Non-Physics Aspects of the Integration of INCL++ and Geant4 into GENIE FSI

Robert Hatcher[1]

[1]`rhatcher@fnal.gov` – *Fermi National Accelerator Laboratory*

November 12, 2019

1 Introduction

The purpose of this note is to document the integration of the external codes INCL++ and Geant4 as alternative schemes for FSI interactions.

The INCL++ code is a C++ version of the Liège Intranuclear Cascade model [1]

> ... [A] nuclear-physics model that is used to simulate nucleon-, pion- and light-ion-induced reactions on nuclei, for incident energies ranging from a few tens of MeV to a few GeV.

Geant4[2] is a widely used code for particle physics. This effort for GENIE was centered around using the Bertini Intranuclear Cascade model [3]

> At the core of the Geant4 Bertini model is the concept of the intranuclear cascade of hadrons and nucleons produced through a series of interactions within the nucleus. It is essentially a classical model solving on average the Boltzmann equation for the transport of a particle through a gas of nucleons. The nuclear medium may be considered as a gas if the effective nucleon size is small and there are few collisions.

The interface code for both of these lives in the `Physics/HadronTransport` subpackage, as the `HINCLCascadeIntranuke` and `HG4BertCascIntranuke` classes respectively. New configuration XML files for each were added to the `config` directory, and an entry for each made in `config/master_config.xml`. Suitable entries were also made to the various `Messenger` configurations for the new message streams.

Both classes provide a `ProcessEventRecord(GHepRecord* evrec)` interface, that then decides whether the mode is FSI (`kGMdLeptonNucleus`) or hadron scattering (`kGMdHadronNucleus`), which in turn setup the configuration in the external model; transform GENIE particles to the form required by the model; run the model; and transform the resultant particles back into GENIE form for entry into the the `GHepRecord`.

[1]`http://irfu.cea.fr/dphn/Spallation/incl.html`
[2]`https://geant4.web.cern.ch`
[3]`https://www.sciencedirect.com/science/article/pii/S0168900215011134`

2 Externals Version

There is no currently released public version of INCL++ that can be built against GENIE. We are in negotiations with the INCL group for them to accept some suggestions for their code to add the interface for injecting the particle(s) within the nucleus, as well as additions to their build system and a new `inclxx-config` script to return important information about the build.

For Geant4, any modern version (i.e. `4.10.4.p02`, circa May 2018, or beyond) should do.

3 Configure

Configuring GENIE for either of these additions follows the usual pattern of running the `configure` script with an extra `--enable` option, and `--with` options to help guide GENIE to the locations of headers and libraries.

For INCL++:

```
--enable-incl \
--with-incl-inc=${INCL_FQ_DIR}/include/inclxx \
--with-incl-lib=${INCL_FQ_DIR}/lib \
--with-boost-inc=${BOOST_FQ_DIR}/include \
--with-boost-lib=${BOOST_FQ_DIR}/lib
```

At this time the use of INCL++ also requires the use of BOOST. This is a requirement that we hope to eliminate in the future.

For GEANT4:

```
--enable-geant4 \
--with-geant4-inc=${GEANT4_FQ_DIR}/include \
--with-geant4-lib=${GEANT4_FQ_DIR}/lib64
```

If the `--with` configurations are not given, `configure` will *try* to auto-detect the right locations based on the usual heuristic, but it is better to explicitly set them.

4 Additions to Make.config and GBuild.h

The `configure` script will record information into `src/make/Make.config` for conditional direction in the Makefiles.

```
GOPT_ENABLE_INCL={YES|NO}
GOPT_ENABLE_GEANT4_INTERFACE={YES|NO}
GOPT_WITH_INCL_INC=
GOPT_WITH_INCL_LIB=
GOPT_WITH_BOOST_INC=
GOPT_WITH_BOOST_LIB=
GOPT_WITH_GEANT4_INC=
GOPT_WITH_GEANT4_LIB=
```

The GOPT_ENABLE_ flags will have either the value YES or NO, while the GOPT_WITH_ flags will have the set paths or be blank.

At build time (based on the autogenerated-headers target) the Framework/Conventions/GBuild.h will have the additional lines (generated by the updated scripts/setup/genie-write-gbuild):

```
//#define __GENIE_INCL_ENABLED__
//#define __GENIE_GEANT4_INTERFACE_ENABLED__
```

If the option is enabled, the line will it not be commented out. These flags are used to hide the relevant classes from the compiler (and their corresponding lines in the LinkDef.h to prevent ROOT dictionary entries) when the external packages they depend on are not enabled.

5 HadronTransport Makefile

The src/make/Make.include will use the values set in Make.config to set values for:

```
INCL_FLAGS     =
INCL_INCLUDES  =
INCL_LIBRARIES =
GEANT4_FLAGS     =
GEANT4_INCLUDES  =
GEANT4_LIBRARIES =
```

These are **not** automatically added to the CXXFLAGS, CPP_INCLUDES or LIBRARIES makefile variables that are used for each GENIE subpackage.

Instead this one makefile, src/Physics/HadronTransport/Makefile, contains the lines:

```
ifeq ($(strip $(GOPT_ENABLE_INCL)),YES)
  # extra flags, include paths, and libraries to link to
  CXXFLAGS              += $(INCL_FLAGS)
  CPP_INCLUDES          += $(INCL_INCLUDES)
  ROOT_DICT_GEN_INCLUDES += $(INCL_INCLUDES) ${INCL_FLAGS}
  EXTRA_EXT_LIBS        += $(INCL_LIBRARIES)
else
  $(info $(PACKAGE) not built against INCL++)
endif

ifeq ($(strip $(GOPT_ENABLE_GEANT4_INTERFACE)),YES)
  # extra flags, include paths, and libraries to link to
  CXXFLAGS              += $(GEANT4_FLAGS)
  CPP_INCLUDES          += $(GEANT4_INCLUDES)
  ROOT_DICT_GEN_INCLUDES += $(GEANT4_INCLUDES) ${GEANT4_FLAGS}
  EXTRA_EXT_LIBS        += $(GEANT4_LIBRARIES)
else
  $(info $(PACKAGE) not built against Geant4)
endif
```

This keeps the unnecessary noise (such as flags and include paths specific to these externals) down when building other subpackages and forces this, and only this, library (`libGPhHadTransp`) to be pre-linked against the necessary external libraries.

This should off-load the burden of deciding what externals to link to away from the point where an application is built, or this library is incorporated into some external framework (e.g. LaRSoft).

6 Followup

6.1 Subdirectories in HadronTransport

Marco wondered about whether it was worthwhile to push these new classes down into subdirectories of `HadronTransport`. At this time I don't see a particularly easy / transparent way of doing that. I don't think additions to this subpackage should be a grave concern as there are currently only 20 classes (and 24 header files) in the subpackage, which is still quite manageable.

6.2 INCL++

As previously noted, there are necessary changes to INCL++ code that is publicly distributed. The integration documented in this note depends on those being accepted by the INCL group with little changes. If in negotiations the interfaces or the suggested build improvements *do* change, then adjustment will have to be made to GENIE to accomodate this.

One change that would benefit GENIE is a change to INCL++'s `G4INCLConfig`, which currently hardcodes (during the `cmake` stage) the data paths for de-excitation model data into the class, with no public interface for changing them. The only current means is via the `ConfigParser` which has a `friend` relationship with `G4INCLConfig` and so can update it. But that class is really targetted at command line parsing and pulls in a BOOST dependency, so eliminating it for something simpler would be an improvement.

6.3 Geant4

One future possible enhancement would be to provide the integration of Geant4 parameter variation interface[4]. This would need to be done with coordination from the Geant4 collaboration and must be carefully considered if run in a framework where GENIE and Geant4 are run in a single job (e.g. a multistage LaRSoft job).

6.4 Combined

Currently, GENIE can only reliably be built against one or the other. Geant4 provides a copy of the INCL++ code as part of its distribution, but it will not have the necessary changes to the INCL++ code for the interface to work. The GENIE configuration and makefiles make the assumption that if INCL++ is enabled it is coming from an external stand-alone version. At this time if both were enabled, it's likely that this might induce a version clash as two instances of the same classes would be defined.

[4]Some of the model parameters for Bertini are documented in "GEANT4 Parameter Tuning Using Professor" `https://arxiv.org/abs/1910.06417`

If at such time the version distributed by GEANT4 *does* provide the interface, then some of the makefile code will need to be reworked in order to allow this as an option. Though this might come at the loss of choices for de-excitation models (stand-alone INCL++ provides `ABLA07`, `ABLAv3p`, `GEMINIXX` and `SMM`, while it's not clear that GEANT4 supports all of those beyond `ABLA07`).

7 Building INCL++

To build a compatible version of INCL++ one must use a version that has changes to both the physics (Narisoa "Marc" Vololonianina's contribution) and to the build system (my changes).

Setup the environment for building (and using once built) INCL++

```
# make proper version of the following products available
#    gcc
#    cmake
#    root
#    boost

### specific for my case
export INCL_TOP=/grid/fermiapp/products/genie/local/inclxx

# for a particular platform
export INCL_PLATFORM=Linux64bit+2.6-2.12

# this is the currently modified version
export INCL_VERSION=inclxx-v5.2.9.5-6aca7e6

export INCL_DIR=${INCL_TOP}/${INCL_VERSION}
# needed to find de-excitation data files
export INCL_SRC_DIR=${INCL_DIR}/source/${INCL_VERSION}
# location of inclxx-config & libraries
export INCL_FQ_DIR=${INCL_DIR}/${INCL_PLATFORM}
```

The basic steps then are:

```
# specific location
export TARBALLDIR=/genie/app/rhatcher/genie_inclxx
# this tarball has Marc's physic changes, and Robert's cmake/build changes
export TARBALL=${TARBALLDIR}/${INCL_VERSION}.tar.gz

mkdir -p ${INCL_DIR}
cd       ${INCL_DIR}
mkdir source
cd    source

# extract tarball
tar xvzf ${TARBALL}
```

```
# proper permissions for CVMFS
 find . -type f -exec chmod +r {} \;
find . -type d -exec chmod +rx {} \;

cd ${INCL_DIR}
mkdir build-temp-${INCL_PLATFORM}
cd    build-temp-${INCL_PLATFORM}

# configure cmake
# -DUSE_FPIC is one of the necessary build changes
# also the ability to "install" the final products
cmake -DUSE_FPIC=ON -DCMAKE_INSTALL_PREFIX:PATH=${INCL_DIR}/${INCL_PLATFORM} \
      ${INCL_SRC_DIR}

# build the products
make

# install the specific libraries, executable, (and headers)
# in a platform specific area
make install

cd ${INCL_DIR}
# remove intermediates
rm -rf build-temp-${INCL_PLATFORM}
```

Every new terminal session will need to define the shell variables through `INCL_FQ_DIR`.

8 Building Geant4

For those that don't have a (compatible) version of Geant4 pre-built, here are some stripped down instructions (circa 2019-10-30):

Main Geant4 page: `https://geant4.web.cern.ch/`

8.1 Fetch the code

Navigate to the current stable release download:
`https://geant4.web.cern.ch/support/download_archive`

Click the download, or directly fetch it via curl (e.g.):

```
curl -O http://geant4-data.web.cern.ch/geant4-data/releases/geant4.10.04.p03.tar.gz
```

Setup a space in which to work and unroll the tarball

```
export WORKSPACE=/path/to/workspace
cd $WORKSPACE

tar xzf /path/to/tarball/geant4.10.04.p03.tar.gz
```

8.2 Configure CMake

Setup your version of the compiler and cmake (if not the system versions)

Determine the C++ standard used for ROOT build (probably `c++11` or `c++14` or `c++17` ...) so as to be compatible when linked to GENIE; you should be able to see it if you use:

```
 root-config --auxcflags
```

If you already have CLHEP installed (say for ROOT in some cases) then adjust the following to set `-DGEANT_USE_SYSTEM_CLHEP=ON` and set `-DCLHEP_ROOT_DIR=/path/to/where/CLHEP/{bin,lib,include}` exist.

Find where XERCES-C is installed (again where the bin, lib, and include directories area; call this `$XERCESC_FQ_DIR`)

```
cd $WORKSPACE
mkdir build
mkdir install

cd build

env CC=gcc CXX=g++ FC=gfortran \
cmake -DCMAKE_INSTALL_PREFIX=$WORKSPACE/install \
  -DGEANT4_BUILD_CXXSTD=c++17 \
  -DGEANT4_BUILD_GRANULAR_LIBS=OFF \
  -DGEANT4_USE_G3TOG4=OFF \
  -DGEANT4_USE_OPENGL_X11=ON \
  -DGEANT4_USE_SYSTEM_CLHEP=OFF \
  -DGEANT4_USE_GDML=ON \
  -DXERCESC_ROOT_DIR=${XERCESC_FQ_DIR} \
  -DGEANT4_INSTALL_DATA=ON \
  -DGEANT4_ENABLE_TESTING=OFF \
  $WORKSPACE/geant4.10.04.p03
```

8.3 Build

Start the build (and download of data files); the option `" -j 3 "` says run three parts in parallel (I use $N_{cores} - 1$)

```
cd $WORKSPACE/build
make -j 3
```

8.4 Install

```
cd $WORKSPACE/build
make install
```

```
cd $WORKSPACE/install
ls -1
###  bin     --  geant4-config  geant4.csh  geant4.sh
###  include --
###  lib64   --
###  share   --  Geant4-10.4.3/data (the necessary data files) + examples
```

In principle you should now be able to remove the build directory (if you need the disk space).

For building GENIE against this version of Geant4 you'll need to define:

```
export GEANT_FQ_DIR=${WORKSPACE}/install
```

To run you may need to set an environment variable so that Geant4 can find those data files; assuming you don't move the directories around you should be okay if you:

```
source $WORKSPACE/install/bin/geant4.sh
```

This will also put the right directories in `LD_LIBRARY_PATH` and `PATH` and make `geant4-config` available for use by `$GENIE/src/make/Make.include`.