Adding New Models to GENIE

Gabriel N. Perdue
Fermilab

Plan

- Event Record Visitors
- Incomplete & Complete Models
- Example: Berger-Sehgal Coherent Pion
- How-To: NewModel package

First of all...

- Note that the trickiest part of implementing a new model is often understanding the physics! Make sure you have a reasonable idea of what you are computing before you begin work.
- Collect all relevant papers and, if possible, data sets.
- It is good practice to try to start building a validation application as soon as you can.

Event Record Visitors

- Implementation is straightforward as long as you understand the Visitor Pattern.
  - Essentially, you string together arbitrary sequences of classes implementing an interface that knows how to process an event record.
  - Each class in turn modifies the record and then makes it available to the next class in line.

Event Record Visitors

- Interface defined in 'src/EVGCore/EventRecordVisitorI.h'.
- One public method: `ProcessEventRecord(GHepRecord *event_rec) const;`
- Note that the method is `const` but the `GHepRecord` is not – the expectation is the method will update the record.
- A given model will be named in 'EventGeneratorListAssembler.xml' and the modules will be specified in 'EventGenerator.xml'.

EventGeneratorListAssembler.xml

```
<param_set name="CCQE">
   <param type="int" name="NGenerators"> 1                                  </param>
   <param type="alg" name="Generator-0"> genie::EventGenerator/QEL-CC </param>
</param_set>
```

EventRecordVisitors
(plus config)

EventGenerator.xml

```
<param_set name="QEL-CC">
   <param type="string" name="VldContext"> </param>
   <param type="int"    name="NModules">   12
   <param type="alg"    name="Module-0">   genie::InitialStateAppender/Default
   <param type="alg"    name="Module-1">   genie::VertexGenerator/Default
   <param type="alg"    name="Module-2">   genie::FermiMover/Default
   <param type="alg"    name="Module-3">   genie::QELKinematicsGenerator/CC-Default
   <param type="alg"    name="Module-4">   genie::QELPrimaryLeptonGenerator/Default
   <param type="alg"    name="Module-5">   genie::QELHadronicSystemGenerator/Default
   <param type="alg"    name="Module-6">   genie::PauliBlocker/Default
   <param type="alg"    name="Module-7">   genie::UnstableParticleDecayer/BeforeHadron
   <param type="alg"    name="Module-8">   genie::NucDeExcitationSim/Default
```

Event Record Visitors

- Look at examples in 'src/EVGModules/', e.g.:
  - 'FermiMover.{h,cxx}', 'HadronTransporter.{h,cxx}', 'PauliBlocker.{h,cxx}', 'VertexGenerator.{h,cxx}', etc.
- The classes in EVGModules tend to be generic.
  - To keep things DRY, try to use the generic modules whenever they are appropriate.
- Look for more specific models in physics-specific directories, e.g.,
  - 'src/QEL/QELKinematicsGenerator.{h,cxx}', 'src/Coherent/COHHadronicSystemGenerator.{h,cxx}', etc.

An Incomplete Model

- Sometimes you may not seek to deliver a complete new model (e.g., instead a new final state interactions model).
- In this case the task is simpler but it is still important to understand the event record visitor flow so you can see where to plug your model in, and what configuration must be edited to get it to run.
- Also, it is important to understand the "deliverables" other event record visitors around yours are expecting, etc.

An Incomplete Model

```
<param_set name="QEL-CC">
   <param type="string" name="VldContext"> </param>
   <param type="int"    name="NModules">   12                                          </param>
   <param type="alg"    name="Module-0">   genie::InitialStateAppender/Default        </param>
   <param type="alg"    name="Module-1">   genie::VertexGenerator/Default             </param>
   <param type="alg"    name="Module-2">   genie::FermiMover/Default                  </param>
   <param type="alg"    name="Module-3">   genie::QELKinematicsGenerator/CC-Default   </param>
   <param type="alg"    name="Module-4">   genie::QELPrimaryLeptonGenerator/Default   </param>
   <param type="alg"    name="Module-5">   genie::QELHadronicSystemGenerator/Default </param>
   <param type="alg"    name="Module-6">   genie::PauliBlocker/Default                </param>
   <param type="alg"    name="Module-7">   genie::UnstableParticleDecayer/BeforeHadronTransport </param>
   <param type="alg"    name="Module-8">   genie::NucDeExcitationSim/Default         </param>
   <param type="alg"    name="Module-9">   genie::HadronTransporter/Default          </param>
   <param type="alg"    name="Module-10">  genie::NucBindEnergyAggregator/Default    </param>
   <param type="alg"    name="Module-11">  genie::UnstableParticleDecayer/AfterHadronTransport  </param>
   <param type="alg"    name="IlstGen">    genie::QELInteractionListGenerator/CC-Default        </param>
</param_set>
```

Note the different configurations...

- For example, consider the algorithm chain for CC Quasi-Elastics (in 'EventGenerator.xml'). If we want to update the FSI model, we need to consider a drop-in for the "HadronTransporter".

- If we check 'src/EVGModules/HadronTransporter.cxx', we find a `HadronTransp-Model` configuration parameter, which is set in 'UserPhysicsOptions.xml':

```
UserPhysicsOptions.xml:  <param type="alg"  name="HadronTransp-Model">  genie::HAIntranuke/Default  </param>
```

A Complete Model

- Establish a series of event record visitors starting with the target and probe and evolving the record to the full final state.
- In principle, the sequence of actions is totally arbitrary. You may choose to begin by computing detailed information about the vertex, but if your physics model demands that detail be computed later, you may order them however you like.
- Typically: choose a vertex, compute differential cross section, compute lepton kinematics, compute hadronic initial state, model final state interactions.

A Complete Model

- In addition to a differential cross section (to establish event kinematics, e.g., x, y Q^2, W, etc.), we must also provide a cross section module that integrates our differential cross section to get the total cross section.
  - Implements the interface 'src/Base/XSecIntegratorI.{h,cxx}'
  - See 'src/CrossSections' for examples.
  - This is used by the spline creation routines.
  - Note: you may need to "cheat" to get around this requirement during model development (i.e., write a cross section spline for your new model by hand first).

A Complete Model

- A complete model also requires new configuration XML and likely modifications to 'UserPhysicsOptions.xml', etc.
- Additionally, it requires some boilerplate so ROOT will build dictionaries.
- These two points are tricky - it is possible to get your code to compile without doing them correctly and the associated run time failures can look very mysterious (and so be a bit frustrating). Carefully stepping through the code with `gdb` will show you where the problem is, but it may take a lot of work to find the mistake this way.
  - Better to just do the right steps from the beginning.
- More on this later (it is best illustrated via example).

Case Study: Berger-Sehgal

- In PRD 79, 053003, Berger and Sehgal proposed a number of modifications the Rein-Sehgal coherent pion model production model (from Nucl Phys B 223 (1983)).
- The models are similar (both use the PCAC hypothesis, which allows us to relate the neutrino cross section to the pion-nucleon cross section), but also different enough to be worth comparing (different predictions for the pion-nucleon cross section, different kinematic factors in the cross sections, different differential variables for the kinematics, etc.).

BS vs RS

Kinematic Term

$$\frac{d\sigma^{NC}}{dQ^2\ dy\ dt} = \frac{G_F^2 f_\pi^2}{4\pi^2} \frac{E}{|\mathbf{q}|} u\ v\ G_A^2 \frac{d\sigma\left(\pi^0 N \to \pi^0 N\right)}{dt}$$

BS

$$\frac{E_\nu u v}{|\mathbf{q}|} = \frac{(1-y)\,E_\nu}{\sqrt{y^2 E_\nu^2 + Q^2}} \left(1 - \frac{Q^2}{4\,(1-y)\,E_\nu^2}\right)$$

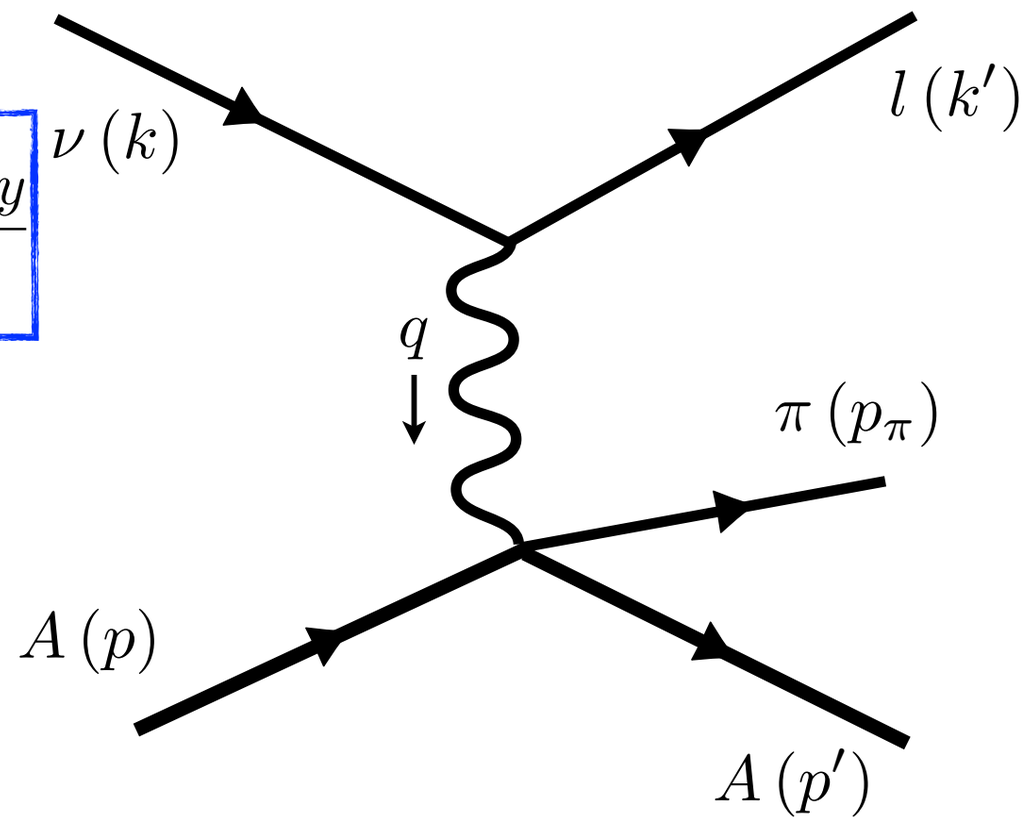
RS ($Q^2=0$)

$$\frac{E_\nu u v}{|\mathbf{q}|} \to \frac{1-y}{y}$$

$$\left[\left(G_A - \frac{1}{2}\frac{Q^2_{\min}}{Q^2+m_\pi^2}\right)^2 + \frac{y}{4}\left(Q^2 - Q^2_{\min}\right)\frac{Q^2_{\min}}{\left(Q^2+m_\pi^2\right)^2}\right]$$

Lepton Mass Correction (BS)

$$Q^2_{\min} = m_l^2 \frac{y}{1-y} \qquad G_A = \frac{m_A^2}{m_A^2 + Q^2}$$



BS "Style"

$$\frac{d\sigma_{\rm el}}{dt} = A_1 e^{-b_1 t}$$

TABLE I. Coefficients A_1, b_1 of Eq. (16).

T_π (GeV)	A_1 (mb/GeV^2)	b_1 (1/GeV^2)
0.076	11 600	116.0
0.080	14 700	109.0
0.100	18 300	89.8
0.148	21 300	91.0
0.162	22 400	89.2
0.226	16 400	80.8
0.486	5730	54.6
0.584	4610	55.2
0.662	4570	58.4
0.776	4930	60.5
0.870	5140	62.2

RS "Style"

$$\frac{d\sigma\left(\pi N \to \pi N\right)}{dt} = A^2 \left.\frac{d\sigma_{el}}{dt}\right|_{t=0} e^{-bt} F_{abs}$$

$$\left.\frac{d\sigma_{el}}{dt}\right|_{t=0} = \frac{1}{16\pi}\left(\frac{\sigma_{tot}^{\pi^+ p} + \sigma_{tot}^{\pi^- p}}{2}\right)^2$$

$$b = \frac{1}{3} R_0^2 A^{2/3}$$

$$F_{abs} = \exp\left(-\frac{9A^{1/3}}{16\pi R_0^2}\sigma_{inel}\right)$$

First Steps

- It is a good idea to step through the calculation of the integrated cross section during spline generation with a debugger (e.g., `gdb`) for the most similar model you can find.
  - This will give you an appreciation for what is required to handle the total cross section calculation (how the numerical integration is carried out, etc.).
- In order to complete the total cross section integral, this will also take you through the differential cross section, which will make it easier to understand how we compute these.

First Steps, cont.

- For example, see:
  - http://home.fnal.gov/~perdue/GENIEDevelopersManual.html#_running_the_code_the_basics
- and...
  - http://home.fnal.gov/~perdue/GENIEDevelopersManual.html#_running_the_code_the_ccqe_walkthrough
- For more on using `gdb`.

First Steps, cont.

- For Berger-Sehgal, the model is very similar to the Rein-Sehgal model, so we may begin by inspecting that model.

- Checking the structure of the Rein-Sehgal, in '$GENIE/src', we find a directory (unfortunately misspelled) that contains the coherent pion code (among other things - there is, strictly speaking, more than one "Rein-Sehgal" model).

Configuration XML

EventGeneratorListAssembler.xml

```
<param_set name="COH">
   <param type="int" name="NGenerators">    2                                  </param>
   <param type="alg" name="Generator-0 ">   genie::EventGenerator/COH-CC    </param>
   <param type="alg" name="Generator-1 ">   genie::EventGenerator/COH-NC    </param>
</param_set>
```

EventGenerator.xml

```
<param_set name="COH-CC">
   <param type="string" name="VldContext"> </param>
   <param type="int"    name="NModules">   6                                                    </param>
   <param type="alg"    name="Module-0">   genie::InitialStateAppender/Default                 </param>
   <param type="alg"    name="Module-1">   genie::VertexGenerator/Default                      </param>
   <param type="alg"    name="Module-2">   genie::COHKinematicsGenerator/Default               </param>
   <param type="alg"    name="Module-3">   genie::COHPrimaryLeptonGenerator/Default            </param>
   <param type="alg"    name="Module-4">   genie::COHHadronicSystemGenerator/Default           </param>
   <param type="alg"    name="Module-5">   genie::UnstableParticleDecayer/AfterHadronTransport </param>
   <param type="alg"    name="ILstGen">    genie::COHInteractionListGenerator/CC-Default       </param>
</param_set>
```

UserPhysicsOptions.xml

```
<!-- we have... -->
<param type="alg" name="XSecModel@genie::EventGenerator/COH-CC"> genie::ReinSeghalCOHPiPXSec/Default    </param>
<param type="alg" name="XSecModel@genie::EventGenerator/COH-NC"> genie::ReinSeghalCOHPiPXSec/Default    </param>

<!-- we would like... -->
<param type="alg" name="XSecModel@genie::EventGenerator/COH-CC"> genie::BergerSehgalCOHPiPXSec/Default  </param>
<param type="alg" name="XSecModel@genie::EventGenerator/COH-NC"> genie::BergerSehgalCOHPiPXSec/Default  </param>
```

We need to replace the cross section model for the `COH-CC` and `COH-NC` EventGenerators and adapt the modules to accept the new cross section model.

Configuration XML

- By examining the configuration XML, we realize that the Rein-Sehgal coherent pion model is the default for both `COH-NC` and `COH-CC`.
- Checking the event record visitor lists for those, we may discover the names of several important classes. Via `find` in '$GENIE/src':

```
$ ls -1 ReinSeghal/ReinSeghalCOHPiPXSec.*

ReinSeghal/ReinSeghalCOHPiPXSec.cxx

ReinSeghal/ReinSeghalCOHPiPXSec.h

$ ls -1 Coherent/*.cxx | grep -v El

Coherent/COHHadronicSystemGenerator.cxx

Coherent/COHInteractionListGenerator.cxx

Coherent/COHKinematicsGenerator.cxx

Coherent/COHPrimaryLeptonGenerator.cxx
```

Main Model Directory

Support directory for Coherent Pion models. There are several in GENIE, and some shared code is held in these classes.

Cross Section

'src/QEL/QELKinematicsGenerator.cxx'

```
void QELKinematicsGenerator::ProcessEventRecord(GHepRecord * evrec) const
{
  if(fGenerateUniformly) {
    LOG("QELKinematics", pNOTICE)
          << "Generating kinematics uniformly over the allowed phase space";
  }

  //-- Get the random number generators
  RandomGen * rnd = RandomGen::Instance();

  //-- Access cross section algorithm for running thread
  RunningThreadInfo * rtinfo = RunningThreadInfo::Instance();
  const EventGeneratorI * evg = rtinfo->RunningThread();
  fXSecModel = evg->CrossSectionAlg();
...
```

Look up cross section model dynamically...

Cross Section

'src/Coherent/COHKinematicsGenerator.cxx'

```
void COHKinematicsGenerator::ProcessEventRecord(GHepRecord * evrec) const
{
  if(fGenerateUniformly) {
    LOG("COHKinematics", pNOTICE)
      << "Generating kinematics uniformly over the allowed phase space";
  }

  //-- Access cross section algorithm for running thread
  RunningThreadInfo * rtinfo = RunningThreadInfo::Instance();
  const EventGeneratorI * evg = rtinfo->RunningThread();
  fXSecModel = evg->CrossSectionAlg();
  if (fXSecModel->Id().Name() == "genie::ReinSeghalCOHPiPXSec") {
    CalculateKin_ReinSeghal(evrec);
  } else if (fXSecModel->Id().Name() == "genie::BergerSehgalCOHPiPXSec") {
    CalculateKin_BergerSehgal(evrec);
  } else if ((fXSecModel->Id().Name() == "genie::AlvarezRusoCOHXSec")) {
    CalculateKin_AlvarezRuso(evrec);
  }
}
```

Look up cross section model dynamically...

Berger-Sehgal

- We simply mimic the directory structure and create a new package directory in '$GENIE/src'. We then copy the `XSec` classes from the Rein-Sehgal model to the new directory, gut the content, and rename the classes:

```
ls $GENIE/src/BergerSehgal

BergerSehgalCOHPiPXSec.cxx

BergerSehgalCOHPiPXSec.h
```

Berger-Sehgal

- Examining the event generator modules like 'Coherent/COHKinematicsGenerator.cxx', we find lines like:

```
if (fXSecModel->Id().Name() == "genie::ReinSeghalCOHPiPXSec") {
  CalculateHadronicSystem_ReinSeghal(evrec);
} else if ((fXSecModel->Id().Name() == "genie::AlvarezRusoCOHXSec")) {
  CalculateHadronicSystem_AlvarezRuso(evrec);
}
```

- We are lucky to find a pattern to simply extend in this case. For all the relevant event generator modules, we extend the `if-else` logic, and add the appropriate functions to the classes:

```
} else if ((fXSecModel->Id().Name() == "genie::BergerSehgalCOHPiXSec")) {
  CalculateHadronicSystem_BergerSehgal(evrec);
```

- In this case, we need 'COHKinematicsGenerator', 'COHHadronicSystemGenerator', 'COHPrimaryLeptonGenerator'.

Berger-Sehgal

- Recall the last line of the parameter set for the coherent pion models in 'EventGenerator.xml':
  - It defined (for `NC` and `CC`):
    - genie::COHInteractionListGenerator/CC-Default
- The `COHInteractionListGenerator` class lives in 'src/Coherent'.
  - It did not need to be changed or updated for the Berger-Sehgal model to work, but in some cases, you will need to implement this class correctly.

Configuration XML

- We must update 'config/master_config.xml' by adding to the "CONFIGURATION FOR XSEC ALGORITHMS" block.
- Note that there are multiple blocks and your model may need entries in the other sections as well.
  - In this case there were many things we didn't need to touch (e.g., the `InteractionListGenerator`), but you may need to watch for in your model.

'config/master_config.xml'

```
<genie_config>

   <!-- ****** CONFIGURATION FOR EVENT GENERATION MODULES ****** -->
   <config alg="genie::EventGenerator">                EventGenerator.xml              </config>
   <config alg="genie::FermiMover">                    FermiMover.xml                  </config>
   <config alg="genie::HadronTransporter">             HadronTransporter.xml           </config>
   <config alg="genie::HAIntranuke">                   HAIntranuke.xml                 </config>
   <config alg="genie::HNIntranuke">                   HNIntranuke.xml                 </config>
...

   <!--  ****** CONFIGURATION FOR XSEC ALGORITHMS ****** -->
   <config alg="genie::AhrensNCELPXSec">               AhrensNCELPXSec.xml             </config>
   <config alg="genie::AlvarezRusoCOHPiPXSec">         AlvarezRusoCOHPiPXSec.xml       </config>
   <config alg="genie::BergerSehgalCOHPiPXSec">        BergerSehgalCOHPiPXSec.xml      </config>
...
```

Configuration XML

- We obviously also need to make a new XML file so the reference in 'config/master_config.xml' is correct.

'config/BergerSehgalCOHPiPXSec.xml'

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!--
Configurations for the Berger-Seghal coherent pi production cross section algorithm

Algorithm Configurable Parameters:
.....................................................................................
Name              Type     Optional   Comment                 Default
.....................................................................................
Ma                double   Yes        Coherent Axial Mass                GPL: COH-Ma
Ro                double   Yes        Nuclear Size Scale                 GPL: COH-Ro
UseModifiedPCAC   bool     Yes        Inc. f/s lepton mass to Adler's PCAC  GPL: COH-UseModifiedPCAC
XSec-Integrator   alg      No
-->

<alg_conf>

  <param_set name="Default">
      <param type="alg" name="XSec-Integrator"> genie::COHXSec/Default </param>
  </param_set>

</alg_conf>
```

Configuration XML

- Next we must update 'config/UserPhysicsOptions.xml' as referenced above.

UserPhysicsOptions.xml

```
<!-- we have... -->
<!--
<param type="alg" name="XSecModel@genie::EventGenerator/COH-CC"> genie::ReinSeghalCOHPiPXSec/Default    </param>
<param type="alg" name="XSecModel@genie::EventGenerator/COH-NC"> genie::ReinSeghalCOHPiPXSec/Default    </param>
-->

<!-- we add -->
<param type="alg" name="XSecModel@genie::EventGenerator/COH-CC"> genie::BergerSehgalCOHPiPXSec/Default  </param>
<param type="alg" name="XSecModel@genie::EventGenerator/COH-NC"> genie::BergerSehgalCOHPiPXSec/Default  </param>
```

- Also, we updated the cross section integrator in the 'BergerSehgalCOHPiPXSec.xml' file - we need to be sure the file exists and is sensible.

- The name of integrator is decoded in 'master_config.xml' & options are set in 'COHXSec.xml'

'config/master_config.xml'

```
<genie_config>
...
   <!--  ****** CONFIGURATION GENERIC CROSS SECTION INTEGRATORS ****** -->
   ...
   <config alg="genie::COHXSec">                    COHXSec.xml                    </config>
...
```

Configuration XML

- The Simpson2D integrator is defined in 'src/Numerical/Simpson2D.{h,cxx}'
- Here the additional options come into play only when the GNU Scientific Library is enabled.

'config/COHXSec.xml'

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<alg_conf>

<!--
Configuration for the coherent pi production cross section algorithm

Configurable Parameters:
.....................................................................................
Name             Type     Optional   Comment                 Default
.....................................................................................
-->

  <param_set name="Default">
    <param type="alg" name="Integrator"> genie::Simpson2D/Default-Logarithmic-FixedNBins-101 </param>

    <param type="string" name ="gsl-integration-type">    adaptive  </param>
    <param type="int"    name ="gsl-max-eval" >             500000  </param>
    <param type="int"    name ="gsl-min-eval" >              10000  </param>
    <param type="double" name ="gsl-relative-tolerance"  >    0.01  </param>

  </param_set>

</alg_conf>
```

ROOT Class Loading

- We need to appropriately update the 'src/BergerSehgal/LinkDef.h' file so ROOT knows how to dynamically load the class. You need to add a line for all the classes:

'src/BergerSehgal/LinkDef.h'

```
#ifdef __CINT__

#pragma link off all globals;
#pragma link off all classes;
#pragma link off all functions;

#pragma link C++ namespace genie;

#pragma link C++ class genie::BergerSehgalCOHPiPXSec;

#endif
```

'src/LlewellynSmith/LinkDef.h'

```
#ifdef __CINT__

#pragma link off all globals;
#pragma link off all classes;
#pragma link off all functions;

#pragma link C++ namespace genie;

#pragma link C++ class genie::LwlynSmithQELCCPXSec;
#pragma link C++ class genie::LwlynSmithFFCC;
#pragma link C++ class genie::LwlynSmithFFNC;
#pragma link C++ class genie::LwlynSmithFF;

#endif
```

Makefiles

- Copy and paste a "local" Makefile, and update appropriately:

'src/BergerSehgal/Makefile'

```
#
# Makefile for GENIE Neutrino Generator
#
# Author: Costas Andreopoulos <costas.andreopoulos \at stfc.ac.uk>
#

SHELL    = /bin/sh
NAME     = all
MAKEFILE = Makefile

# Include machine specific flags and locations (inc. files & libs)
#
include $(GENIE)/src/make/Make.include

PACKAGE           = BergerSehgal
DICTIONARY        = _ROOT_DICT_BergerSehgal
LIBNAME           = libGBergerSehgal
EXTRA_EXT_LIBS    =

all     : rootcint lib lib-link
install : install-inc install-lib

# Include standard package makefile targets
#
include $(GENIE)/src/make/Make.std-package-targets

FORCE:
```

'src/LlewellynSmith/Makefile'

```
#
# Makefile for GENIE Neutrino Generator
#
# Author: Costas Andreopoulos <costas.andreopoulos \at stfc.ac.uk>
#

SHELL    = /bin/sh
NAME     = all
MAKEFILE = Makefile

# Include machine specific flags and locations (inc. files & libs)
#
include $(GENIE)/src/make/Make.include

PACKAGE           = LlewellynSmith
DICTIONARY        = _ROOT_DICT_LlewellynSmith
LIBNAME           = libGLlewellynSmith
EXTRA_EXT_LIBS    =

all     : rootcint lib lib-link
install : install-inc install-lib

# Include standard package makefile targets
#
include $(GENIE)/src/make/Make.std-package-targets

FORCE:
```

Makefiles

- Finally, patch the top-level Makefile with any new packages you introduced.
- It will be necessary to modify multiple lines, so choose an existing package close to yours alphabetically and search the entire Makefile to be sure you don't miss anything!

'$GENIE/Makefile'

```
core-medium-energy-range: FORCE
  @echo " "
  @echo "** Building core medium energy range physics models..."
  cd ${GENIE}/src;\
  cd AlvarezRuso;         make; cd ..; \
  cd BergerSehgal;        make; cd ..; \
  cd BodekYang;           make; cd ..; \
...
```

Interactive Sessions

- Finally, you want to be sure you may load any new libraries you have created during interactive sessions.

- After a successful build, study the '$GENIE/lib' directory to make sure your new library is present:

```
$ ls -1 $GENIE/lib/*BergerSehgal*

libGBergerSehgal-2.8.0.so

libGBergerSehgal.so
```

- Add a line to load the 'libGBergerSehgal.so' equivalent to '$GENIE/src/scripts/gcint/loadlibs.C':

```
...
gSystem->Load("libGGiBUU.so");
gSystem->Load("libGBergerSehgal.so");
gSystem->Load("libGReinSeghal.so");
...
```

The NewModel Package

`git clone https://github.com/GENIEMC/NewModel.git`

- This pulls down a set of configuration xml and class fragments that can be used as a "starter kit" for a new, complete, physics model.

- It is tricky to round out all the corner cases for model development, so we didn't include auto-patching scripts for moving the fragments into their destination xml, etc.

- Because it is tricky to cover all possible cases, please feel free to fork the repository and submit pull requests if you notice important missing features, etc.!

Mea Culpa

- Doubtlessly this presentation will have forgotten some key details!
- The best way to understand what needs to be done is to step through similar models using a debugger and just start implementing your own.
- Please contact the GENIE collaboration when you run into trouble – we want to see your work succeed!

Thanks!

Back-up