GIDI+ API support for GNDS: current status and future plans

Presented to:

Consultancy Meeting on model code output & application nuclear data form structure
IAEA, Vienna, Austria

Bret Beck

March 2021

LLNL-PRES-??????

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

Lawrence Livermore National Laboratory

Outline

- Overview of LLNL GNDS support
- map, multi-group and flux files
- GIDI+
  - PoPI
  - GIDI
  - MCGIDI
- Future work and plans
- Code releases

Outline

- Overview of LLNL GNDS support
- map, multi-group and flux files
- GIDI+
  - PoPI
  - GIDI
  - MCGIDI
- Future work and plans
- Code releases

Overview

- FUDGE
  - used to convert ENDF-6 and LLNL ENDL files into GNDS, and GNDS to ENDF-6.
  - is used to modify and plot GNDS data.
  - FUDGE is used to process GNDS for Monte Carlo and multi-group transport (deterministic).
- We are developing EMU that uses FUDGE to sample realizations.
- GIDI+
  - GIDI is used by our transport codes to read GNDS data.
  - GIDI is used by our deterministic transport codes to access multi-group data.
  - MCGIDI is used by our Monte Carlo transport codes to lookup and sample data.

Overview II

- FUDGE, EMU and GIDI/MCGIDI work on an individual PROjectile + TARget + Evaluation file.
  - Called a protare in GIDI and MCGIDI
- A list of protares are referenced in a "map" file to create a library.
  - Map file specifications will probably be in GNDS 2.0.
- We developed multi-group and flux formats based on GNDS containers that are used for processing (i.e., grouping) and collapsing.
- GIDI+ (and FUDGE) work with LLNL's GNDS 1.10 and are almost compatible with 2.0. Note, the official version of GNDS is 1.9.
- In addition to GNDS reactionSuite (i.e., protare) and PoPs files, FUDGE and GIDI support map, multi-group and flux files.

Map, protare, PoPs, group and flux files complete our nuclear data needs.

Overview III

- GNDS/FUDGE/GIDI+ are particle agnostic
  - Any particle can be projectile
  - Any particle can be outgoing particle
  - Just need to be defined in PoPs (mainly mass)
- GNDS/FUDGE/GIDI+ are reaction agnostic
  - This is, do not have a fininte list of MTs
  - For example, supports any number of (n,n') reactions, not limited to 40 like ENDF
  - Another example is the 3 reaction for TNSL

Outline

- Overview of LLNL GNDS support
- map, multi-group and flux files
- GIDI+
  - PoPI
  - GIDI
  - MCGIDI
- Future work and plans
- Code releases

Map file -- test.map

```
<map library=”test22" format="0.2">

<protare projectile="n"  target=”O16"  evaluation=”fromJoe”
                         path=”fromJoe/n-008_O_016.xml"/>
<protare projectile="n"  target=”U235"  evaluation=”fromJoe”
                         path=”fromJoe/n-092_U_235.xml"/>

<protare projectile="n"  target=”U235"  evaluation=”Ian”
                         path=”fromIan/n-092_U_235.xml"/>
<protare projectile="n"  target=”U238"  evaluation=”Ian”
                         path=”fromIan/n-092_U_238.xml"/>

      <!--Implement with the GIDI ProtareTNSL class -->
<TNSL projectile="n"  target="OinBeO"  evaluation="ENDF/B-8.0”
                         path="tsl/tsl-OinBeO.xml">
                         standardTarget="O16" standardEvaluation="ENDF/B-8.0”/>

<import LLNL.map></map>
```

Multi-group boundaries and flux file

- Multi-group boundaries format uses GNDS <group> node to store the label and boundaries for a group.

```
<group label="LLNL_gid_23">
  <grid index="0" label="energy" unit="MeV" style="boundaries">
    <values>2.0908e-6 2.0908e-4 1.8817e-3 .010245 .07002 0.27097 .7527 15.754</values></grid></group>
```

- Flux stored as flux(T,E,mu) using a GNDS 3d function.

```
<XYs3d label="LLNL_fid_1">
  <axes>
    <axis index="3" label="temperature" unit="MeV/k"/>
    <axis index="2" label="energy_in" unit="MeV"/>
    <axis index="1" label="mu" unit=""/>
    <axis index="0" label="flux" unit="1/s"/></axes>
  <XYs2d outerDomainValue="0.0">
    <Legendre outerDomainValue="0.0"><values>85</values></Legendre>
    <Legendre outerDomainValue="21.0"><values>85</values></Legendre></XYs2d></XYs3d>
```

Outline

- Overview of LLNL GNDS support
- map, multi-group and flux files
- GIDI+
  - PoPI
  - GIDI
  - MCGIDI
- Future work and plans
- Code releases

GIDI+ main user packages

- PoPI
  - Properties of Particle Interface
  - C++ API to read and allow access to GNDS PoPs data
- GIDI
  - General Interaction Data Interface
  - C++ API to read and access to GNDS data
  - Developed to give GNDS protare access for transport codes
  - Follows outline of GNDS
  - Has Map and Protare classes
  - Also reads multi-group and flux files
- MCGIDI
  - Monte Carlo General Interaction Data Interface
  - C++ API for use in Monte Carlo transport codes
  - Extracts data from a GIDI::Protare
  - Stores data in more suitable way for Monte Carlo transport
  - Cross section look up by temperature and projectile energy for host code
  - Samples a reaction for a protare
  - Samples outgoing particle data for a reaction

Directory structure of packages GIDI3, MCGIDI and PoPI

```
Makefile
Doc/          # Documentation
Speeds/       # Extra
Src/          # All *.hpp and *.cpp files
Test/         # A suite of tests run with "make check"
bin/          # Some useful executables and their sources
include/      # "make" puts needed user *.hpp files here
lib/          # "make" puts needed user library files here
```

Example of executable in GIDI/bin:

readAllProtaresInMapFile # Reads all protares in a map file.

Examples of a PoPI file, a map file and various protare files are found in the Test directories. Currently, GIDI has 80 and MCGIDI has 74 tests.

GIDI+ (or gidiplus)

- To use GIDI and MCGIDI requires additional packages. GIDI, MCGIDI and these additional packages are dubbed GIDI+.
  - pugixml-1.8
    - Third party XML parser
    - Written in C++
  - HDF5 will soon be added for reading XML/HDF5 files
  - statusMessageReporting
    - Handles message passing between C packages
    - Written in C
  - numericalFunctions
    - Supports 1d numerical functions including addition, multiplication
    - Written in C
  - PoPI
  - GIDI
  - MCGIDI

statusMessageReporting and numericalFunctions are also used by FUDGE.

PoPI C++ API

- Property of Particles Interface (PoPI)
- Implements the PoPs part of GNDS
- Uses strings for particle IDs as defined in GNDS
  - (e.g., “O16”, “n”, “U235”, “u235_e6”)
- Current LLNL PoPs files
  - pops.xml (currently only defines ground state nuclei)
  - metastables_alias.xml (e.g., “Am242_m1” for “Am242_e2”)
  - LLNL_alias.xml (e.g., “92235” for “U235”)

GNDS PoPs supports aliasing. Alias are also strings (e.g., “92235” for “U235”).

GIDI C++ API

- General Interaction Data Interface (GIDI)
- A C++ API for reading a GNDS reactionSuite (i.e., protare).
  - Uses PoPI to read the PoPs part.
- Retrieving and collapsing multi-group data for use in deterministic codes (or Monte Carlo but that is better handled by MCGIDI).
- The Protare class is a virtual class. Actual classes are ProtareSingleton, ProtareComposite and ProtareTNSL.
- Support reading/writing GNDS 1.10 and 2.0(?) but, like FUDGE, uses 2.0 internally.
- Implemented in LLNL's determinstic transport code Ardra

Simple GIDI example

```
GIDI::Map map( “test.map”, pops );

GIDI::Construction::Settings construction( GIDI::Construction::e_all,
                                           GIDI::Construction::e_nuclearAndAtomic );
GIDI::Protare *protare = map.protare( construction, pops, PoPI::IDs::neutron, "O16" );

      // LLNL protares are processed with 23 temperatures.
GIDI::Styles::TemperatureInfos temperatures = protare->temperatures( );

GIDI::Settings::MG settings( protare->projectile( ).ID( ), label, true );
GIDI::Vector crossSection = protare->multiGroupCrossSection( settings, particles );
```

typedef std::vector<Styles::TemperatureInfo> TemperatureInfos;

```
(venv-3.7.2) # temperatures.py ~/GIDI_plus/GIDI/Test/Data/MG_MC3Ts/neutrons/n-008_O_016.xml
/g/g16/beck6/Git/GIDI_plus/GIDI/Test/Data/MG_MC3Ts/neutrons/n-008_O_016.xml
  temperature 0.0 K: eval
     temperature [K]      heated  griddedCrossSection URR_probabilityTables  heatedMultiGroup   SnElasticUpScatter
    ------------------------------------------------------------------------------------------------------------
               300.1  heated_000       MonteCarlo_000                        MultiGroup_000
            1.16e+04  heated_001       MonteCarlo_001                        MultiGroup_001
            1.16e+06  heated_002       MonteCarlo_002                        MultiGroup_002
```

MCGIDI C++ API for GNDS

- Monte Carlo GIDI (MCGIDI)
- A C++ API for Monte Carlo transport codes
- Like GIDI, the Protare class is a virtual class. Actual classes are ProtareSingleton, ProtareComposite and ProtareTNSL.
- Can do LLNL model A and B (MCNP) upscatter for outgoing particles
- Supports broadcasting for MPI and GPUs
- Implemented in LLNL's Monte Carlo transport code Mercury

Simple MCGIDI example

```
double energy = 14.1, crossSection, reactionCrossSection;

MCGIDI::DomainHash domainHash( 4000, 1e-8, 10 );
MCGIDI::Protare *MCProtare = MCGIDI::protareFromGIDIProtare( *protare, pops, MC
                        particles, domainHash, temperatures, reactionsToExclude );

int hashIndex = domainHash.index( energy );
crossSection = MCProtare->crossSection( URR_infos, hashIndex, temperature, energy );
reactionCrossSection = MCProtare->reactionCrossSection( ir, URR_infos, hashIndex,
                        temperature, energy );

int reactionIndex = MCProtare->sampleReaction( URR_infos, hashIndex,
                                temperature, energy, crossSection, rng, rngState );
reaction->sampleProducts( MCProtare, energy, input, products );
```

Outline

- Overview of LLNL GNDS support
- map, multi-group and flux files
- GIDI+
  - PoPI
  - GIDI
  - MCGIDI
- Future work and plans
- Code releases

Some general stuff

- Finish specification of GNDS 2.0 support
- Add units support to GIDI+
- Speed up MCGIDI
- “True” on-the-fly heating
- Multi-grouping on-the-fly
- Thermal nuclear data
  - This is Maxwell averaged nuclear data
  - Used, for example, in astrophysics
- Update GEANT4 for latest GIDI+

Expanding PoPs database

- Our current PoPs data only has nuclear ground state data.
  - Disk size is 2 MB
- Ian Thompson has created a PoPs database that includes nuclear excitation levels from RIPL
  - Disk size is 96 MB
  - Takes GIDI 8 seconds to read in and FUDGE over a minute
  - Still far from complete (e.g., missing decay data)
- We are developing a ‘Map Of Particles’ (mop) structure that outlines a PoPs database and makes reading in a few particles much faster and uses much less memory.

HAPI (Hierarchical API)

- Why HAPI
  - To supports GNDS in other “meta-languages” (XML, HDF5, XML/HDF5, etc.)
  - Need to improve load time (90% time spent converting double strings to binaries)
  - Takes about 70 minutes to read in all of ENDF/B-VIII.0 processed with 23 temperatures. Has evaluate, heated, MC and multi-group data.
- Status
  - All in XML, or values nodes in HDF5 and the rest in XML
  - Caleb Mattoon and Adam Kunen (computer support) are about to release GIDI+ with HAPI

Mode	XML	Hybrid Uncomp	Hybrid gzip
m=0 (all)	2.0x	6.6x	2.6x
m=1 (MC)	2.9x	9.0x	3.2x
m=2 (SN)	3.6x	10.7x	6.7x

More efficient GNDS files

- Read time components for XML/HDF5 hybrid
  A. 8% parsing XML
  B. 25% reading bulk arrays from HDF5
  C. The rest: GIDI constructing
- We will also make changes to how FUDGE writes GNDS files.
  - These changes will still be GNDS 2.0 compliant
  - These changes will reduce load times for A, B and C above
  - This should give us another factor of two faster loading
- We plan to allow for the reading of only the temperature data needed. This would greatly reduce C when only a single temperature is requested

LLNL transport codes read data in parallel which greatly reduces load times.

Support for all parts of GNDS

- Currently, GIDI only reads and supports the parts of the GNDS needed by transport codes
- In particular, it does not support
  - Documentation nodes
  - Covariance data in the GNDS file
  - Resonance data in the GNDS file
- We will be adding support for all parts of GNDS

Improved gamma (photon) and electron support

- GIDI and MCGIDI support photo-atomic and photo-nuclear data
  - ProtareComposite is a sub-class of Protare class
  - It allows one to treat photon as projectile as one instance.
    - Has one ProtareSingle for photo-atomic data and one for photo-nuclear data
  - In the future it may be used for element protares
    - E.g., a protare with "n + Fe" may contain "n + (Fe54, Fe56, Fe57 and Fe80)"
- We plan to add support for X-ray Fluorescence Data
  - GIDI
  - MCGIDI for sampling
- LLNL's Monte Carlo code Mercury is looking to add electrons
  - We plan to add electron data to GIDI and MCGIDI
  - Also processing in FUDGE

Outline

- Overview of LLNL GNDS support
- map, multi-group and flux files
- GIDI+
  - PoPI
  - GIDI
  - MCGIDI
- Future work and plans
- Code releases

Code releases

- We are releasing all codes under https://github.com/LLNL
  - FUDGE
    - Named “fudge”
    - Version 4.2.3
      - Python 2.7
    - Next version
      - Python 3.6+
      - Pip install
    - BSD license (probably will switch to MIT license)
  - GIDI+
    - Named “gidiplus”
    - Version 3.18.129
    - GNDS 1.10 (internal LLNL version)

- Releasing all codes under MIT license, except currently FUDGE

We need to release a GNDS 2.0 version soon.