LA-UR-20-21302
Approved for public release; distribution is unlimited.

Title: FAUST - Benchmark and Validation Framework

Author(s): Haeck, Wim
Thompson, Nicholas William
Clark, Alexander Rich
Dominguez Grado, Juan Angel
Kistle, Hadyn Marie
Herman, Michal W.

Intended for: NCSP Technical Program Review, 2020-02-10/2020-02-12 (Santa Fe, New Mexico, United States)

Issued: 2020-02-10

FAUST

Benchmark and Validation Framework

W. Haeck, N. Thompson, A. Clark,
J. Dominguez, H. Kistle, M. Herman

February 10, 2020

NNSA
National Nuclear Security Administration
Managed by Triad National Security, LLC for the U.S. Department of Energy's NNSA

Introduction

Python packages for nuclear data applications and benchmarking

Goals and objectives for FAUST

- Provide input and output processing for different calculation codes
- Allow for exchanging results between different applications and codes
- Running benchmarks and processing the results
- Automate and simplify plot and report generation
- Provide a basis for developing applications useful for nuclear data evaluators

To be used both inside and outside LANL

- All packages are currently hosted internally at LANL
- For example: the LANL ML project and the OECD/NEA WPEC VaNDaL project

Sponsored by ASC-PEM-NP, ASC-ADTM-ML, ASC-V&V-QM, NCSP

Overview of current features

Available packages:

- `result` : storing, serializing, deserialising calculation results
- `sensitivity` : filtering, analysing and using sensitivity data
- `benchmarks` : the material balance table for comparing benchmark inputs
- `mcnp` : extracting data from the mctal and the output file
- `sensmg` : extracting keff and leakage sensitivity profiles (sens_k_x and sens_l_x file)
- `scale` : extracting sensitivity profiles from sdf files (N. Thompson, NEN-2)
- `elementary` : utility package to convert nuclide and reaction identifiers

Work in progress:

- `sensitivity.crater` : evaluate observable changes due to nuclear data changes
- `mcnp.input` : parsing, creating and modifying MCNP input files
- `partisn` : reading and writing binary data files (A. Clark, XCP-3/XCP-5)
- `lmx` : detector events in subcritical multiplication experiments (H. Kistle, NEN-2)

`mcnp` : input and output processing for MCNP

The `mcnp` package is dedicated to MCNP input and output

Extracting results from the MCNP output and mctal file

- Effective multiplication factor, point kinetic values and related information
- Sensitivity profiles (from the output file only)
- Print tables 40, 50 and 60 (material, surfaces and volume information)
- Tally results (from the mctal file only)

Extracting derived data from the print tables

- Balance tables for input comparison (OECD/NEA WPEC VaNDaL project)

Creating, reading and modifying MCNP input file

- Extract information from the input file (dimensions, materials, etc.)
- Modify and regenerate the input file at the request of the user
  - For example: uncertainty analysis in benchmark evaluation

mcnp : input and output processing for MCNP

Processing data from the output file

```
# extract results for multiple output objects in a single go
mcnp = McnpOutput( [ McnpEffectiveMultiplicationFactor(), McnpPointKinetics() ] )
mcnp.extract( 'HEU-MET-FAST-001-001.mcnp.o' )

# retrieve information from the output object
keff = mcnp.data[0]  # mcnp.data is the list of output objects
atff = keff.aboveThermalFissionFraction
aecf = keff.averageEnergyCausingFission
```

Reading and modifying the input file

```
# parse an existing input file
mcnp = McnpInput( 'HEU-MET-FAST-001-001.mcnp' )

# retrieve and modify material related data
heu = mcnp.materials()[0]
u5 = heu.composition[ '92235' ]
heu.composition[ '92235' ] = 1.01
heu.composition.update( { '92235' : 1.0 } )
```

`sensitivity` : tools for sensitivity analysis

Filtering and analysis of the maximum sensitivity

```
# create a filter
select = SensitivityFilter( lowerSensitivity = 10., nuclides = [ 'U235', 'U238' ] )

# select godiva profiles
godiva = profiles[ 'HEU-MET-FAST-001-001-s' ]
godiva = [ profile for profile in godiva if select( profile ) ]

# retrieve maximum sensitivity and sort
table = MaximumSensitivityTable( profiles )
print( table )
```

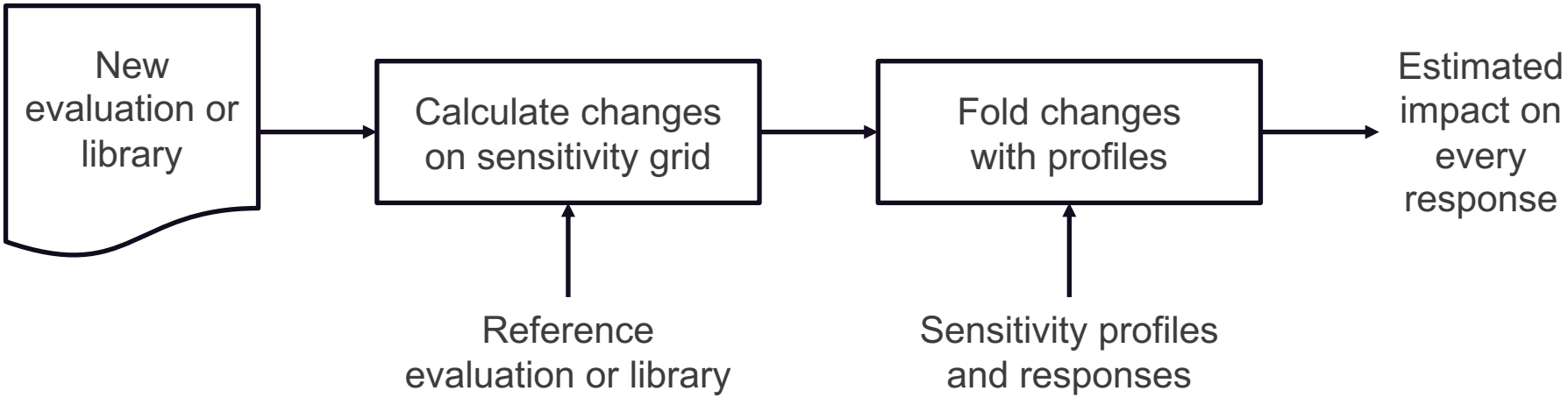
Case	Nuclide	Reaction	Max sensitivity at energy bin [MeV]	Maximum sensitivity [pcm/%]	Integrated sensitivity [pcm/%]
HEU-MET-FAST-001-001-s	U235	total nu	0.4 - 0.9	209.90	982.42
HEU-MET-FAST-001-001-s	U235	prompt nu	0.4 - 0.9	208.48	976.13
HEU-MET-FAST-001-001-s	U235	total	0.4 - 0.9	186.03	804.28
HEU-MET-FAST-001-001-s	U235	fission	0.4 - 0.9	142.86	654.11
HEU-MET-FAST-001-001-s	U235	elastic	0.1 - 0.4	37.78	108.06
HEU-MET-FAST-001-001-s	U235	inelastic	0.4 - 0.9	22.50	79.20

`crater` : estimate impact of nuclear data changes

An analysis tool to estimate the impact of nuclear data changes

- Sensitivity profiles provide impact of changing a parameter *p* on a given response R

$$R' = R\left(1 + \sum_{p}\sum_{g} C_{p,g} S^{R}_{p,g}\right)$$



`crater` : estimate impact of nuclear data changes

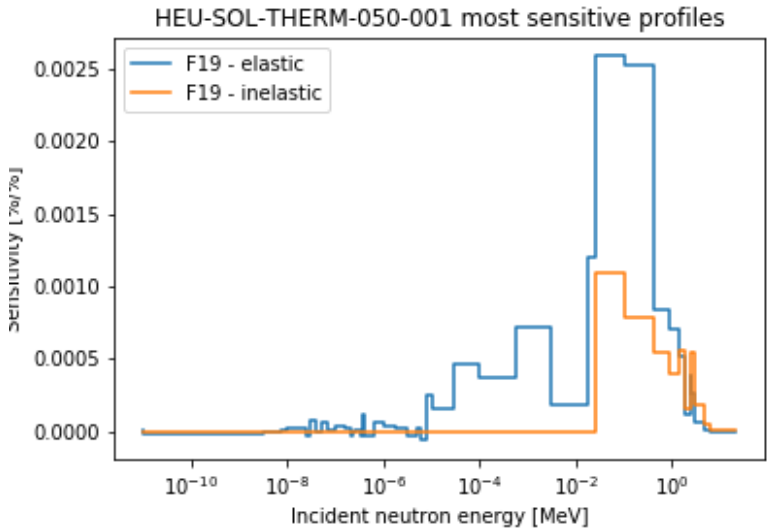
```
# load observables and profiles - both are from MCNP calculations
observables = fromJSON( '/local/json/keff.endf80.20200127.json' )
profiles = fromJSON( '/local/json/sensitivities.crossSection.endf80.20200127.json' )

# create a crater instance using the previously loaded observables and profiles
crater = Crater( observables, profiles )

# create input
changes = CraterInput()
changes.xs.structure = [ 1e-11, ..., 20.0 ] # standard 44 group structure

# change F19 inelastic scattering (can be done by ratio or relative change)
changes.xs.addRatio( 'F19', 'inelastic',
                     [ 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
                       1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
                       1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
                       1., 1., 1., 0.937, 1.964, 1.128, 1., 1., 1., 1.,
                       1., 1., 1., 1. ] )

# verify changes - check if the nuclide,reaction pairs actually have sensitivities
crater.verify( changes )

# calculate the impact of these changes - using the same format as the original MCNP observables
newObservables = crater.impact( changes )
```

`crater` : estimate impact of nuclear data changes

For this F19 example, 1029 benchmark cases were considered

- 107 of these lead to changes in the keff value
- 79 of these change by more than 10 pcm
- Mainly for `HEU-SOL-THERM-050`, `U233-SOL-INTER-001`, `U233-SOL-THERM-015`

Case	Before	After	Impact
HEU-SOL-THERM-050-001	1.00587	1.00693	0.00106
HEU-SOL-THERM-050-002	1.00120	1.00200	0.00080
HEU-SOL-THERM-050-003	1.00249	1.00375	0.00126
HEU-SOL-THERM-050-004	1.00257	1.00347	0.00090
HEU-SOL-THERM-050-005	0.99976	1.00024	0.00048
HEU-SOL-THERM-050-006	1.00755	1.00837	0.00082
HEU-SOL-THERM-050-007	0.99622	0.99740	0.00118
HEU-SOL-THERM-050-008	0.99633	0.99719	0.00086
HEU-SOL-THERM-050-009	0.99471	0.99591	0.00120
HEU-SOL-THERM-050-010	0.97854	0.97921	0.00067
HEU-SOL-THERM-050-011	0.99026	0.99077	0.00051



Where we would like to go …

`mephisto` : a benchmark run and result collection system

- Scalable to work on a single PC or a cluster (or maybe a heterogeneous system)