Status of GIDI and MCGIDI API for GND

WPEC Subgroup 43, May 16 2017

Bret Beck

Lawrence Livermore National Laboratory

LLNL-PRES-730621

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Breaking news!!!

- For processing we need to define structures for storing (and API)
  - multi-group information
  - flux data: f(E,l,T)

General comments

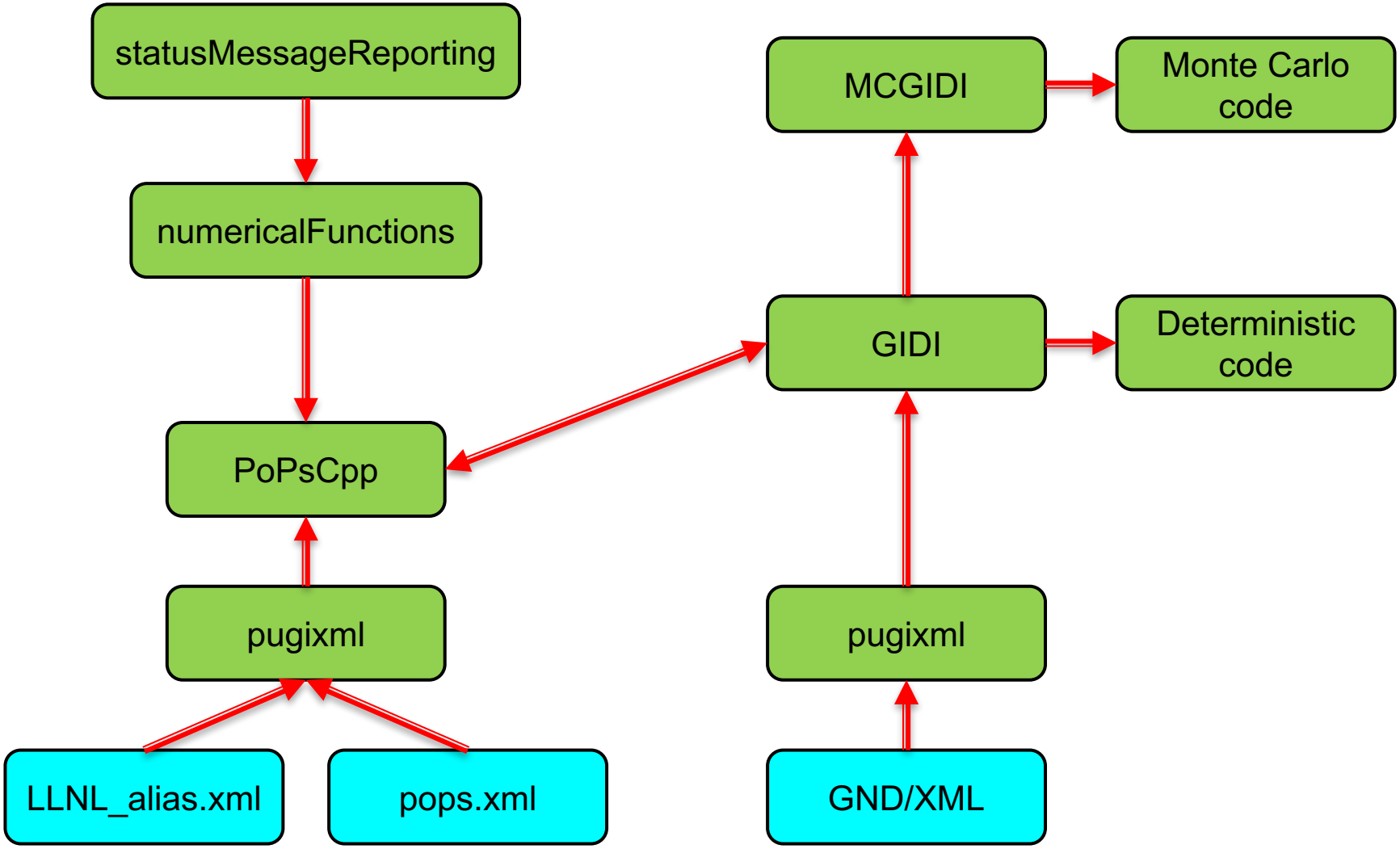
- GND is well suited for an object oriented (OO) language.
  - Ergo, I recommend that we focus on ‘modern languages’ that are OO
    - C++, Java and Python

- FUDGE can process GND data
  - Convert unit (e.g., eV to MeV)
  - Reconstruct resonances
  - Calculate average product data (like KERMA)
  - Generate (xs, pdf, cdf) for Monte Carlo transport sampling
    - E.g., $P(E',\mu|E) \rightarrow P(E'|E)\, P(\mu|E,E')$
  - Heat cross sections: only for neutron as projectile
  - Monte Carlo at multiple temperatures
  - Pn Multi-group at multiple temperatures
  - Read/write to GND/XML
    - Convert ENDF-6 to GND/Python

List of packages needed

- pugixml (C++): This is a third party package
- PoPsCpp (C++): API for particle database
- statusMessageReporting (C): Ignore this
- numericalFunctions (C): Routines to handle pointwise f(x) data.
- GIDI (C++):
  - Also has routines to read group boundaries and fluxes needed for group collapsing
  - There needs to be separate packages and formats for
    - group boundaries and
    - fluxes
- MCGIDI (C++):

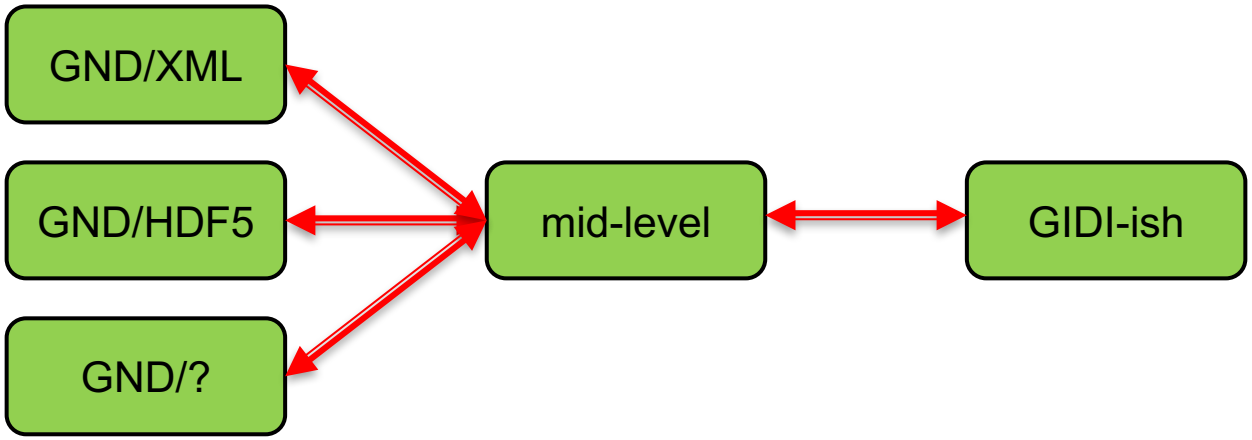
Flow diagram of data and codes



I will now describe each package.

pugixml

- Thanks to Jeremy/Austin for pointing this package out to me
- Converts XML into C++ classes in a tree structure
- http://pugixml.org
- This package is exceptionally well designed
  - For example, node.attribute("label")
    - returns a ‘null handle’ instead of throw-ing if ”label” missing.
  - Does not check XML
  - We should design a mid-level package that models this package



PoPsCpp

- C++ API developed at LLNL to read particle database SG38 is defining.
- Seems to be working properly
- Shows that each particle should only have one name (i.e., id)
  - Need to fix ’isotope’ and ‘nuclearLevel’ elements

statusMessageReporting

- C package to returns status and message when an error occurs

- Originally designed for use with numericalFunctions interface to python
  - When an error occurs in a C code called from python, it
    - should not exit
    - needs to return an instructive message

numericalFunctions

- C code to store, add, subtract, multiply, divide, etc. a pointwise representation of f(x) (i.e., a 1-d function)
  - Binary math operators for rhs as a double or another 1-d function
- Originally written for FUDGE which is mainly a python code
  - to speed up calculations that are slow in python
- Also useful for GIDI and MCGIDI

GIDI

- Currently only reads GND/XML
- Used by Ardra (LLNL determinist code) to get GND data
  - Multi-group cross sections, Q-values, deposition energy
  - Multi-temperature
  - Transfer matrices
  - Collapsing
- Used by Mercury (LLNL Monte Carlo code) to
  - read data
  - determine data it wants and
  - input to MCGIDI
- What's missing per LLNL legacy
  - Multi-group upscattering – neutron elastic scattering corrected for target thermal motion

MCGIDI

- Used by LLNL Monte Carlo code Mercury to
  - sample reaction for an isotope
  - sample products for a reaction
- Uses GIDI as input
- Multi-temperature cross sections
- Pointwise cross sections and pdf/cdf distributions
- Supports MPI broadcasting
- What's missing per LLNL legacy
  - Multi-group support
  - Upscattering (outgoing particle corrected for target thermal motion)
  - Angle biasing
  - Energy deposition
  - Photo-atomic support

Some example methods

- GIDI uses ‘GIDI’ name space
- MCGIDI uses ‘MCGIDI’ name space
- The main projectile/target/evaluation class for both is
  - GIDI::Protare
  - MCGIDI::Protare
  - ProTarE = PROjectile + TARget + Evaluation
    - n-008_O_016.endf for ENDF/B.VII.1 (MAT = 825)
    - p-008_O_016.endf for ENDF/B.VII.1 (MAT = 825)
    - n-008_O_016.endf for ENDF/B.VIII.0 (MAT = 825)
  - This is what is currently called ’reactionSuite’ in GND

Comment about c++

- class CLASS with members
  - type1 m_a
  - type2 m_b
  - type3 m_c

- Constructor is

```
CLASS::CLASS( type1 a, type2 b, type3 c ) :
                    a( m_a ),
                    b( m_b ) {

                    c = m_c;
            …
}
```

- CLASS::CLASS(type1 a, type2 b, type3 c) :

```
a( m_a ), b( m_b ), c( m_c ) {
            …
}
```

Example of GIDI

```
Reaction::Reaction( pugi::xml_node const &node,
                    PoPs::database const &pops,
                    PoPs::database const &internalPoPs ) :
        m_label( node.attribute( "label" ).value( ) ),
        m_ENDF_MT( node.attribute( "ENDF_MT" ).as_int( ) ),
        m_ENDL_C( 0 ),
        m_ENDL_S( 0 ),
        m_neutronIndex( pops["n"] ),
        m_crossSection( node.child( "crossSection" ), pops, internalPoPs, … ),
        m_availableEnergy( node.child( "availableEnergy" ), pops, internalPoPs, … ),
        m_availableMomentum( node.child( "availableMomentum" ), pops, … ) {

    ENDL_CFromENDF_MT( m_ENDF_MT, &m_ENDL_C, &m_ENDL_S );

    m_outputChannel = new OutputChannel( node.child( "outputChannel" ), … );
}
```

Example of MCGIDI

```
Reaction::Reaction( GIDI::Reaction const &reaction,
                    SetupInfo &setupInfo,
                    settings::MC const &settings ) :

    m_label( reaction.label( ) ),
    m_ENDF_MT( reaction.ENDF_MT( ) ),
    m_ENDL_C( reaction.ENDL_C( ) ),
    m_ENDL_S( reaction.ENDL_S( ) ),
    m_neutronIndex( reaction.neutronIndex( ) ),
    m_hasFission( reaction.hasFission( ) ),
    m_projectileMass( setupInfo.m_projectileMass ),
    m_targetMass( setupInfo.m_targetMass ),
    m_outputChannel( reaction.getOutputChannel( ), setupInfo, settings ) {

}
```

What else is missing

- GND
  - Alias particle needs minor work
  - Isotope / isotopeWithExcitedNucleus
  - ?
- GIDI
  - Multi-band
  - ?
- MCGIDI
  - S(a,b)
  - Unresolved resonance region probability tables
  - Multi-band

Releasing packages

- FUDGE
  - We plan to release another version of FUDGE by June 1
    - Includes: statusMessageReporting, numericalFunctions as before
    - Will also have multi-group processing for deterministic transport codes.
- GIDI and PoPsCpp
  - We will release GIDI and PoPsCpp
    - Need to go thru LLNL ‘Review and Release’
    - Not sure how long that will take
- MCGIDI
  - Will release if we get approval