Energy dependence of total cross section for neutrino and anti-neutrino interactions at energies below 35 GeV," *Sov. J. Nucl. Phys.*, vol. 30, p. 528, 1979.

[59] D. S. Baranov *et al.*, "Measurements of the $\nu_\mu N$ total cross section at 2 GeV - 30 GeV in SKAT neutrino experiment," *Phys. Lett.*, vol. B81, p. 255, 1979.

[60] S. J. Barish *et al.*, "Study of Neutrino Interactions in Hydrogen and Deuterium: Inelastic Charged Current Reactions," *Phys. Rev.*, vol. D19, p. 2521, 1979.

[61] N. J. Baker *et al.*, "Total cross sections for $\nu_\mu n$ and $\bar{\nu}_\mu p$ charged current interactions in the 7-ft bubble chamber," *Phys. Rev.*, vol. D25, pp. 617–623, 1982.

[62] T. Eichten *et al.*, "Measurement of the neutrino - nucleon anti-neutrino - nucleon total cross sections," *Phys. Lett.*, vol. B46, pp. 274–280, 1973.

[63] W. Lerche *et al.*, "Experimental Study of the Reaction $\nu_\mu p \to \mu^- p \pi^+$. GARGAMELLE Neutrino Propane Experiment," *Phys. Lett.*, vol. B78, pp. 510–514, 1978.

[64] V. V. Ammosov *et al.*, "Study of the reaction $\nu_\mu p \to \mu^- \Delta^{++}$ at energies 3 GeV to 30 GeV," *Sov. J. Nucl. Phys.*, vol. 50, pp. 67–69, 1989.

[65] H. J. Grabosch *et al.*, "Cross section measurements of single pion production in charged current neutrino and anti-neutrino interactions," *Z. Phys.*, vol. C41, p. 527, 1989.

[66] J. Bell *et al.*, "Cross section measurements for the reactions $\nu_\mu p \to \mu^- \pi^+ p$ and $\nu_\mu p \to \mu^- K^+ p$ at high energies," *Phys. Rev. Lett.*, vol. 41, p. 1008, 1978.

[67] T. Kitagaki *et al.*, "Charged current exclusive pion production in neutrino deuterium interactions," *Phys. Rev.*, vol. D34, pp. 2554–2565, 1986.

[68] P. Allen *et al.*, "Single π^+ production in charged current neutrino - hydrogen interactions," *Nucl. Phys.*, vol. B176, p. 269, 1980.

[69] P. Allen *et al.*, "A study of single meson production in neutrino and anti-neutrino charged current interactions on protons," *Nucl. Phys.*, vol. B264, p. 221, 1986.

[70] D. Allasia *et al.*, "Investigation of exclusive channels in neutrino / anti-neutrino deuteron charged current interactions," *Nucl. Phys.*, vol. B343, pp. 285–309, 1990.

[71] J. Campbell *et al.*, "Study of the reaction $\nu_\mu p \to \mu^- \pi^+ p$," *Phys. Rev. Lett.*, vol. 30, pp. 335–339, 1973.

[72] G. M. Radecky *et al.*, "Study of single pion production by weak charged currents in low energy neutrino - deuterium interactions," *Phys. Rev.*, vol. D25, pp. 1161–1173, 1982.

[73] D. Day *et al.*, "Study of neutrino - deuterium charged current two pion production in the threshold region," *Phys. Rev.*, vol. D28, pp. 2714–2720, 1983.

[74] T. Yang, C. Andreopoulos, H. Gallagher, and P. Kehayias, "A hadronization model for the MINOS experiment," *AIP Conf. Proc.*, vol. 967, pp. 269–275, 2007.

[75] H. Gallagher, "Using electron scattering data to tune neutrino Monte Carlos," *AIP Conf. Proc.*, vol. 698, pp. 153–157, 2004.

[76] S. Wood *http://hallcweb.jlab.org/resdata.*

[77] D. Bhattacharya, "Neutrino and antineutrino inclusive charged-current cross section measurement with the MINOS near detector," *FERMILAB-THESIS-2009-11.*

[78] M. R. Whalley, "A new neutrino cross section database," *Nucl. Phys. Proc. Suppl.*, vol. 139, pp. 241–246, 2005.

[79] C. Andreopoulos and H. Gallagher, "Tools for neutrino interaction model validation," *Nucl. Phys. Proc. Suppl.*, vol. 139, pp. 247–252, 2005.

[80] P. Adamson *et al.*, "A Study of Muon Neutrino Disappearance Using the Fermilab Main Injector Neutrino Beam," *Phys. Rev.*, vol. D77, p. 072002, 2008.

[81] K. Hiraide, "Measurement of Charged Current Charged Single Pion Production in SciBooNE," 2008.

[82] M. C. Sanchez, "Electron neutrino appearance in the MINOS experiment," *AIP Conf. Proc.*, vol. 981, pp. 148–150, 2008.

[83] D. Zieminska *et al.*, "Charged particle multiplicity distributions in neutrino n and neutrino p charged current interactions," *Phys. Rev.*, vol. D27, pp. 47–57, 1983.

[84] T. Sjostrand, S. Mrenna, and P. Skands, "PYTHIA 6.4 physics and manual," *JHEP*, vol. 05, p. 026, 2006.

[85] C. Andreopoulos, "The GENIE universal, object-oriented neutrino generator," *Acta Phys. Polon.*, vol. B37, pp. 2349–2360, 2006.

[86] Z. Koba, H. B. Nielsen, and P. Olesen, "Scaling of multiplicity distributions in high-energy hadron collisions," *Nucl. Phys.*, vol. B40, pp. 317–334, 1972.

[87] W. Wittek *et al.*, "Production of pi0 mesons and charged hadrons in anti- neutrino neon and neutrino neon charged current interactions," *Z. Phys.*, vol. C40, p. 231, 1988.

[88] S. Barlag *et al.*, "CHARGED HADRON MULTIPLICITIES IN HIGH-ENERGY anti-muon neutrino n AND anti-muon neutrino p INTERACTIONS," *Zeit. Phys.*, vol. C11, p. 283, 1982.

[89] M. Derrick *et al.*, "Properties of the Hadronic System Resulting from anti- Muon-neutrino p Interactions," *Phys. Rev.*, vol. D17, p. 1, 1978.

[90] A. M. Cooper-Sarkar, "Hadron final state results from BEBC," 1982. Invited talk presented at Neutrino '82, Balatonfured, Hungary, Jun 14-19, 1982.

[91] A. B. Clegg and A. Donnachie, "A description of jet structure by p_T limited phase space," *Zeit. Phys.*, vol. C13, p. 71, 1982.

[92] B. Andersson, G. Gustafson, G. Ingelman, and T. Sjostrand, "Parton Fragmentation and String Dynamics," *Phys. Rept.*, vol. 97, pp. 31–145, 1983.

[93] P. Allen *et al.*, "Multiplicity distributions in neutrino - hydrogen interactions," *Nucl. Phys.*, vol. B181, p. 385, 1981.

[94] A. Bodek and U. K. Yang, "Modeling neutrino and electron scattering cross sections in the few gev region with effective lo pdfs," 2003.

[95] A. A. Ivanilov *et al.*, "Inclusive γ and π^0 production in neutrino A and anti-neutrino A interacions up to 30 GeV," *Yad. Fiz.*, vol. 41, pp. 1520–1534, 1985.

[96] H. Grassler *et al.*, "Multiplicities of secondary hadrons produced in neutrino p and anti-neutrino p charged current interactions," *Nucl. Phys.*, vol. B223, p. 269, 1983.

[97] D. Allasia *et al.*, "Fragmentation in neutrino and anti-neutrino charged current interactions on proton and neutron," *Z. Phys.*, vol. C24, pp. 119–131, 1984.

[98] J. P. Berge *et al.*, "Inclusive Negative Hadron Production from High-Energy anti-neutrino Nucleus Charged Current Interactions," *Phys. Rev.*, vol. D18, p. 3905, 1978.

[99] V. Ammosov *et al.*, "A study of semi-inclusince gamma production in charged current anti-neutrino - nucleon interactions," *Nuovo Cim.*, vol. A51, p. 539, 1979.

[100] D. Drakoulakos *et al.*, "Proposal to perform a high-statistics neutrino scattering experiment using a fine-grained detector in the numi beam," 2004.

[101] P. Bosetti *et al.*, "Strange particle production in neutrino and anti-neutrino neon interactions," *Nucl. Phys.*, vol. B209, p. 29, 1982.

[102] N. J. Baker *et al.*, "Strange particle production in neutrino - neon charged current interactions," *Phys. Rev.*, vol. D34, pp. 1251–1264, 1986.

[103] D. DeProspo *et al.*, "Neutral strange particle production in neutrino and anti-neutrino charged current interactions on neon," *Phys. Rev.*, vol. D50, pp. 6691–6703, 1994.

[104] M. Derrick *et al.*, "Hadron production mechanisms in anti-neutrino - proton charged current interactions," *Phys. Rev.*, vol. D24, p. 1071, 1981.

[105] D. S. Baranov *et al.*, "An estimate for the formation length of hadrons in neutrino interactions," *PHE 84-04, 1984.*

[106] R. Merenyi *et al.*, "Determination of pion intranuclear rescattering rates in ν_μ Ne versus ν_μ D interactions for the atmospheric neutrino flux," *Phys. Rev.*, vol. D45, pp. 743–751, 1992.

[107] G. D. Harp, "Extension of the isobar model for intranuclear cascades to 1 GeV," *Phys. Rev.*, vol. C10, pp. 2387–2396, 1974.

[108] S. G. Mashnik, A. J. Sierk, K. K. Gudima, M. I. Baznat, and N. V. Mokhov *LANL Report LA-UR-05-7321 (2005), RSICC Code Package PSR-532.*

[109] S. G. Mashnik, A. J. Sierk, K. K. Gudima, and M. I. Baznat, "CEM03 and LAQGSM03: New modeling tools for nuclear applications," *J. Phys. Conf. Ser.*, vol. 41, pp. 340–351, 2006.

[110] S.G.Mashnik, "Private communication,"

[111] A. Engel, W. Cassing, U. Mosel, M. Schafer, and G. Wolf, "Pion - nucleus reactions in a microscopic transport model," *Nucl. Phys.*, vol. A572, pp. 657–681, 1994.

[112] G. Battistoni *et al.*, "The FLUKA code: Description and benchmarking," *AIP Conf. Proc.*, vol. 896, pp. 31–49, 2007.

[113] L. L. Salcedo, E. Oset, M. J. Vicente-Vacas, and C. Garcia-Recio, "COMPUTER SIMULATION OF INCLUSIVE PION NUCLEAR REACTIONS," *Nucl. Phys.*, vol. A484, p. 557, 1988.

[114] D. Ashery *et al.*, "True absorption and scattering of pions on nuclei," *Phys. Rev.*, vol. C23, pp. 2173–2185, 1981.

[115] I. Navon *et al.*, "True absorption and scattering of 50 MeV pions," *Phys. Rev.*, vol. C28, p. 2548, 1983.

[116] C. H. Q. Ingram *et al.*, "Measurement of quasielastic scattering of pions from $O16$ at energies around the $\Delta(1232)$ resonance," *Phys. Rev.*, vol. C27, pp. 1578–1601, 1983.

[117] J. D. Zumbro *et al.*, "Inclusive scattering of 500-MeV pions from carbon," *Phys. Rev. Lett.*, vol. 71, pp. 1796–1799, 1993.

[118] B. Kotlinski *et al.*, "Pion absorption reactions on N, Ar and Xe," *Eur. Phys. J.*, vol. A9, pp. 537–552, 2000.

[119] R. A. Arndt, W. J. Briscoe, I. I. Strakovsky, and R. L. Workman, "Extended Partial-Wave Analysis of piN Scattering Data," *Phys. Rev.*, vol. C74, p. 045205, 2006.

[120] G. web site *http://gwdac.phys.gwu.edu.*

[121] S. G. Mashnik, R. J. Peterson, A. J. Sierk, and M. R. Braunstein, "Pion induced transport of pi mesons in nuclei," *Phys. Rev.*, vol. C61, p. 034601, 2000.

[122] A. S. Carroll *et al.*, "Pion-Nucleus Total Cross-Sections in the (3,3) Resonance Region," *Phys. Rev.*, vol. C14, pp. 635–638, 1976.

[123] A. S. Clough *et al.*, "Pion-Nucleus Total Cross-Sections from 88-MeV to 860- MeV," *Nucl. Phys.*, vol. B76, p. 15, 1974.

[124] W. Bauhoff *At. Data and Nucl. Data Tables*, vol. 35, p. 429, 1986.

[125] A.K.Ichikawa *et al.*, "Private communication,"

[126] R.Hatcher, M.Messier, *et al.*, "Private communication,"

[127] G. D. Barr, T. K. Gaisser, P. Lipari, S. Robbins, and T. Stanev, "A three-dimensional calculation of atmospheric neutrinos," *Phys. Rev.*, vol. D70, p. 023006, 2004.

[128] G. Battistoni, A. Ferrari, T. Montaruli, and P. R. Sala, "Progresses in the validation of the FLUKA atmospheric nu flux calculations," *Nucl. Phys. Proc. Suppl.*, vol. 110, pp. 336–338, 2002.