LA-UR-21-26578
Approved for public release; distribution is unlimited.

Title: FAUST - A Benchmark and Validation framework for Nuclear Data

Author(s): Haeck, Wim
Clark, Alexander Rich
Bravo, Caiser Alexander
Kistle, Hadyn Marie
Thompson, Nicholas William
Herman, Michal W.

Intended for: 2021 MCNP User Symposium, 2021-07-12/2021-07-16 (Los ALamos, New Mexico, United States)

Issued: 2021-07-12

FAUST
A Benchmark and Validation framework for Nuclear Data

W. Haeck, A. Clark, C. Bravo, H. Kistle,
N. Thompson, M. Herman

July 13, 2021

Outline

1. Introduction and overview
2. Crater : calculating the impact of nuclear data changes

Introduction and overview

- Objectives OF FAUST
- Overview of features and example code

Introduction

- Some of the main tasks of the XCP-5 Nuclear Data Team at LANL:
  - Maintain nuclear data libraries for LANL simulation codes (MCNP, PARTISN, etc.)
  - Verify and validate new data libraries when they become available

- 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
    - Benchmark selection for testing purposes using sensitivity and similarity
    - Sensitivity analysis and folding to assess the impact of nuclear data changes
    - Nuclear data format and physics testing

Quick overview of current features

- FAUST is developed as a set of independent python packages

- Available packages that are currently available:
  - `result` : storing, serializing, deserialising calculation and experimental results
  - `sensitivity` : filtering, analysing and applying sensitivity data
  - `mcnp` : extracting data from output files, manipulating the input file
  - `sensmg` : extracting values and sensitivity profiles
  - `scale` : extracting sensitivity profiles from sdf files
  - `lmx` : Feynman and Rossi-alpha analysis for neutron noise analysis

Examples for the `mcnp` package

- Processing data from the MCNP 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 } )
```

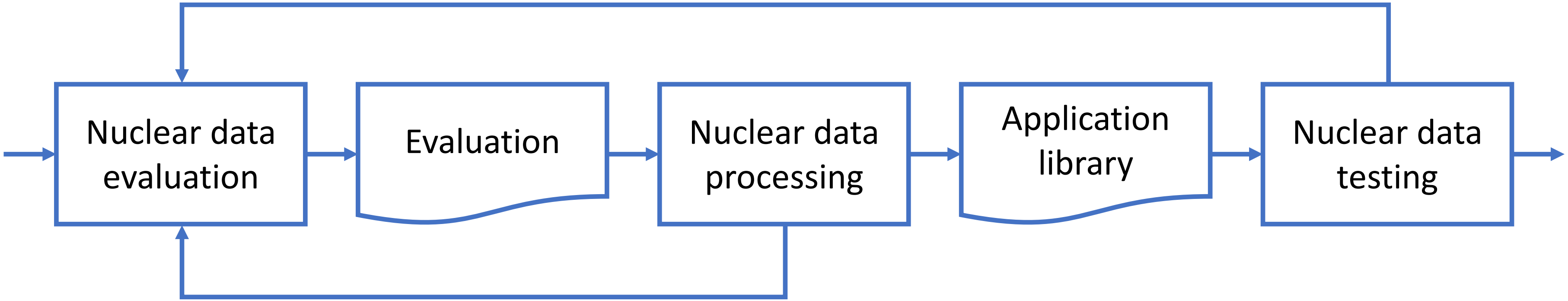
Crater : calculating the impact of nuclear data changes

- Accelerating the nuclear data pipeline
- Using sensitivity profiles
- Example and comparison with MCNP calculated results

Accelerating the nuclear data pipeline

- Accelerating the nuclear data pipeline
  - Getting good nuclear data to applications faster
  - An objective of the nuclear data community
- Nuclear data testing: just one of the feedback loops
  - This process takes time: hundreds to thousands of calculations
  - A solution: using sensitivity profiles to assess impact

$$S = \frac{p}{r}\frac{\partial r}{\partial p}$$



Using sensitivities to assess changes to a response

- Sensitivity: change in a response due to change of a parameter

$$S = \frac{p}{r}\frac{\partial r}{\partial p} \approx \frac{p}{r}\frac{\Delta r}{\Delta p}$$

Assuming the change is small enough so that linear perturbation theory applies

$$\Delta r = S\frac{\Delta p}{p}r \qquad r' = r\left(1 + S\frac{\Delta p}{p}\right) \qquad r' = r\left(1 + \sum_g S_g \frac{\Delta p_g}{p_g}\right)$$

- When applying multiple changes in independent parameters

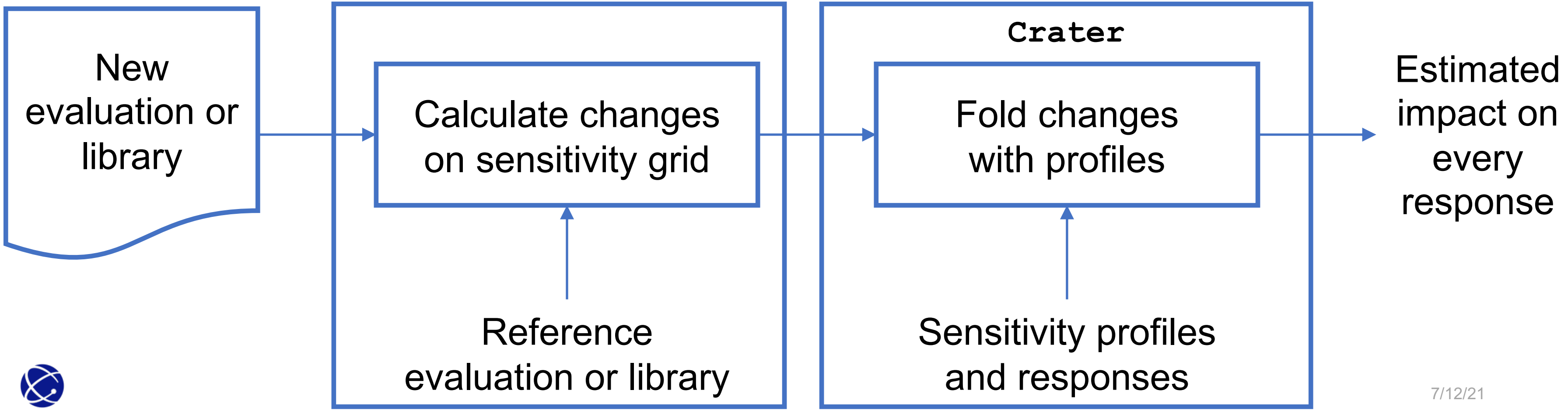
$$r' = r\left(1 + \sum_i \sum_g S_{i,g} \frac{\Delta p_{i,g}}{p_{i,g}}\right)$$

- NDAST (OECD/NEA): an existing tool that allows us to do this

`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

$$\boldsymbol{r'} = \boldsymbol{r}\left(\mathbf{1} + \sum_{\boldsymbol{p}} \sum_{\boldsymbol{g}} \boldsymbol{c}_{\boldsymbol{p},\boldsymbol{g}} \boldsymbol{S}^{\boldsymbol{r}}_{\boldsymbol{p},\boldsymbol{g}}\right)$$



Response values and sensitivity profiles


```
[ { "type" : "effectiveMultiplicationFactor",
    "data" : { "values" : [ 1.0000 ],
               "uncertainties" : [ 0.0001 ] } },
  { "type" : "sensitivityProfile",
    "response" : "effectiveMultiplicationFactor",
    "parameter" : "crossSection",
    "particleId" : "neutron",
    "nuclide" : "U235",
    "reaction" : "fission",
    "material" : "total",
    "data" : { "values" : [ -1.7129e-17, 1.4106e-09 ],
               "uncertainties" : [ 0.0034, 0.0033 ],
               "structure" : [ { "name" : "energy-in",
                                 "type" : "histogram",
                                 "limits" : [ 1e-11, 10.0, 20.0 ],
                                 "unit" : "MeV"} ],
               "units" : { "value" : "%/%", "uncertainty" : "relative" } } } ]
```


ENDF/B-VIII.0 MCNP results for ~1100 ICSBEP benchmarks

- Values for keff, beff and Leff
- Three group spectra
- Fission fractions
- Average energy causing fission
- Sensitivity profiles
- And much more …

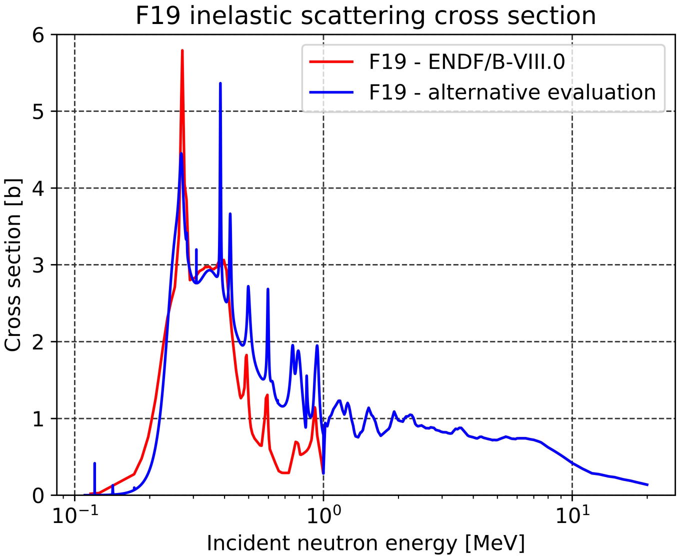
A lot of data: 1.3 GB json file for the cross section sensitivities alone

Example code

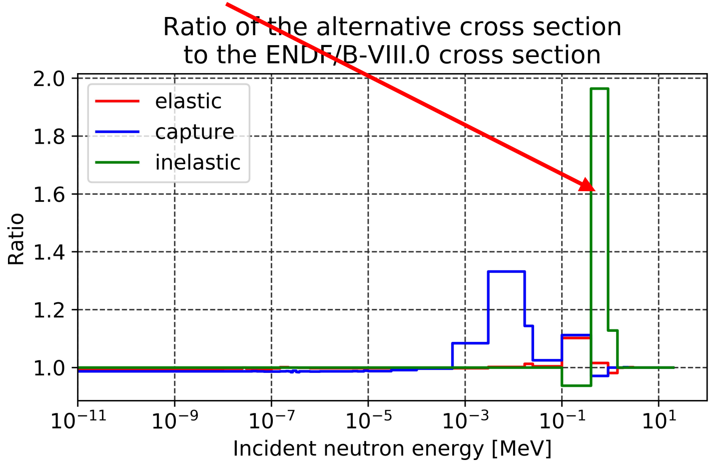
```
# load observables and cross section sensitivity profiles - both are from MCNP calculations
observables = fromJSON( '/local/json/keff.endf80.20200127.json' )
profiles = fromJSON( '/local/json/sensitivities.crossSection.endf80.20200127.json' )

# 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 cross section (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. ] )

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

An example: an alternative evaluation for F19

- ML work at LANL indicated a potential issue in ENDF/B-VIII.0 F19
  - It was found that inelastic scattering was too low between 0.5 and 1 MeV
  - An alternative evaluation was proposed for ENDF/B-VIII.0 but ultimately not adopted



One could argue that this is not a small change



An example: an alternative evaluation for F19

- For this F19 example, 1029 benchmark cases were considered
  - Only 107 of these contain F19
  - 79 of these change by more than 10 pcm
  - The largest change is smaller than 150 pcm
  - Mainly for HEU-SOL-THERM, U233-SOL-INTER, U233-SOL-THERM

This represents a change of ~0.15%

- For this example, we have verified these values against MCNP6.2

- Before we get to the results, some fun facts:
  - Crater: just a few seconds of calculation time for 1000 benchmarks
  - MCNP6.2: from hours to days to weeks
    - Depending on the number benchmarks
    - Depending on precision

OK, so this isn't entirely true since GB sized json files take some time to load

An example: an alternative evaluation for F19

Case	Original MCNP6.2	Crater estimate	MCNP6.2 estimate	Crater Δkeff	MCNP6.2 Δkeff	MCNP6.2 σ(Δkeff)	Impact
HST-009-001	1.00094	1.00154	1.00198	60	104	18	< 3
HST-009-002	1.00192	1.00237	1.00231	45	39	20	< 1
HST-009-003	1.00204	1.00240	1.00246	36	42	18	< 1
HST-009-004	0.99695	0.99710	0.99728	15	33	18	< 1
HST-050-001	1.00587	1.00693	1.00654	106	67	21	< 2
HST-050-002	1.00120	1.00200	1.00204	80	84	21	< 1
HST-050-003	1.00249	1.00375	1.00360	126	111	21	< 1
HST-050-004	1.00257	1.00347	1.00344	90	87	23	< 1
HST-050-005	0.99976	1.00024	1.00093	48	117	23	> 3
HST-050-006	1.00755	1.00837	1.00844	82	89	20	< 1
HST-050-007	0.99622	0.99740	0.99748	118	126	20	< 1
HST-050-008	0.99633	0.99719	0.99752	86	119	21	< 2
HST-050-009	0.99471	0.99591	0.99620	120	149	20	< 2
HST-050-010	0.97854	0.97921	0.97941	67	87	20	< 2
HST-050-011	0.99026	0.99077	0.99069	51	43	20	< 1

- Compared to MCNP6.2, Crater gives “statistically equivalent” results
  - 60% of the Crater Δkeff are within one standard deviation of the MCNP6.2 Δkeff

Conclusions

- At LANL, work is underway to develop a comprehensive package of python tools for benchmarking, sensitivity analysis, etc.

- Crater : calculate the impact of nuclear data changes on responses
  - Folds changes in a nuclear data parameter with the corresponding sensitivity profile
  - A fast alternative to performing the actual transport calculations
  - Currently works only for cross section changes

- The future of Crater
  - Extend the capability to angular data and particle spectra
  - Extend the capability to higher dimensional observables

Acknowledgements

- Research reported in this publication was supported by the U.S. Department of Energy LDRD program at Los Alamos National Laboratory.

- We gratefully acknowledge the support of the Advanced Simulation and Computing (ASC) program at Los Alamos National Laboratory.

- This work was supported in part by the DOE Nuclear Criticality Safety Program, funded and managed by the National Nuclear Security Administration for the Department of Energy.