NuTools - Support #22584

Rescattering code

05/13/2019 01:00 PM - Gianluca Petrillo

Status:	Feedback	**Start date:**	05/13/2019
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour

Description

It is my understanding that the meaning of the rescattering code stored in simb::MCParticle depends on the generator filling it, and also on which of its available algorithms the generator is using.
How can I determine which is the actual meaning of the code, starting from a simb::MCTruth object?

History

#1 - 05/15/2019 12:37 PM - Robert Hatcher

- Status changed from New to Feedback

I believe what you are referring to here is

```
   int                     frescatter;     ///< rescatter code
```

in simb::MCParticle (this class is now in nusimdata, though currently filled by nutools (and possibly other) code)

By default when creating a simb::MCParticle it gets initialized to s_uninitialized which is defined as:

```
const int MCParticle::s_uninitialized = std::numeric_limits<int>::min();
```

As best I can determine, this member data exists only so that it can support a lossless representation of GENIE event record where particles are recorded as genie::GHepParticle which also have a "rescatter" code associated with them (albeit with -1 flagging "unset"). This allows them to be tagged with additional information of what sub-process might have created them. For instance, when they undergo a "final state interaction" (FSI, e.g. charge exchange, inelastic scattering, absorption ...) within the nuclear environment.

As far as I can ascertain **no** other source of simb::MCParticle in the LArSoft stack have need **or***make use* of this ability to add an extra tag to simb::MCParticles in MCTruth (the output of the event generator) or ParticleList (the output of the Geant4 stage) objects.

As for the interpretation, you're not meant to use them (as you say they can differ due to different FSI models being used by different tunes of GENIE). What they are used for is ***in the GENIE framework*** to support (some) particular reweightings of FSI processes. One does this by using nutools/GENIE2ART function:

```
namespace evgb {
genie::EventRecord* RetrieveGHEP(const simb::MCTruth& truth,
                                 const simb::GTruth&  gtruth,
                                 bool useFirstTrajPosition = true);
```

to reconstruct the genie::EventRecord and use the GENIE tools to do reweighting on that.

If it is actually meaningful and useful for users, one *could* request that GENIE make a breaking change to make the rescatter codes unique amongst the various FSI models, and documented. And then, with GENIE input, carve out some number space that others will respect if they were to ever attempt to use this member data for their own purposes.

#2 - 05/15/2019 04:25 PM - Gianluca Petrillo

- % Done changed from 0 to 100

I think your statement boils down to "those codes are for internal use by the GENIE-based reweighting framework", and I can only assume that framework to be able to resolve the ambiguity of their definition by some proper assumptions.

I also believe no LArSoft code uses it, except for the utility I was writing that dumps the code value on screen and leaves its interpretation to the user. I was hoping to be able to help in that interpretation, but it might be beyond the available information (unless GenieHelper only supports a single model?).
Nevertheless, I would not be highly surprised if somebody used them in their analysis code, maybe just to attempt to categorise events according to

what is believed to happen inside the event, with all the risks and caveat of drawing conclusions from a non-observable.

Thank you for the information.

#3 - 05/15/2019 05:46 PM - Robert Hatcher

I'm not sure what "utility I was writing that dumps the code value on screen" you're writing, but be aware for MCTruth/GTruth representations of GENIE events there exists a module that converts those back to genie::EventRecords and (optionally) prints them out. For any discussion of GENIE events I prefer to see events in that format rather than (yet-another-arbitrary) dumping scheme.

This capability is distributed as part of nutools and can be invoked via:

```
lar -n <nevents> -c dump_genie_job.fcl  <myfile.artgenie.root>
```

Re:

> Nevertheless, I would not be highly surprised if somebody used them in their analysis code, maybe just to attempt to categorise events according to what is believed to happen inside the event, with all the risks and caveat of drawing conclusions from a non-observable.

Yes, it's hard to stop users from doing "foolish" things. My concerns in this regard are less about these FSI rescatter "fates" than users over/misuse of the tempting flags: simb::MCNeutrino.fInteractionType (problematic, see redmine issue #18025 -- resurfaced recently and discussed in today's GENIE User Forum), simb::MCNeutrino.fMode, simb::GTruth.fGScatter and simb::GTruth.fGint

Again, if someone wants to make a formal request that GENIE standardize these FSI "fates" and document them, then we can hold that discussion. That's not unreasonable, though to make it truly foolproof we'd probably have to make a breaking change (but perhaps not -- I'm going to have very preliminary discussions with the primary author of that bit of code).