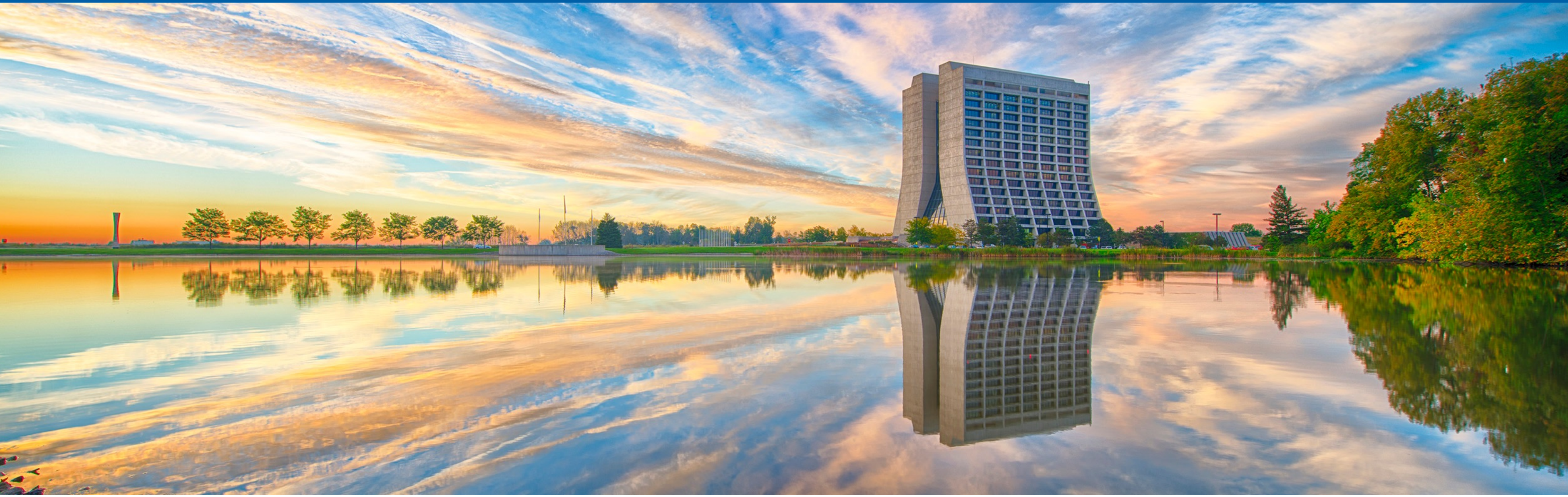

Summary of Theory API discussion

Steven Gardiner

ECT Workshop on Testing and Improving Models of Neutrino Nucleus Interactions in Generators

June 2019

Why make a theory API for neutrino generators?

- One way of **addressing the organizational "pain points"** mentioned in Laura's plenary talk on Monday
  - Lag between theory development & generator inclusion
  - Road for theorists to contribute to generators may seem unclear to them and be poorly incentivized (citations!)
  - Limited number of generator authors ⟶ limited bandwidth for model inclusion
- Another thought (pointed out by P. Machado)
  - Growing theory interest in looking at BSM physics in neutrino experiments
  - Many models, unlikely to be a priority for generator developers
  - Some tinkering has already been done, but a real API would greatly facilitate this work

Factorizable hadron tensor interface

- One strategy for speeding up inclusion of theory models in generators

- Relies on precomputed a nuclear tensor, with elements interpolated on a (q0, q3) grid $d\sigma \propto L_{\mu\nu} W^{\mu\nu}$

- SuSAv2 calculation (QE + MEC) adapted for GENIE

  - Existing Valencia MEC implementation also restructured to use general interface
  - Demonstration that theory groups are able and willing to provide needed inputs

- **Explored in-depth at this workshop (WG 3.3, see S. Dolan's talk later today)**

- Some (current) limitations:
  - lepton kinematics only
  - no ability to tune / reweight parameters used to make tables

- Need to "invent" hadron kinematics could be mitigated by including more tabulated responses (15 instead of 5), at least for single nucleon knockout

- Theory API group explored what other approaches might be possible
  - **Not mutually exclusive**

One option: interface for N-fold differential cross sections

$$\frac{d^n\sigma}{dX^n}$$

- Primarily discussed via email in advance of the workshop
- Object of interest is a differential cross section in an arbitrary phase space
  - Code specifies the variables to be thrown, allowed ranges
  - Function takes those values as input (with other needed information, e.g., neutrino 4-momentum), returns cross section
  - Kinematics sampled, used to construct outgoing 4-momenta
    - Some cross section calculations are already generators themselves
    - In principle, could handle everything pre-FSI in contributed code and pass result into event record
  - FSI model then takes over

One option: interface for N-fold differential cross sections

$$\frac{d^n\sigma}{dX^n}$$

- Very general, but this comes at the price of complexity
  - If we ask theorists for too much, we may not get anything!
- GENIE already has this in some sense
  - To get results from an external cross section function, the existing interface would just need to be exposed
  - However, each cross section requires a custom generator module (samples kinematics, finds maximum for rejection sampling, etc.)
  - We don't mind having large tables / high-dimensional phase spaces. Physics payoff is worth it!

A middle way?

- Luis proposed a third option in our parallel session
  - Extensive discussion about what models / effects can be represented

- Rather than providing integrated nuclear responses (as in the factorizable hadron tensor interface), express the cross section in terms of three objects:

$$d\sigma \propto L_{\mu\nu} W^{\mu\nu}$$

  - A **hadronic** (as opposed to **nuclear**) tensor
  - A particle spectral function
  - A hole spectral function

$$d\sigma \propto \int d^4p \, H^{\mu\nu} \, A_{\mathsf{fs}}(p+q) \, A_{\mathsf{h}}(p)$$

- Still quite general (N, NN, Nπ, etc.)
  - All pieces not necessarily provided by all calculations, e.g., $A_{\mathsf{fs}}(p+q)$ typically handled by intranuclear cascade
  - Complexity grows rapidly with number of particles produced

- Phase space determined by the channel represented by $H^{\mu\nu}$
  - Throw kinematics, pass 4-momenta needed to calculate these objects to external code
  - Use results to evaluate differential cross section

Takeaways from the parallel session discussion

- **Some clear consensus in the room on some issues, but less on two important questions**

  1. Is a theory interface of this kind something that we want to prioritize as a community, or is the model incorporation “pain point” status quo acceptable?

  2. What common language (differential cross section, hadron tensor, etc.) is needed to describe and implement our models of interest most easily?

- Some aspects of the more technical discussion have relevance to these questions. I will revisit them in my closing slides

- While some details still need to be worked out, the technical implementation of such an interface appears feasible.

- **The challenge lies more in defining our requirements clearly than in executing them**
  - This requires theory / experiment collaboration

- I will share a few highlights from the discussion on topics that appeared to converge

Form of the “theory API”

- Goal is a method for facilitating inclusion of models in generators, but despite the label “theory API,” this could take at least two different forms:
  - Specification that each participating generator implements separately
    - “Here are the functions we need theorists to write, with their inputs and outputs. Our code will then talk to yours.”
    - Uniform interface makes new models easy to include
  - Standalone tool (NUISANCE-like) that serves as an intermediary between models and multiple generators
    - Team behind this tool maintains the communication at both ends
  - Something else?
- **Clear preference for the “specification” option**
  - Generator talks directly to theory code, each group maintains its piece
  - “Opt-in” participation: generators can decide if it’s worth the effort to add the interface

The programming language divide

- C++ is the “lingua franca” among HEP experimentalists
- Fortran is often used by theorists, and at least two modern neutrino generators (NEUT & GiBUU) use it as well
- Interoperability between these two languages (at least) seems like a necessary requirement for a truly “universal” theory API
  - There may be other languages (Python? C?) of interest
- **Modern compilers make language interoperability straightforward**
  - Hayato-san: “As for NEUT, we don't care whether the code is written in Fortran or C or C++ as far as they are not using the ‘latest’ features only in the latest standards like c++14 or Fortran2018 etc.”
- No need to translate code as long as interfaces are well-defined

Code distribution & citations

- **How should the contributed theory code be shared with interested users?**
  - Packaged together with official releases of the generators?
  - Download links on theorists' websites?
- **Proposed method:**
  - Fermilab prepares a website to serve as a central hosting location for theory contributions
  - Model "plugins" downloaded with installation instructions for compatible generators
  - Each model labeled with a **version number** and a **paper that should be cited**
    - Changes can be made while giving experimentalists an unambiguous way of knowing what was used in their simulations
- The website would need a permanent name (preferably catchy!)

Validation

- One of the rate-limiting steps in the current approach to adding models is validation
  - Easy to make mistakes when adapting theory code
- For GENIE, obtaining access to theorists' original code has served us well
  - Run the theorists' implementation and GENIE's
  - The two should be **very close** if we've done our job right
  - Ensures an accurate implementation, but time-consuming
- Table-based strategies (e.g., hadronic tensor interface) still need to be compared to full calculation, ideally the original code used to produce them
- API to directly interface with theory code helps to address this problem
- Testing functions could be included as part of the interface
  - Sanity checks to perform as part of installation (is everything built and configured properly?)
  - Breaking changes to dependencies (e.g., ROOT) could cause problems. This is one way to catch them

Model configuration & uncertainties

- **Important part of the recipe for generators**
  - Experiments need to assess their systematic uncertainties associated with the cross section models in their simulations
  - Typically done via reweighting
  - **Theory guidance on uncertainties valuable! We could use a lot more of it.**
- Table-based approaches can deal with this in principle
  - Vary the model, provide a different table each time
  - Metadata needed to describe what table goes with which variation
  - **Number of tables can become unwieldy**
- **With a direct code interface, configuration of parameters can be done directly**
  - Model specifies a vector of input parameters with associated uncertainties
  - API allows generator to initialize those parameters appropriately (pass a vector of values)
- Some subtleties to consider (e.g., parameter consistency between channels)

Some final thoughts

- **Solidifying our understanding of what theorists are able & willing to provide is key**
  - Some helpful input in the parallel session, but we need more potential model contributors to make their preferences known
  - Is one of the options described in this talk appropriate for your model? Can we reasonably expect you to provide that representation for us?
  - Answer may be shaped by which theory groups engage
- **“Generator implements a spec” development model**
  - Lowers barrier for new theory API attempts
  - Different generators can decide how much of a priority this is
  - If one devotes some cycles to trying this for a group of models, open-source API and lessons learned can be shared with others

Some final thoughts

- **Proof of concept a useful next step**
  - Already exists for table-based nuclear tensors (Valencia MEC, SuSAv2 QE + MEC)
  - Candidates for the $H^{\mu\nu}$ approach might include 2-body SF treatment (N. Rocco)
    - CCQE in GENIE v3 handled in a similar way, but all internally to the generator
- **With one or more concrete examples, path to a written specification becomes more clear**