Title: Improved Verification and Validation Testing and Tools

Author(s): Rising, Michael Evan
Josey, Colin James
Kulesza, Joel A.

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

Issued: 2021-07-16 (rev.1)

Disclaimer:
Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Improved Verification and Validation Testing and Tools

***Michael E. Rising**, Colin J. Josey, and Joel A. Kulesza*

Monte Carlo Codes Group (XCP-3)

2021 MCNP® User Symposium
July 12-16, 2021

LA-UR-21-26448

Overview

- Primary goal of software testing
  - The role of verification and validation
- Previously released V&V suites
- New Python-based framework
- Additional test suite(s)

Primary goal of software testing

- Test the code for *correctness*
- Correctness is defined with respect to some standard
  - Comparison to another code (version)
  - Comparison to (semi-)analytic results
  - Comparison to experiment measurements

Primary goal of software testing

Current MCNP6
Testing Practices

- Test the code for *correctness*
- Correctness is defined with respect to some standard
  - Comparison to another code (version)

    Behavioral testing done for every code change during development

    Full end-to-end testing attempting to isolate behaviors / features

  - Comparison to (semi-)analytic results

    Ensuring the algorithms indeed solve the transport equation

    Simplified problems and mock data used to isolate code / algorithm implementation

  - Comparison to experiment measurements

    Ensuring the combination of algorithms and data compare well to nature / reality

    Applies only to application area being tested and compared

Primary goal of software testing

Current MCNP6
Testing Practices

- Test the code for *correctness*
- Correctness is defined with respect to some standard
  - Comparison to another code (version)
    - Behavioral testing done for every code change during development
    - Full end-to-end testing attempting to isolate behaviors / features

REGRESSION

  - Comparison to (semi-)analytic results
    - Ensuring the algorithms indeed solve the transport equation
    - Simplified problems and mock data used to isolate code / algorithm implementation

VERIFICATION

  - Comparison to experiment measurements
    - Ensuring the combination of algorithms and data compare well to nature / reality
    - Applies only to application area being tested and compared

VALIDATION

Role of Verification and Validation

- Verification
  - Where analytical and semi-analytical solutions to the transport equation may exist, we want to ensure that MCNP is solving the correct equations

- Validation
  - Combination of code (MCNP) and nuclear data (ENDF/NJOY/ACE) work together to produce results comparable to reality

- Full end-to-end tests exercising many separate features

  (input parsing, problem setup, nuclear data usage & collision physics, transport & random walk algorithm, tallying, dose/response functions, output, etc.)

- Long-standing reputation can be linked to extensive and robust V&V

Previously Released V&V Suites

MCNP6.2 Release

- Verification
  - k-effective (VERIFICATION_KEFF)
  - 3-D fixed-source streaming (KOBAYASHI)
  - Variety of shielding problems (VERIFICIATION_SHLD_SVDM)

- Validation
  - k-effective (VALIDATION_ CRITICALITY & VALIDATION_CRIT_EXPANDED)
  - 3-D fixed-source neutron and photon problems (VALIDATION_SHIELDING)

Previous Releases

- High-energy physics (CEM & LAQGSM)

Previously Released V&V Suites

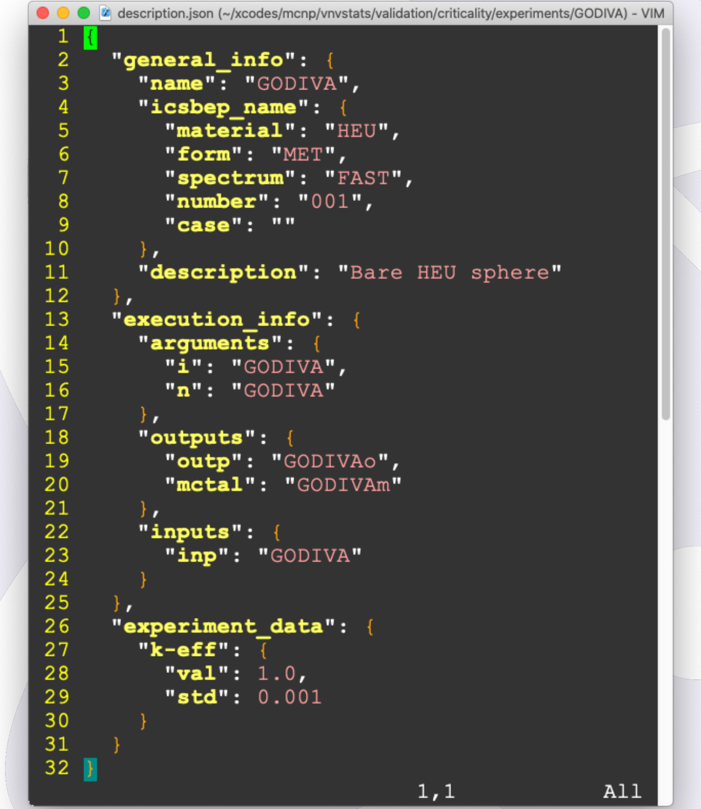
Limitations in previously released V&V suites

- Mixture of Makefile, Perl, Windows .bat scripts used to execute problems (`ALL`)
    - Missing execution scripts entirely (`CEM` & `LAQGSM`)
- Problems cannot be run directly without preprocessing or suite-specific XSDIR files (`CRITICALITY` & `CRIT_EXPANDED`)
- Misleading suite not doing actual verification (`SHLD_SVDM`)
- Postprocessing results scripts inconsistent and/or missing (`SHIELDING`, `CEM` & `LAQGSM`)
- No job submission / cluster support (`ALL`)
- Plotting / visualization support missing, broken, or incomplete (`ALL`)
- Any sort of documentation requires manual intervention (`ALL`)

New Python-based Framework

- **Consistency** across suites
- **Extensible** to more suites and problem types
- **Automated** for all steps
  - Setup
  - Execute
  - Postprocess
  - Document
- Requires Python3
- Runs on Linux, Mac OS, & Windows

description.json file


description.json (~/xcodes/mcnp/vnvstats/validation/criticality/experiments/GODIVA) - VIM
1,1
All

```
{
  "general_info": {
    "name": "GODIVA",
    "icsbep_name": {
      "material": "HEU",
      "form": "MET",
      "spectrum": "FAST",
      "number": "001",
      "case": ""
    },
    "description": "Bare HEU sphere"
  },
  "execution_info": {
    "arguments": {
      "i": "GODIVA",
      "n": "GODIVA"
    },
    "outputs": {
      "outp": "GODIVAo",
      "mctal": "GODIVAm"
    },
    "inputs": {
      "inp": "GODIVA"
    }
  },
  "experiment_data": {
    "k-eff": {
      "val": 1.0,
      "std": 0.001
    }
  }
}
```

New Python-based Framework

- Can be immediately used for any version of the code

  (input and data options must be considered)

- For developers
  - Can test code and data frequently
  - V&V reports are essential for a release

- For everyone else
  - Can add application-specific V&V suites
  - Can support SQA needs

Directory structure

```
vnvstats /
    |- README.md
    |- support /
        |- mcnpvnv.py           (MCNP-specific V&V functionality)
        |- vnv /                (Generic V&V functionality)
            |- README.md
            |- __init__.py
            |- benchcalc.py     (Benchmark/experiment handling)
            |- commandline.py   (Command line parser and execution)
            |- compare.py       (Compare results from calculations/experiments)
            |- plotndoc.py      (Tabulation of results, plotting and documentation)
            |- slurmin.py       (Support SLURM submission and job execution)
            |- tests /
    |- validation /
        |- criticality /
            |- README.md
            |- VnV.py           (Drives criticality suite)
            |- experiments /    (Contains each benchmark model and description)
        |- shielding /
        |- lockwood /
        |- ...
    |- verification /
        |- keff /
        |- kobayashi /
        |- ...
```

New Python-based Framework

- List
  - Query test suite for available test problems

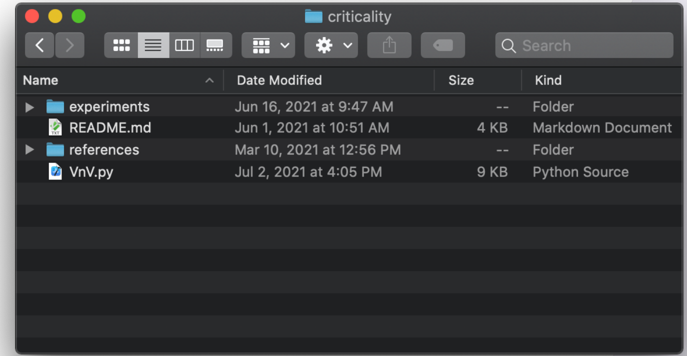
```
python VnV.py list
```

```
criticality $ python VnV.py list
All available tests in validation criticality:
  BAWXI2
  BIGTEN
  FLAT23
  FLAT25
  FLATPU
  FLSTF1
  GODIVA
  GODIVR
  HISHPG
  ICT2C3
  IMF03
  IMF04
  JEZ233
  JEZ240
  JEZPU
  LST2C2
  ORNL10
  ORNL11
  PNL2
  PNL33
  PUBTNS
  PUSH2O
  SB25
  SB5RN3
  STACY36
  THOR
  TT2C11
  UH3C6
  UMF5C2
  ZEBR8H
  ZEUS2
```

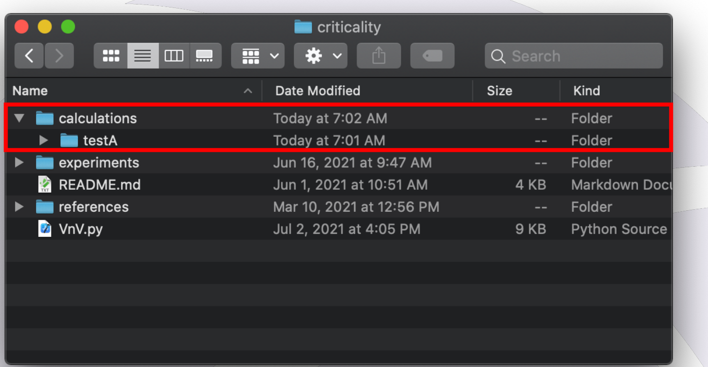
New Python-based Framework

- Setup
  - Creates a calculation tree of benchmarks selected

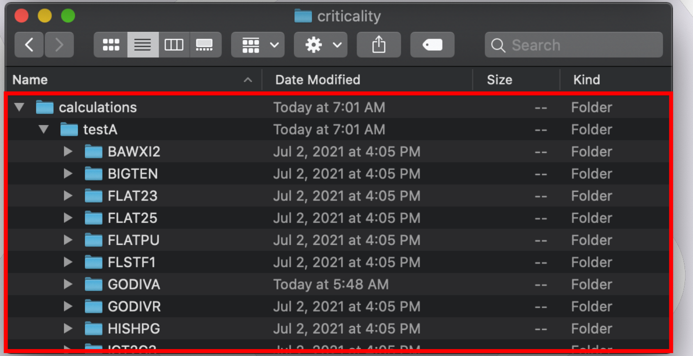
```
python VnV.py setup --calcdir_name testA
```



Name	Date Modified	Size	Kind
▶ experiments	Jun 16, 2021 at 9:47 AM	--	Folder
README.md	Jun 1, 2021 at 10:51 AM	4 KB	Markdown Document
▶ references	Mar 10, 2021 at 12:56 PM	--	Folder
VnV.py	Jul 2, 2021 at 4:05 PM	9 KB	Python Source



Name	Date Modified	Size	Kind
▼ calculations	Today at 7:02 AM	--	Folder
▶ testA	Today at 7:01 AM	--	Folder
▶ experiments	Jun 16, 2021 at 9:47 AM	--	Folder
README.md	Jun 1, 2021 at 10:51 AM	4 KB	Markdown Docu
▶ references	Mar 10, 2021 at 12:56 PM	--	Folder
VnV.py	Jul 2, 2021 at 4:05 PM	9 KB	Python Source



Name	Date Modified	Size	Kind
▼ calculations	Today at 7:01 AM	--	Folder
▼ testA	Today at 7:01 AM	--	Folder
▶ BAWXI2	Jul 2, 2021 at 4:05 PM	--	Folder
▶ BIGTEN	Jul 2, 2021 at 4:05 PM	--	Folder
▶ FLAT23	Jul 2, 2021 at 4:05 PM	--	Folder
▶ FLAT25	Jul 2, 2021 at 4:05 PM	--	Folder
▶ FLATPU	Jul 2, 2021 at 4:05 PM	--	Folder
▶ FLSTF1	Jul 2, 2021 at 4:05 PM	--	Folder
▶ GODIVA	Today at 5:48 AM	--	Folder
▶ GODIVR	Jul 2, 2021 at 4:05 PM	--	Folder
▶ HISHPG	Jul 2, 2021 at 4:05 PM	--	Folder

  - Example of calculation tree with only listed benchmarks

```
python VnV.py setup --calcdir_name testB BIGTEN FLAT25 GODIVA
```

New Python-based Framework

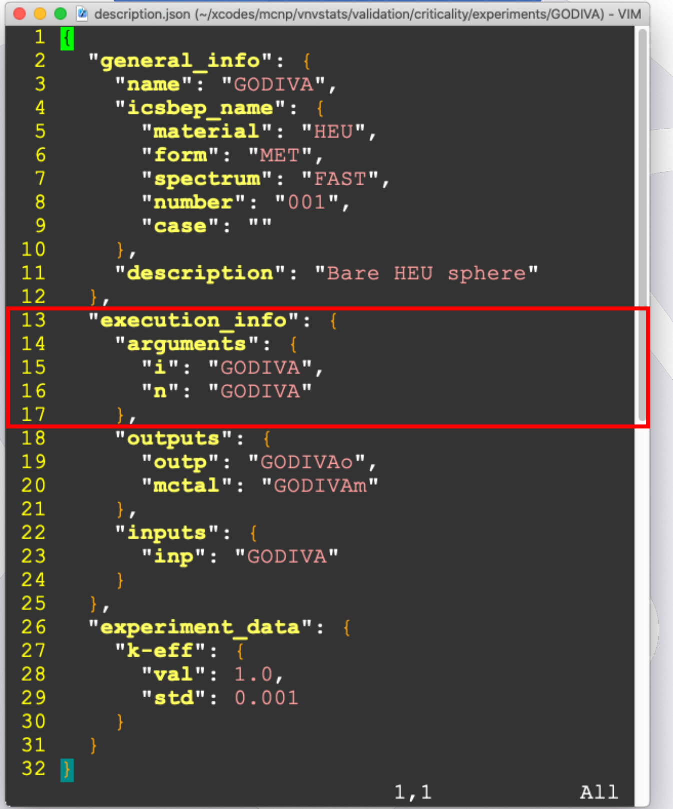
- Execution
  - Runs all problems in existing calculation directory

```
python VnV.py execute --calcdir_name testA
```

  - Builds command line from execution_info group
  - Option examples:

```
--executable_name mcnp6
--jobs 2                    concurrent execution
--ntrd 8                    threads for each job
--nmpi 4                    ranks for each job
```

description.json file


description.json (~/xcodes/mcnp/vnvstats/validation/criticality/experiments/GODIVA) - VIM

```
{
  "general_info": {
    "name": "GODIVA",
    "icsbep_name": {
      "material": "HEU",
      "form": "MET",
      "spectrum": "FAST",
      "number": "001",
      "case": ""
    },
    "description": "Bare HEU sphere"
  },
  "execution_info": {
    "arguments": {
      "i": "GODIVA",
      "n": "GODIVA"
    },
    "outputs": {
      "outp": "GODIVAo",
      "mctal": "GODIVAm"
    },
    "inputs": {
      "inp": "GODIVA"
    }
  },
  "experiment_data": {
    "k-eff": {
      "val": 1.0,
      "std": 0.001
    }
  }
}
```

New Python-based Framework

- Execution Submission
  - Submits all problems in existing calculation directory via slurm/sbatch

```
python VnV.py execute_slurm --calcdir_name testA
```

  - Option examples:

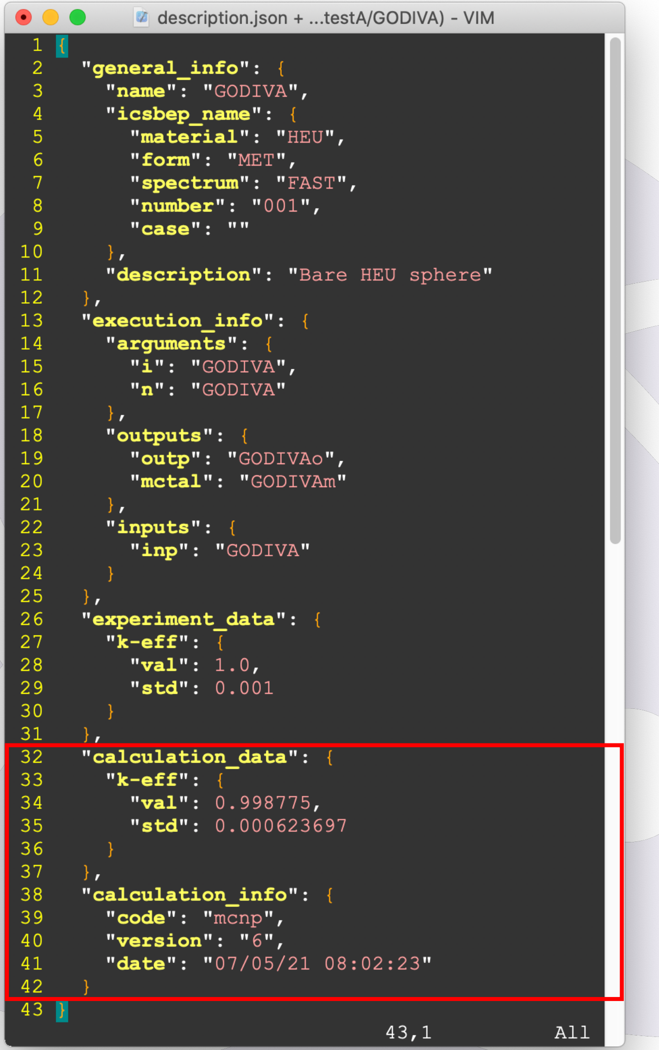
`--nodes 1`	node allocation
`--time 120`	time allocation in minutes
`--stride 8`	jobs per sbatch job submitted
`--wait`	wait for execution to complete before proceeding
`--pre_cmd`	commands to run before and/or after MCNP execution within sbatch submission script
`--post_cmd`	

New Python-based Framework

- Postprocessing
  - Reads calculation output files and processes results into calculation description.json

    ```
    python VnV.py postprocess \
    --calcdir_name testA
    ```

  - Adds calculation_data and calculation_info objects to JSON file
    - experiment_data and calculation_data directly comparable
  - All suites will likely postprocess MCNP results differently
  - Using MCNPTools wherever possible



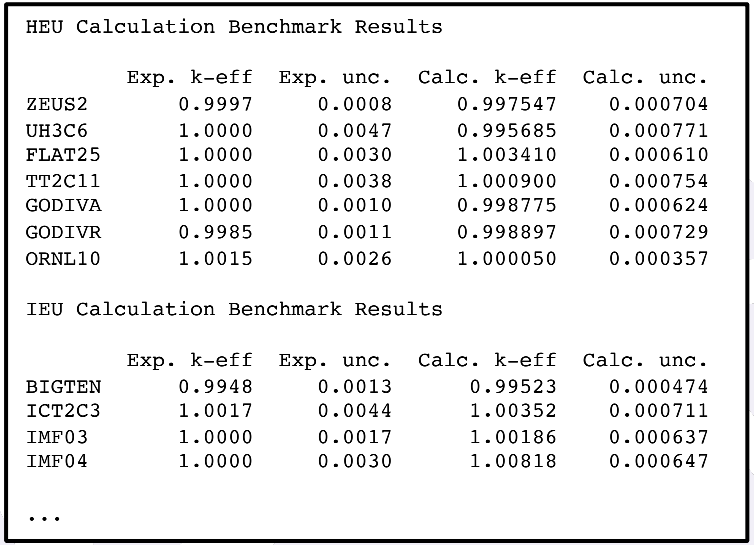
```
{
  "general_info": {
    "name": "GODIVA",
    "icsbep_name": {
      "material": "HEU",
      "form": "MET",
      "spectrum": "FAST",
      "number": "001",
      "case": ""
    },
    "description": "Bare HEU sphere"
  },
  "execution_info": {
    "arguments": {
      "i": "GODIVA",
      "n": "GODIVA"
    },
    "outputs": {
      "outp": "GODIVAo",
      "mctal": "GODIVAm"
    },
    "inputs": {
      "inp": "GODIVA"
    }
  },
  "experiment_data": {
    "k-eff": {
      "val": 1.0,
      "std": 0.001
    }
  },
  "calculation_data": {
    "k-eff": {
      "val": 0.998775,
      "std": 0.000623697
    }
  },
  "calculation_info": {
    "code": "mcnp",
    "version": "6",
    "date": "07/05/21 08:02:23"
  }
}
                                        43,1          All
```

New Python-based Framework

- Documentation
  - Retrieves experiment and simulation results from calculation description.json and prepares documentation

```
python VnV.py document \
--calcdir_name testA
```

  - Results are tabulated into text and LaTeX form
  - Plots are generated into PNG outputs
  - Between LaTeX text, tables, and PNG plots, a V&V report is nearly done



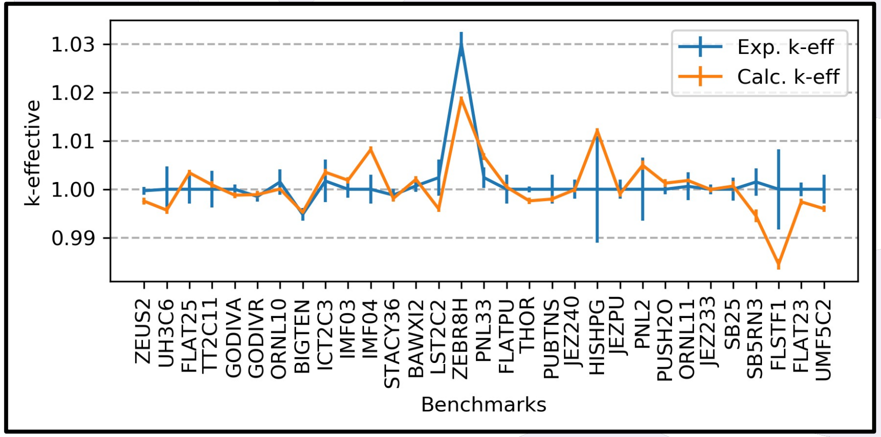
```
HEU Calculation Benchmark Results

          Exp. k-eff   Exp. unc.   Calc. k-eff   Calc. unc.
ZEUS2         0.9997      0.0008      0.997547     0.000704
UH3C6         1.0000      0.0047      0.995685     0.000771
FLAT25        1.0000      0.0030      1.003410     0.000610
TT2C11        1.0000      0.0038      1.000900     0.000754
GODIVA        1.0000      0.0010      0.998775     0.000624
GODIVR        0.9985      0.0011      0.998897     0.000729
ORNL10        1.0015      0.0026      1.000050     0.000357

IEU Calculation Benchmark Results

          Exp. k-eff   Exp. unc.   Calc. k-eff   Calc. unc.
BIGTEN        0.9948      0.0013       0.99523     0.000474
ICT2C3        1.0017      0.0044       1.00352     0.000711
IMF03         1.0000      0.0017       1.00186     0.000637
IMF04         1.0000      0.0030       1.00818     0.000647

...
```



New Python-based Framework

- Nominal workflow
  - Setup, execute, postprocess, and document a suite of test problems

```
python VnV.py setup \
  --calcdir_name MCNP63_VV

python VnV.py execute \
  --calcdir_name MCNP63_VV

python VnV.py postprocess \
  --calcdir_name MCNP63_VV

python VnV.py document \
  --calcdir_name MCNP63_VV
```

DRAFT

MCNP®

Code Version 6.3.0

Verification & Validation Testing

Los Alamos
NATIONAL LABORATORY

Note: this is under active development and some changes may occur before official release

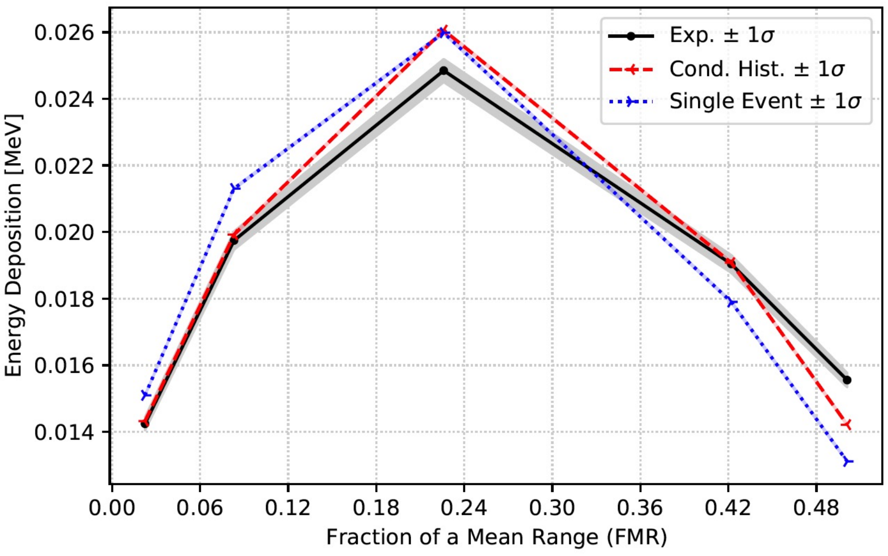
Additional Test Suite(s)

- Beyond the actual MCNP input files, two ingredients are required to create a new suite:
  - **description.json** files, each benchmark (easy)
    - `execution_info` : maps to MCNP command line options/arguments and input/output files
    - `experiment_data` : benchmark results used to compare to calculation results
  - **VnV.py** script, each suite (medium/hard)
    - `list` : same for all test suites
    - `setup` : same for all test suites (except where additional options are wanted, see bonus slide)
    - `execute` : same for all test suites
    - `execute_slurm` : same for all test suites
    - `postprocess` : unique to every test suite
    - `document` : unique to every test suite

This is likely where the most time is spent getting each suite setup

Additional Test Suite(s)

- Finished incorporating Lockwood validation test suite
  - Electron transport energy deposition
    - Condensed history algorithm
    - Single event electrons
  - Several materials
  - 334 separate MCNP inputs
  - Reasonably computationally expensive (need cluster / high performance computing)



Aluminum 0-degree 0.3 MeV Results

- Resurrecting LAQGSM and CEM validation test suites
  - No Makefile or other scripts to execute code and/or postprocess results
  - Gaining experience through old tests, documentation and trail of bread crumbs…

Summary

- All MCNP team supported V&V test suites are now developed in a separate repository from the MCNP source code within a Python-based framework
  - Python tools and scripts
  - Benchmark inputs and description JSON files
- This entire framework will be distributed with the upcoming MCNP6.3 release
- Most V&V test suites distributed with MCNP6.2 will be distributed in new framework
- New V&V test suites are done or being worked on for the MCNP6.3 release
- Looking forward to feedback and potential contributions

Questions?

Contact: mrising@lanl.gov

Suite specific command line options

- Easy to add command line options in VnV.py scripts for each individual suite

```
command_args["setup"].add_argument(
    "--data",
    type=str,
    choices=["endf66", "endf70", "endf71", "endf80"],
    default="endf71",
    help="Data library to use, default endf71",
)
```

- Criticality and Rossi-alpha suites have --data option for setup step:

```
python VnV.py setup --calcdir_name test_endf71 --data endf71
python VnV.py setup --calcdir_name test_endf80 --data endf80
```

- Separate calculation tree for each --data option selected