LLNL's GNDS APIs for transport codes.

WPEC 2018 / GNDS-B
Paris, France
May 16 2018

Bret Beck

Lawrence Livermore National Laboratory

LLNL-PRES-749977

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

Codes

- FUDGE
  - Python based GNDS infrastructure toolkit
- PoPs
  - C++ Properties of Particles library
- GIDI
  - C++ reader for transport codes
- MCGIDI
  - C++ sampler for Monte Carlo transport codes

We are trying to be 100% data driven!

FUDGE processing

- Converts units
- Reconstructs cross sections and angular data for resonance parameters
- Calculated average product energy
- Creates cdf from pdf for Monte Carlo sampling
  - Stored as xs, pdf, cdf (xs are mus or outgoing energies)
- Heats cross sections
- Puts cross sections on a common grid for Monte Carlo
- Multi-groups data (with upscatter for Sn transport)

Need to add unresolved resonance probability table processing, multi-band and thermal scattering law data

GNDS files created by FUDGE

- Main LLNL library
  - ENDL2009.3
    - Each protare has 22 Sn and Monte Carlo temperature – in progress
      - Room to 100 keV
  - Projectiles are
    - Neutron, proton, deuteron, triton, helium-3, alpha and photo-atomic
    - About 600 targets for the neutron projectile
- ENDF/B-VII.1 and ENDF/B-VIII
  - Two temperatures
    - 0 k and room
  - n, p, d, t, h, a, photo-atomic and photo-nuclear
- Currently storing data in ascii/XML
  - ENDL2009.3
    - Large ~200 GB
    - Slow to read in
    - Caleb will present on XML/HDF5 hybrid

PoPs (Properties of Particles)

- We created a stand alone PoPs database (file)
  - Used to define particle ids, masses, spins, nuclear level energy, etc.
    - Transport codes get indices and masses from us (i.e., nuclear team)
    - Indices assigned in order as particles are loaded
  - GIDI and MCGIDI use unique integer ‘id’s (indices) and not string id’s internally
    - Probably not important for GIDI but may be for MCGIDI
    - For example, particle id “O15” may be index 421

- Also have two alias files
  - One for meta-stables – what is a meta-stable?
  - One for legacy ZA support (1000 * Z + A)
    - User decks still use ZA which transport codes store as a string (e.g., “8015”)
      - Alias maps the string to a particle’s id (“8015” -> ”O15”)

- Transport codes must create a PoPs instance and pass it to GIDI classes.

Map file

- A GNDS file defines one “protare”
  - PROjectile + TARget + Evaluation
- A map file links many ”protare”s into one library
  - Like an xsdir file

```
<map>
<protare evaluation="ENDF/B-7.1" projectile="n" target="O16" path="neutrons/n-008_O_016.xml"/>
<protare evaluation="ENDF/B-7.1" projectile="n" target="Fe56" path="neutrons/n-026_Fe_056.xml"/>
<protare evaluation="ENDF/B-7.1" projectile="n" target="Th227" path="neutrons/n-090_Th_227.xml"/>

<protare evaluation="ENDF/B-7.1" projectile="H1" target="Li6" path="protons/p-003_Li_006.xml"/>

<protare evaluation="ENDF/B-7.1" projectile="photon" target="O16" path="photo-nuclear/g-008_O_016.xml"/>

<protare evaluation="ENDL-99"       projectile="photon" target="O” path="photo-atomic/photoat-008_O_000.xml"/>
<protare evaluation=" ENDF/B-7.1" projectile="photon" target="O” path="photo-atomic/photoat-008_O_000.xml"/>
<protare evaluation="ENDF/B-7.1" projectile="photon" target="Th”
            path="photo-atomic/photoat-090_Th_000.xml"/>

<import path=“another/map/file.map”/>
</map>
```

GIDI

- GIDI has a Map class
  - Given a projectile, target [and evaluation] returns path to a protare file

```
std::string protareFilename(
            int projectileIndex,
            int targetIndex,
            std::string const &evaluation = "" );
```

  - Is it the users responsibility to read in all required files

```
pops = PoPs::Database( mainPopsFilename );
pops->addDatabase( LLNL_aliasFilename );
pops->addDatabase( metastable_aliasFilename);
```

    - May need to have a way to combine these into one file

GIDI

- Protare class

```
Protare(     std::string const &fileName,
             PoPs::Database const &pops );
```

  - Uses pugixml to parse XML file
    - Pugixml does not check for proper XML but it is still slow to read in

- Getting specific data for a given temperature

```
Styles::TemperatureInfos = protare->temperature( )
```

  - Returns a list of TemperatureInfo objects with (sorted by temperature)
    - Temperature
    - Ungridded data
    - Gridded data (For Monte Carlo transport)
    - Multi-group data
    - Upscatter multi-group data

Processing example

```
processProtare.py
    -t 2.586e-08          # Room temperature
    -t 1e-6               # 10^-6 MeV/k

    -mc                   # Monte Carlo transport
    -mg                   # Multi-group transport
    -up                   # Upscatter MG transport

    n-001-H_001.xml       # GNDS evaluation
```

Styles example for two temperatures

```
<evaluated                label="eval"/>

<averageProductData       label="apd"             derivedFrom="eval"/>
<MonteCarlo_cdf           label="MonteCarlo"      derivedFrom="apd"/>

<multiGroup               label="MultiGroup"/>


<heated                   label="heated_0"        derivedFrom="MonteCarlo">
       <temperature value="2.586e-08" unit="MeV/k"/></heated>
<griddedCrossSection      label="MonteCarlo_0"    derivedFrom="heated_0"/>
<heatedMultiGroup         label="MultiGroup_0"    derivedFrom="heated_0">
<SnElasticUpScatter       label="UpScat_0"        derivedFrom="MultiGroup_0"/>


<heated                   label="heated_1"        derivedFrom="MonteCarlo”>
       <temperature value="1e-06" unit="MeV/k"/></heated>
<griddedCrossSection      label="MonteCarlo_1"    derivedFrom="heated_1"/>
<heatedMultiGroup         label="MultiGroup_1"    derivedFrom="heated_1"/>
<SnElasticUpScatter       label="UpScat_1”        derivedFrom="MultiGroup_1"/>
```

Cross section for elastic scattering

```
<XYs1d       label="eval" >                 ... </XYs1d>

<XYs1d       label="heated_0" >             ... </XYs1d>
<Ys1d        label="MonteCarlo_0" >         ... </Ys1d>
<gridded1d   label="MultiGroup_0" >         ... </gridded1d>

<XYs1d       label="heated_1" >             ... </XYs1d>
<Ys1d        label="MonteCarlo_1" >         ... </Ys1d>
<gridded1d   label="MultiGroup_1" >         ... </gridded1d>
```

Distributions

- Neutron distribution for ”neutron + proton” reaction

```
<angularTwoBody    label="eval”>              ... </angularTwoBody>
<angularTwoBody    label="MonteCarlo”>        ... </angularTwoBody>
<multiGroup3d      label="MultiGroup_0”>      ... </multiGroup3d>
<multiGroup3d      label="UpScat_0”>          ... </multiGroup3d>
<multiGroup3d      label="MultiGroup_1”>      ... </multiGroup3d>
<multiGroup3d      label="UpScat_1”>          ... </multiGroup3d>
```

- Proton distribution for ”proton + photon” reaction

```
<angularTwoBody    label="eval”>              ... </angularTwoBody>
<angularTwoBody    label="MonteCarlo”>        ... </angularTwoBody>
<multiGroup3d      label="MultiGroup_0”>      ... </multiGroup3d>
<multiGroup3d      label="MultiGroup_1”>      ... </multiGroup3d>
```

Styles::TemperatureInfos returned

index	temperature	ungridded data	gridded data	multi-group data	upscatter multi-group data
0	2.586e-08	heated_0	MonteCarlo_0	MultiGroup_0	UpScat_0
1	1e-6	heated_1	MonteCarlo_1	MultiGroup_1	UpScat_1

GIDI/MCGIDI "processing"

- Currently, only processing not done by FUDGE are:
  - GIDI:
    - Energy deposition
      - Fudge calculates available energy and average outgoing per product
    - Multi-group collapsing
  - MCGIDI
    - photo-atomic
      - For coherent scattering only form factor (FF) and anomalous scattering data are stored. MCGIDI calculates running integrals of FF(x) and FF(x)^2
      - For incoherent scattering only the scattering function (SF) data are stored. MCGIDI calculates running integrals of SF(x)
- FUDGE currently does not process URR probability tables, multi-band data and neutron thermal scattering law data

Final comments

- Is the API SG 43 is developing for evaluators or transport codes or both?
  - The need is different
- Multi-group boundaries structure
- Flux structure
- GNDS map file structure
- PoPs “map” file structure
- GNDS/PoPs ”map” file structure