A Generalized Hadronic Tensor Framework for GENIE

DRAFT

S. Gardiner[*]

August 29, 2019

1 Introduction

In this note, I present a new interface for models of lepton-nucleus scattering to be incorporated into the GENIE event generator. It is largely based on the approach used to add the Valencia MEC model to GENIE for the 2.12.6 release [1], but it generalizes and improves upon the methods used to compute differential cross sections therein. Like the original Valencia MEC implementation (which has been refactored to use the general interface), the phase space used for cross section calculations within the framework includes lepton kinematic variables only. The new interface is therefore most appropriate for use in cases where a theory calculation predicts only the leptonic final state (e.g., current MEC models in GENIE [2]) or where a lepton-only implementation can serve as a precursor to (and validation tool for) a more detailed future treatment of a particular model.

1.1 Formalism

The new cross section interface makes use of an object $W^{\mu\nu}$ called the *hadronic tensor*, which may be used to write lab-frame differential cross sections in the form

$$\frac{d\sigma}{dE_{\mathbf{k}'}d\Omega_{\mathbf{k}'}} = \frac{\mathcal{C}}{\pi^2}\frac{|\mathbf{k}'|}{|\mathbf{k}|}L_{\mu\nu}\,W^{\mu\nu}. \tag{1}$$

Here $\mathbf{k}$ ($\mathbf{k}'$) is the 3-momentum of the initial (final) lepton, $E_{\mathbf{k}'}$ is the final lepton total energy, $\Omega_{\mathbf{k}'}$ is the scattering solid angle, and the coupling factor $\mathcal{C}$ is defined by

$$\mathcal{C} \equiv \begin{cases} \frac{1}{2}\,G_F^2\,|V_{\text{ud}}|^2 & \text{CC processes} \\ \frac{1}{2}\,G_F^2 & \text{NC processes} \\ \frac{\alpha^2}{q^4} & \text{EM processes} \end{cases} \tag{2}$$

where G_F is the Fermi constant, V_{ud} is the CKM matrix element connecting the up and down quarks, α is the fine-structure constant, and q is the 4-momentum transfer. The leptonic tensor $L_{\mu\nu}$ may be written as

$$L_{\mu\nu} \equiv \frac{1}{8}\,\text{Tr}\left[\tilde{\gamma}_\mu(\not{k} + m_\ell)\tilde{\gamma}_\nu(\not{k}' + m_\ell')\right] \tag{3}$$

where

$$\tilde{\gamma}_\mu \equiv \begin{cases} \gamma_\mu(1-\gamma_5) & \text{CC and NC processes} \\ \gamma_\mu & \text{EM processes} \end{cases} \tag{4}$$

[*]mailto:gardiner@fnal.gov

and m_ℓ (m'_ℓ) is the mass of the initial (final) lepton. The hadronic tensor may be written in the form[1][2] [3]

$$W^{\mu\nu} = \frac{1}{2M_i}\overline{\sum_f}(2\pi)^3\,\delta^{(4)}(P'_f - P - q)\langle f|j^\mu(0)|i\rangle\langle f|j^\nu(0)|i\rangle^* \tag{5}$$

where P (M_i) is the 4-momentum (mass) of the initial target nucleus, P'_f is the total 4-momentum of the final-state hadronic system, and j^μ is the appropriate current operator for the scattering process of interest. The bar over the sum represents an average over the orientations of the initial nuclear spin, and the sum over final spins includes an implied integral over $\frac{d^3\mathbf{p}_j}{(2\pi)^3 2E_j}$ for each component particle j of the final-state hadronic system f.

Since the hadronic degrees of freedom are "integrated out" of the expression for $W^{\mu\nu}$ given in eq. (5), the hadronic tensor depends only on the 4-momentum transfer q. Following the conventions of ref. [3], I choose to work in the laboratory frame[3] with coordinates chosen so that the 3-momentum transfer $\mathbf{q}$ points along the positive z direction. With this choice of frame, only five unique real-valued components of the hadronic tensor are needed to compute neutrino-nucleus cross sections: W^{00}, $\mathrm{Re}(W^{0z})$, W^{xx}, $\mathrm{Im}(W^{xy})$, and W^{zz}. These reduce to two, W^{00} and W^{xx}, in the case of electromagnetic scattering cross sections. Note that, since $W^{\mu\nu}$ is Hermitian, all diagonal elements $W^{\mu\mu}$ are real.

The neutrino-nucleus differential cross section may now be written in the form [3, 4]

$$\frac{d\sigma}{dE_{\mathbf{k}'}\,d\cos\theta_{\mathbf{k}'}} = \frac{4\,|\mathbf{k}'|\,E_{\mathbf{k}'}\,\mathcal{C}}{\pi}\left[B_1 + \frac{{m'_\ell}^2}{E_{\mathbf{k}'}(E_{\mathbf{k}'} + |\mathbf{k}'|)}\,B_2\right] \tag{6}$$

where

$$B_1 \equiv 2W_1\sin^2(\theta_{\mathbf{k}'}/2) + W_2\cos^2(\theta_{\mathbf{k}'}/2) \mp W_3(E_{\mathbf{k}} + E_{\mathbf{k}'})\,\sin^2(\theta_{\mathbf{k}'}/2) \tag{7}$$

$$\begin{aligned} B_2 \equiv{}& W_1\cos\theta_{\mathbf{k}'} - \frac{1}{2}W_2\cos\theta_{\mathbf{k}'} + \frac{1}{2}W_3\left[E_{\mathbf{k}'} + |\mathbf{k}'| - (E_{\mathbf{k}} + E_{\mathbf{k}'})\cos\theta_{\mathbf{k}'}\right] \\ &+ \frac{1}{2}W_4\left[{m'_\ell}^2\,\cos\theta_{\mathbf{k}'} + 2E_{\mathbf{k}'}(E_{\mathbf{k}'} + |\mathbf{k}'|)\,\sin^2(\theta_{\mathbf{k}'}/2)\right] - \frac{1}{2}W_5\left[E_{\mathbf{k}'} + |\mathbf{k}'|\right] \end{aligned} \tag{8}$$

and the upper (lower) sign in front of the W_3 term should be chosen for neutrinos (antineutrinos). The structure functions W_j may be expressed in terms of five hadronic tensor elements:

$$W_1 = \frac{W^{xx}}{2} \tag{9}$$

$$W_2 = \frac{1}{2}\left[W^{00} + W^{xx} + \frac{(q^0)^2}{|\mathbf{q}|^2}(W^{zz} - W^{xx}) - 2\frac{q^0}{|\mathbf{q}|}\,\mathrm{Re}(W^{0z})\right] \tag{10}$$

$$W_3 = \frac{\mathrm{Im}(W^{xy})}{|\mathbf{q}|} \tag{11}$$

$$W_4 = \frac{1}{2|\mathbf{q}|^2}(W^{zz} - W^{xx}) \tag{12}$$

$$W_5 = \frac{1}{|\mathbf{q}|}\left[\mathrm{Re}(W^{0z}) - \frac{q^0}{|\mathbf{q}|}(W^{zz} - W^{xx})\right]. \tag{13}$$

For electromagnetic lepton-nucleus scattering, the differential cross section is [5]

$$\frac{d\sigma}{dE_{\mathbf{k}'}\,d\cos\theta_{\mathbf{k}'}} = 2\pi\,\sigma_{\mathrm{Mott}}\left[\frac{q^4}{|\mathbf{q}|^4}\,W^{00} + \left(\frac{2\sin^2(\theta_{\mathbf{k}'}/2)}{\cos^2(\theta_{\mathbf{k}'}/2)} - \frac{q^2}{|\mathbf{q}|^2}\right)W^{xx}\right] \tag{14}$$

where the Mott cross section is given by

$$\sigma_{\mathrm{Mott}} = \frac{\alpha^2\,\cos^2(\theta_{\mathbf{k}'}/2)}{4E_{\mathbf{k}}^2\,\sin^4(\theta_{\mathbf{k}'}/2)}. \tag{15}$$

[1]The weak charged current operator j^μ used in the Valencia calculation includes a factor of $\cos\theta_c = V_{\mathrm{ud}}$, where θ_c is the Cabibbo angle. The factor of $|V_{\mathrm{ud}}|^2$ that appears in the CC value of $\mathcal{C}$ given in eq. (2) should therefore be omitted when working with precomputed hadronic tensor tables prepared for the Valencia model, since the tensor elements already include this factor.

[2]Here I choose states normalized so that $\langle\mathbf{p}|\mathbf{p}'\rangle = (2\pi)^3\,2p^0\,\delta^{(3)}(\mathbf{p} - \mathbf{p}')$. This normalization agrees with the conventions of the Valencia model.

[3]That is, the rest frame of the initial target nucleus

2 Implementation details

The hadronic tensor framework described in this note is implemented in `GENIE` using two abstract base classes: (1) `genie::HadronTensorI`, which represents the hadronic tensor $W^{\mu\nu}$ for a particular reaction mode and target nucleus, and (2) `genie::HadronTensorModelI`, a `genie::Algorithm` which provides an XML-configurable interface to a set of hadronic tensors to be used by a cross section model. The source files defining these interfaces and their derived classes may be found in a new subfolder of the Generator source code repository: `src/Physics/HadronTensors`.

2.1 HadronTensorI and derived classes

Figure 1 shows the relationships between the abstract base class `HadronTensorI` and its derived classes. All `HadronTensorI` objects have member functions that compute each of the sixteen hadronic tensor elements, with `tt()` returning W^{00}, `xy()` returning W^{xy}, etc. A complex number is needed to represent the off-diagonal elements of the tensor in the general case, so each of these functions returns a `std::complex<double>`. Pure virtual member functions are also provided to compute the quantity $L_{\mu\nu}W^{\mu\nu}$ (`contraction()`), and to indicate the range of validity of the tensor calculation along the q^0 (`q0Min()`, `q0Max()`) and $|\mathbf{q}|$ (`qMagMin()`, `qMagMax()`) axes. The only data member necessarily shared by all `HadronTensorI` objects is `fTargetPDG`, which is the Particle Data Group (PDG) code for the target nucleus represented by the tensor.

The `ValenciaHadronTensorI` inherits from `HadronTensorI` but remains an abstract class. In addition to enforcing some of the conventions of the Valencia model in the hadron tensor elements (e.g., Hermiticity and choice of coordinates, as discussed in section 1.1), it also defines a member function `dSigma_dT_dCosTheta()` which computes the differential cross section $d\sigma/dE_{\mathbf{k}'}\,d\cos\theta_{\mathbf{k}'}$ using the expressions given in this note. The first of two required inputs to this function is a pointer to an `Interaction` object (with the `Tl` and `ctl` kinematic variables set). The second is a “Q-value” used to adjust the value of q^0 to account for binding energy effects. In some models, the hadronic tensor should be evaluated for $\tilde{q}^0$ instead of $q^0 = E_{\mathbf{k}} - E_{\mathbf{k}'}$, where

$$\tilde{q}^0 \equiv q^0 - Q_{\text{val}} \tag{16}$$

and Q_{val} is the Q-value. For models that do not perform this sort of correction, a value of $Q_{\text{val}} = 0$ should be passed to `dSigma_dT_dCosTheta()`.

The `TabulatedValenciaHadronTensor` class is currently the sole concrete implementation of the `HadronTensorI` interface. It inherits from `ValenciaHadronTensorI` and computes tensor elements using bilinear interpolation between precomputed values specified on a 2D grid in (q^0, $|\mathbf{q}|$) space. While much of the implementation is directly adapted from the original Valencia MEC code, two improvements in the current approach are worth discussing in this note. First, format of the data files used to define the precomputed tensor values is now self-describing. Second, the bilinear interpolation is now done more efficiently thanks to a new template class called `BLI2DNonUnifObjectGrid`. The next two subsections briefly describe each of these changes.

2.1.1 Input file format used by TabulatedValenciaHadronTensor

For notational simplicity in this subsection, I will denote the energy transfer q^0 and the magnitude of the 3-momentum transfer $|\mathbf{q}|$ by ω and κ, respectively.

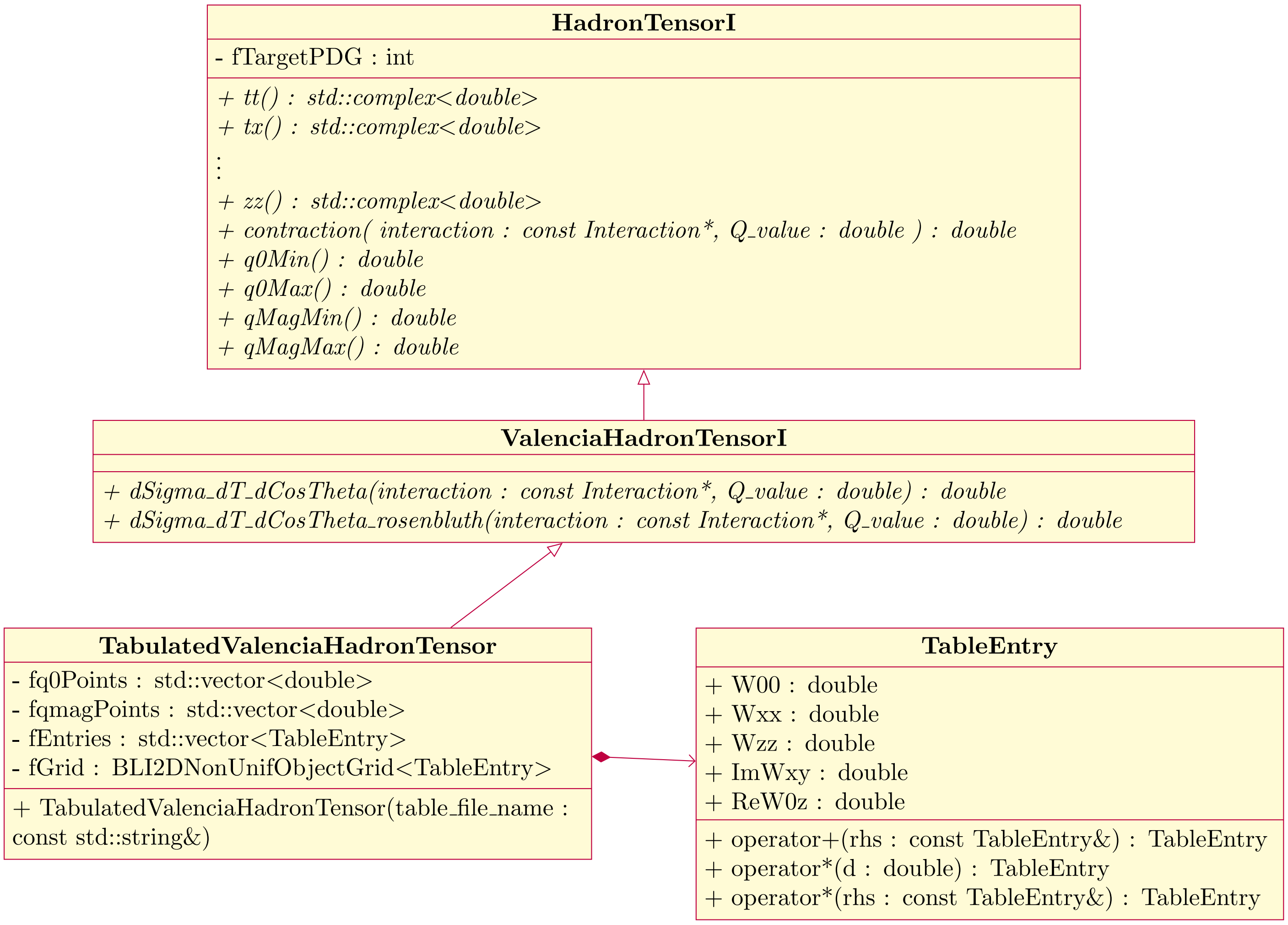
Precomputed elements of the hadronic tensor $W^{\mu\nu}$ for a particular model and reaction mode (EMQE, CCMEC, etc.) are provided to the `TabulatedValenciaHadronTensor` class in the form of a whitespace-delimited text file that begins a comment line followed by the header

```
Z A type m n
```

in which `Z` (`A`) is the proton (mass) number of the target nucleus, `type` is a label (an arbitrary string which contains no whitespace) describing the type of nuclear tensor represented by the file, and `m` (`n`) is the number of grid points used along the ω (κ) axis. This header is followed by specifications of the ω and κ grid in the form

```
w_flag w_spec
k_flag k_spec
```

Figure 1: UML class diagram for HadronTensorI and its derived classes. Not all class members are shown.


HadronTensorI
- fTargetPDG : int
+ tt() : std::complex<double>
+ tx() : std::complex<double>
⋮
+ zz() : std::complex<double>
+ contraction(interaction : const Interaction*, Q_value : double) : double
+ q0Min() : double
+ q0Max() : double
+ qMagMin() : double
+ qMagMax() : double
ValenciaHadronTensorI
+ dSigma_dT_dCosTheta(interaction : const Interaction*, Q_value : double) : double
+ dSigma_dT_dCosTheta_rosenbluth(interaction : const Interaction*, Q_value : double) : double
TabulatedValenciaHadronTensor
- fq0Points : std::vector<double>
- fqmagPoints : std::vector<double>
- fEntries : std::vector<TableEntry>
- fGrid : BLI2DNonUnifObjectGrid<TableEntry>
+ TabulatedValenciaHadronTensor(table_file_name : const std::string&)
TableEntry
+ W00 : double
+ Wxx : double
+ Wzz : double
+ ImWxy : double
+ ReW0z : double
+ operator+(rhs : const TableEntry&) : TableEntry
+ operator*(d : double) : TableEntry
+ operator*(rhs : const TableEntry&) : TableEntry

where `w_flag` and `k_flag` are integer codes that define the representation used for the ω and κ grids, respectively. Two values are currently allowed for these flags. For `w_flag` = 0, a regular grid of ω values is used, and `w_spec` has the form

```
w_start w_step
```

where `w_start` is the smallest tabulated value of ω and `w_step` is the step size between adjacent grid points. For `w_flag` = 1, the ω grid is given explicitly in the file and may have irregular spacing between points. In this case, `w_spec` contains a white-space delimited list of `m` strictly increasing ω values. The κ grid is specified using an identical format for `n` grid points. All numerical values of ω and κ used to define the grid should be given in GeV to match the `GENIE` convention for natural units.

Following the header, the elements of the nuclear tensor are tabulated for each grid point. The five elements are tabulated for each grid point in order of increasing ω and κ, with κ increasing more rapidly. Thus, the tensor elements appear in the file in the order

$$\begin{array}{cccc} \mathbf{W}(\omega_1,\kappa_1) & \mathbf{W}(\omega_1,\kappa_2) & \ldots & \mathbf{W}(\omega_1,\kappa_n) \\ \mathbf{W}(\omega_2,\kappa_1) & \mathbf{W}(\omega_2,\kappa_2) & \ldots & \mathbf{W}(\omega_2,\kappa_n) \\ & & \vdots & \\ \mathbf{W}(\omega_m,\kappa_1) & \mathbf{W}(\omega_m,\kappa_2) & \ldots & \mathbf{W}(\omega_m,\kappa_n) \end{array}$$

where ω_i is the ith grid point on the ω axis, κ_j is the jth grid point on the κ axis, and each of the $\mathbf{W}(\omega,\kappa)$ represents the sequence of real-valued tensor elements

$$W^{00}(\omega,\kappa) \quad \mathrm{Re}\left[W^{0z}(\omega,\kappa)\right] \quad W^{xx}(\omega,\kappa) \quad \mathrm{Im}\left[W^{xy}(\omega,\kappa)\right] \quad W^{zz}(\omega,\kappa)$$

given in GeV^{-1} as floating-point numbers.

2.1.2 The BLI2DNonUnifObjectGrid template class

In the original hadronic tensor implementation for the Valencia MEC model [1], the five tensor components needed to compute cross sections were each interpolated individually using separate `BLI2DGrid` objects. Because the same 2D grid is used to specify all five tensor components, this results in four redundant grid searches (over potentially many grid points) in order to complete a single cross section evaluation.

To avoid this inefficiency, I created a new template class called `BLI2DNonUnifObjectGrid`. This class performs bilinear interpolation in essentially the same way as the existing `BLI2DNonUnifGrid` class, but the type used to evaluate the z coordinate at each grid point is now an arbitrary C++ object (whose type is taken as a template parameter `ZObject`) instead of a `double`. The only restriction on the `ZObject` type used by `BLI2DNonUnifObjectGrid` is that it must implement the member functions `operator*(double)` (scalar multiplication), `operator*(const Object&)` (vector dot product), and `(operator+(const Object&)` (vector addition).

To evaluate tensor elements, the `TabulatedValenciaHadronTensor` class instantiates a `BLI2DNonUnifObjectGrid` using objects of type `TabulatedValenciaHadronTensor::TableEntry` (abbreviated from now on as `TableEntry`) to represent the z coordinate. Each `TableEntry` object contains a set of the five tensor elements labeled **W** in section 2.1.1. When the bilinear interpolation is performed by `BLI2DNonUnifObjectGrid`, all five are evaluated simultaneously, avoiding redundant grid lookups.

Taking advantage of some modern features of the C++ standard library (but not yet venturing into C++11), the `BLI2DNonUnifObjectGrid` class stores grid points using vectors instead of C-style arrays, and it relies on `std::lower_bound()` to search the grid instead of the manual linear search used by the existing `BLI2DGrid` objects.

2.2 HadronTensorModelI and its derived classes

In the new hadronic tensor framework, cross section models (represented in `GENIE` by `XSecAlgorithmI` objects) are given access to the tensors needed to perform calculations by configuring an instance of `HadronTensorModelI` as a subalgorithm. An inheritance diagram for this abstract base class is shown in fig. 2.

The `HadronTensorModelI` interface adds a single member function called `GetTensor()` to the `Algorithm` class. This function takes two arguments: the PDG code for the target nucleus of interest and a code representing the kind of tensor (e.g., one for a CCMEC or EMQE reaction) to be retrieved. The available codes are tabulated as the enum type `HadronTensorType_t` in the `HadronTensorI.h` header file. Note that not all codes are used by all hadronic tensor models.

The `TabulatedHadronTensorModelI` abstract class inherits from `HadronTensorModelI` and may be used for cross section calculations that rely on precomputed tensor data files. Tensor objects needed for calculations are loaded lazily and cached in a `std::map` for subsequent retrieval. The keys of this map have type `HadronTensorID`, which is a simple struct combining the target PDG code and a `HadronTensorType_t` enum variable. The XML configuration parameters used to specify file search paths and file names for each tensor are described in the following subsection. The actual parsing of the file is handled by the `ParseTensorFile()`, which is left as a pure virtual function in `TabulatedHadronTensorModelI`. Each of the existing concrete subclasses of `TabulatedHadronTensorModelI` (`NievesMECHadronTensorModel`, `SuSAv2MECHadronTensorModel`, and `SuSAv2QELHadronTensorModel`) implements this function by simply constructing a new `TabulatedValenciaHadronTensor` object to be stored in the cache.

2.2.1 TabulatedHadronTensorModelI XML configuration

The names of the XML parameters used to configure an instance of `TabulatedHadronTensorModelI` are

DataPath A string specifying a folder in which to search for hadronic tensor data files

DataPathType A string describing how **DataPath** should be interpreted. Allowed values are currently `"relative"` (the path is given relative to the `$GENIE` folder) and `"absolute"` (the path is an absolute system path).

Type@Pdg=Target A string giving the file name (without the path prepended) to load for a given `HadronTensorType_t` (**Type**) and target nucleus PDG code (**Target**). For example, `MEC_EM@Pdg=1000060120` is the parameter name that should be used to specify a file to load for a hadronic tensor representing an EMMEC reaction on ^{12}C. The strings representing each `HadronTensorType_t` are given in the function `string_to_tensor_type()` in `TabulatedValenciaHadronTensorModelI.cxx`.

See the files `config/NievesMECHadronTensorModel.xml` and `config/SuSAv2QELHadronTensorModel.xml` for concrete examples of a working XML configuration.

3 Validation

Some basic checks of the refactored Valencia MEC model against the original implementation have been done, with more planned for the near future. As an example, section 3 shows close agreement between total CCMEC cross section splines calculated using both versions of the code. The small discrepancies that do exist are attributable to removal of some hard-coded constants and differences in the order of operations for the bilinear interpolation (which of the two axes is used first).

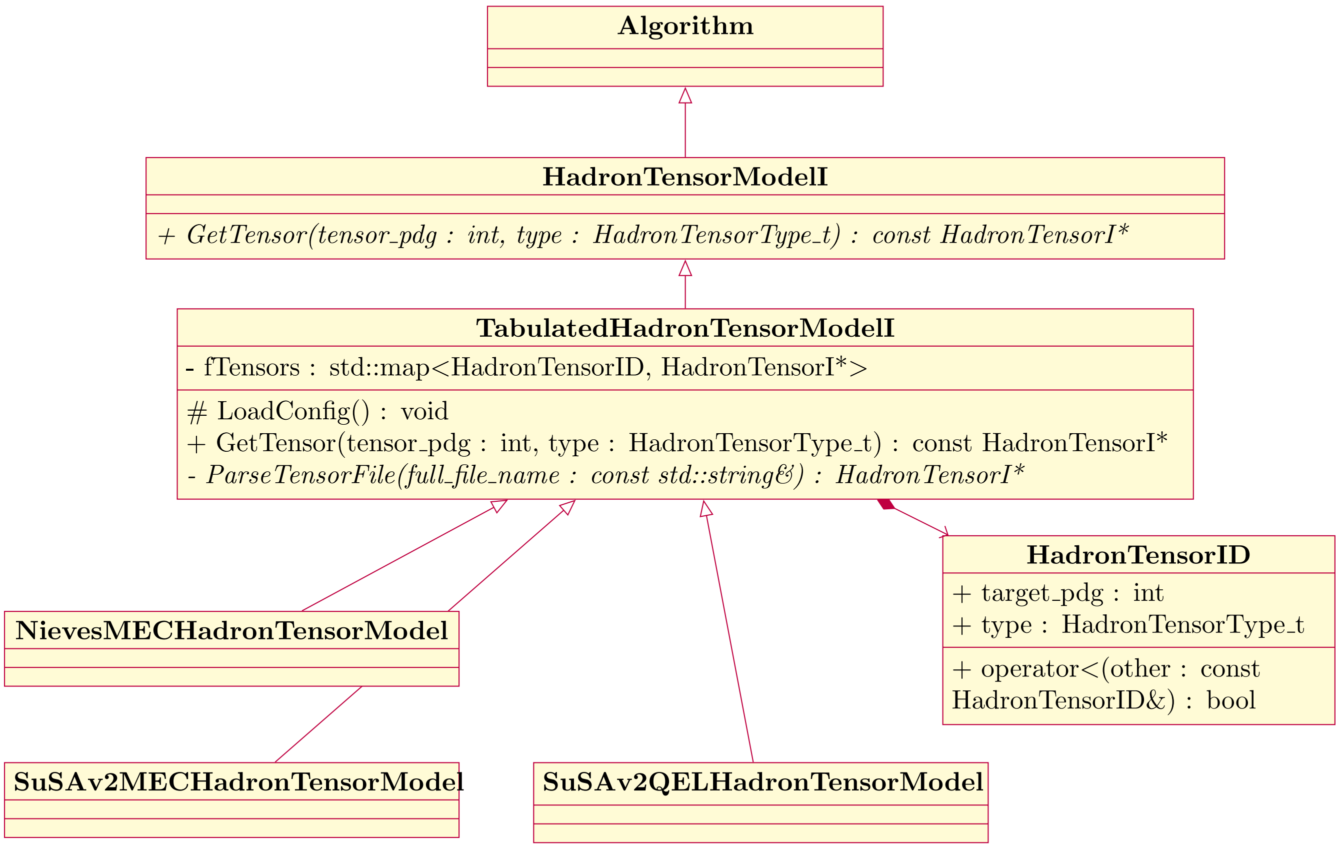
This framework has also been extensively exercised as the SuSAv2 model implementation was being validated against the original calculations. For more details on those results, see ref. [6].

4 Open Issues

4.1 Free Nucleon Cross Sections

For the case of QE scattering on a free nucleon target, there is only a single hadron in the final state. As discussed in section 1.1 with reference to eq. (5), this implies that only a single integration over $\frac{d^3\mathbf{p}'}{(2\pi)^3 2E_{\mathbf{p}'}}$ (where $\mathbf{p}'$ is the outgoing nucleon's 3-momentum) appears in the sum over final states used to evaluate the hadronic tensor. An integration over one of the kinematic variables describing the outgoing lepton (e.g., $dE_{\mathbf{k}'}$) must therefore be performed in order to eliminate the energy-conserving delta function from the expression

Figure 2: UML class diagram for HadronTensorModelI and its derived classes. Not all class members are shown.


Algorithm
HadronTensorModelI
+ GetTensor(tensor_pdg : int, type : HadronTensorType_t) : const HadronTensorI*
TabulatedHadronTensorModelI
- fTensors : std::map<HadronTensorID, HadronTensorI*>
# LoadConfig() : void
+ GetTensor(tensor_pdg : int, type : HadronTensorType_t) : const HadronTensorI*
- ParseTensorFile(full_file_name : const std::string&) : HadronTensorI*
HadronTensorID
+ target_pdg : int
+ type : HadronTensorType_t
+ operator<(other : const HadronTensorID&) : bool
NievesMECHadronTensorModel
SuSAv2MECHadronTensorModel
SuSAv2QELHadronTensorModel

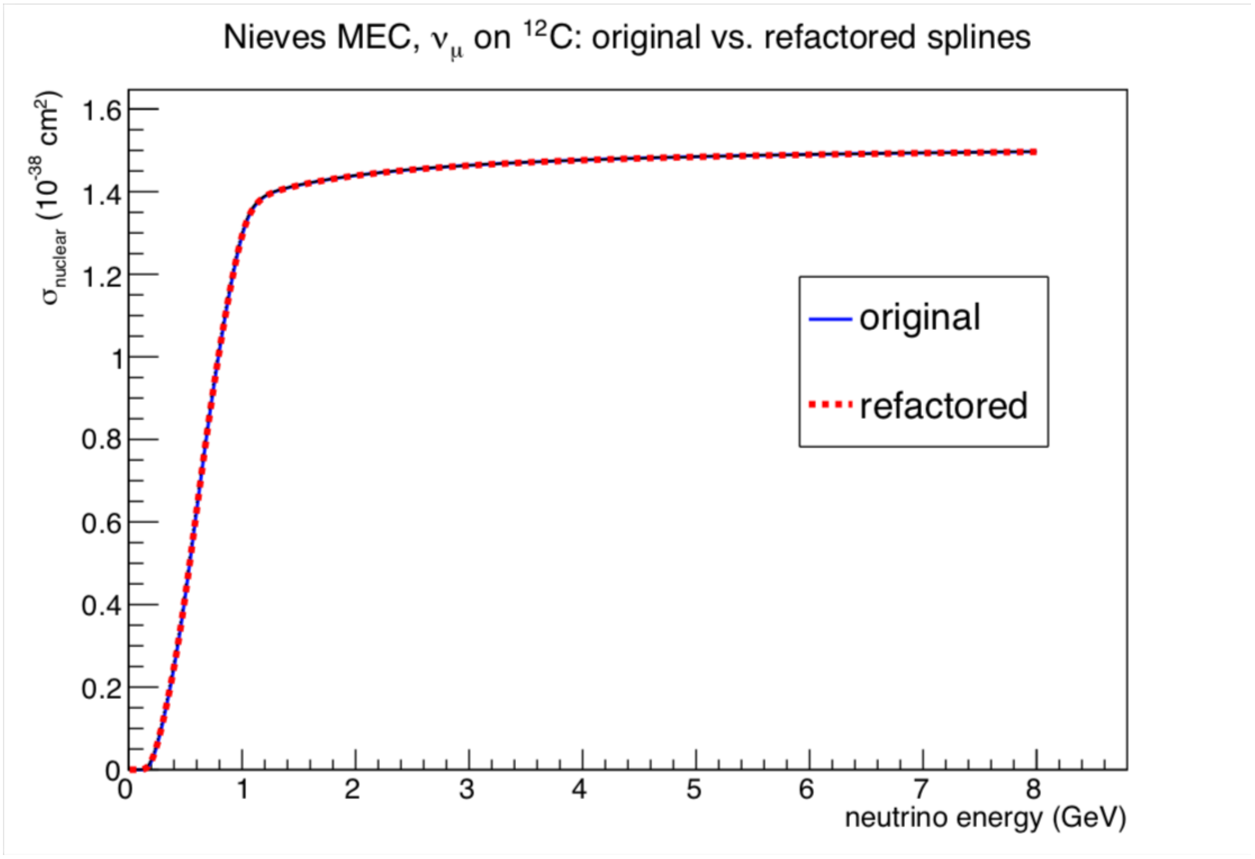


Figure 3: Comparison of CCMEC splines generated for the Valencia MEC model using the original (blue) and refactored (red, dashed) implementations

for the differential cross section. This renders eq. (1) inappropriate to use for simulating lepton-nucleon scattering, since the outgoing lepton energy and scattering angles are all treated as free parameters.

For the case of SuSAv2 QE event generation, then, a strategy other than the one presented here should be used for free nucleon targets. One proposal is to use the Llewellyn-Smith model (with form factors chosen to match the SuSAv2 conventions) to generate quasielastic events using standard `GENIE` methods for the free nucleon case. Doing so, however, would require making the event generation chain for QE target-dependent, which appears to be difficult to do within the current `GENIE` framework.

4.2 Duplicate cross section functions

As shown in fig. 1, the `ValenciaHadronTensorI` class has two member functions which both compute $d\sigma/dE_{\mathbf{k}'}\, d\cos\theta_{\mathbf{k}'}$, one using the expressions shown in this note, and the other which uses corresponding expressions from the Rosenbluth formalism. The Rosenbluth functions were added during implementation of the SuSAv2 model [6] to ease debugging. However, the two forms of the cross section should be equivalent. Is there any value in keeping both functions? Or should we remove one version in favor of a more unified framework?

References

[1] J. Schwehr, D. Cherdack and R. Gran, *GENIE implementation of IFIC Valencia model for QE-like 2p2h neutrino-nucleus cross section*, `1601.02038`.

[2] T. Katori, *Meson exchange current (MEC) models in neutrino interaction generators*, *AIP Conf. Proc.* **1663** (2013) 030001 [`1304.6014`].

[3] J. Nieves, J. E. Amaro and M. Valverde, *Inclusive quasielastic charged-current neutrino-nucleus reactions*, *Phys. Rev. C* **70** (2004) 055503 [`nucl-th/0408005`].

[4] J. Nieves, J. E. Amaro and M. Valverde, *Erratum: Inclusive quasielastic charged-current neutrino-nucleus reactions [Phys. Rev. C 70, 055503 (2004)]*, *Phys. Rev. C* **72** (2005) 019902.

[5] A. Gil, J. Nieves and E. Oset, *Many-body approach to the inclusive (e, e′) reaction from the quasielastic to the Δ excitation region*, *Nucl. Phys. A* **627** (1997) 543.

[6] S. Dolan, G. Megias and S. Bolognesi, *Implementation of the SuSAv2-MEC 1p1h and 2p2h models in GENIE and analysis of nuclear effects in T2K measurements*, `1905.08556`.