

Streamlining the Coupling of BISON and Dakota through the NEAMS Workbench



Kaylee M. Cunningham
Robert A. Lefebvre
Michael R. Tonks
Jacob A. Hirschhorn
Ian T. Greenquist
Jeffrey J. Powers

July 2021



DOCUMENT AVAILABILITY

Reports produced after January 1, 1996, are generally available free via US Department of Energy (DOE) SciTech Connect.

Website www.osti.gov

Reports produced before January 1, 1996, may be purchased by members of the public from the following source:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone 703-605-6000 (1-800-553-6847)
TDD 703-487-4639
Fax 703-605-6900
E-mail info@ntis.gov
Website <http://classic.ntis.gov/>

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange representatives, and International Nuclear Information System representatives from the following source:

Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831
Telephone 865-576-8401
Fax 865-576-5728
E-mail reports@osti.gov
Website <https://www.osti.gov/>

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Nuclear Energy and Fuel Cycle Division

**STREAMLINING THE COUPLING OF BISON AND DAKOTA THROUGH THE
NEAMS WORKBENCH**

Kaylee M. Cunningham^{*}
Robert A. Lefebvre[†]
Michael R. Tonks^{*}
Jacob A. Hirschhorn[†]
Ian T. Greenquist[†]
Jeffrey J. Powers[†]

^{*}University of Florida

[†]Oak Ridge National Laboratory

July 2021

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-BATTELLE, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

FIGURES.....	iv
ACRONYMS.....	v
ACKNOWLEDGMENTS	vi
ABSTRACT	1
1. INTRODUCTION	1
1.1 VTR	1
1.2 NEAMS	2
2. BACKGROUND	2
2.1 NEAMS WORKBENCH.....	2
2.2 BISON	2
2.3 DAKOTA	3
2.4 IFR-1 EXPERIMENT	3
2.5 X430 EXPERIMENT	3
3. SETUP.....	3
3.1 WORKBENCH SETUP	3
3.2 MOOSE AND DAKOTA COMBINED SETUP	5
3.3 BISON SETUP	5
4. NEAMS WORKBENCH-DAKOTA INTEGRATION	5
4.1 IFR-1 BENCHMARK	6
4.2 OBSTACLES	11
4.3 X430 EXPERIMENT ANALYSIS	12
5. CONCLUSION	12
6. REFERENCES	14
APPENDIX A. BISON-DAKOTA INTEGRATION THROUGH NEAMS WORKBENCH TUTORIAL	A-1

FIGURES

Figure 1. MOOSE dos2unix setup script.	4
Figure 2. Workbench job scheduler specifications particular to the UF HiperGator cluster.....	4
Figure 3. Communication type for Windows SSH.....	4
Figure 4. Updated entry.sh file.	5
Figure 5. Updating the username in moose_submit.sh.....	5
Figure 6. Energy per fission versus burnup simple correlation matrix generated by NEAMS Workbench on Windows.	6
Figure 7. Energy per fission versus burnup simple correlation matrix generated by NEAMS Workbench on Mac.....	7
Figure 8. Peak cladding temperature (K) versus evaluation number (corresponding to different energy per fission values). Figure generated by Workbench.	8
Figure 9. Peak end of life outlet temperature (K) versus evaluation number (corresponding to different fission gas release values). Figure generated by Workbench.	9
Figure 10. Burnup (at. %) versus evaluation number (corresponding to different cladding wall height values). Figure generated by Workbench.	10
Figure 11. Cladding creep strain versus evaluation number (corresponding to different cladding wall height values). Figure generated by Workbench.	11
Figure 12. Depiction of Dakota input file.....	12
Figure A.1. How to access NEAMS Workbench Help Documentation.	A-3
Figure A.2. Selecting “Create Template.”	A-4
Figure A.3. Line 136 templated using angle brackets.	A-5
Figure A.4. Illustration of Dakota selected in drop-down bars, auto-complete feature, and input validation feature.	A-6
Figure A.5. Launch the Dakota input by selecting “Run. ”	A-6
Figure A.6. Generic Dakota input file.	A-7
Figure A.7. An example of a Dakota driver file.	A-8
Figure A.8. Generic Dakota driver file (note that the scheduler header and submit path are specific to a SLURM based scheduler).	A-8
Figure A.9. Launch the Dakota input by selecting “Run. ”	A-9
Figure A.10. Demonstration of “Open associated files”.	A-9
Figure A.11. Using Processors tool to generate figures.	A-10
Figure A.12. Workbench plotter tool.....	A-10

ACRONYMS

Dakota	Design Analysis Kit for Optimization and Terascale Applications
DOE-NE	Department of Energy Office of Nuclear Energy
EBR-II	Experimental Breeder Reactor II
FFTF	Fast Flux Test Facility
INL	Idaho National Laboratory
MOOSE	Multiphysics Object Oriented Simulation Environment
NEAMS	Nuclear Energy Advanced Modeling and Simulation
ORNL	Oak Ridge National Laboratory
SSH	Secure shell
UF	University of Florida
VTR	Versatile Test Reactor

ACKNOWLEDGMENTS

This work is the result of ongoing efforts supporting the Versatile Test Reactor project and was sponsored by the US Department of Energy Office of Nuclear Energy.

ABSTRACT

Metallic nuclear fuels for use in advanced reactors are an active area of research and development. Robust, accurate metallic fuel performance models are necessary for the design, analysis, and licensing of such reactors. However, metallic fuel performance models require additional development; they are not as mature as uranium dioxide fuel performance models. To support further metallic fuel development, Oak Ridge National Laboratory and the University of Florida have streamlined the coupling of the BISON fuel performance code with Design Analysis Kit for Optimization and Terascale Applications (Dakota) statistical analysis tool through the Nuclear Energy Advanced Modeling and Simulation (NEAMS) Workbench. This work included performing three different sensitivity analyses on metallic nuclear fuel models in BISON. The analyses examined were a general model of the IFR-1 experiment, the X430 experiment T654 pin, and the X430 experiment T651 pin. The results suggest that BISON and Dakota can be integrated through NEAMS Workbench to perform sensitivity and uncertainty analyses and visualize the results.

1. INTRODUCTION

The US Department of Energy Office of Nuclear Energy (DOE-NE) aims to stimulate and support the construction of new nuclear reactors across the country by providing new testing capabilities for fuels and materials while continuing to improve safety, scale, and cost. As a result, various efforts have been introduced to accelerate this process, including the Versatile Test Reactor (VTR) project and the Nuclear Energy Advanced Modeling and Simulation (NEAMS) program.

This report documents work completed between June 2020 and December 2020 under the “Integration of BISON and Dakota using NEAMS Workbench to assist in modeling metal fuel performance” project, which was a collaboration between the University of Florida (UF) and Oak Ridge National Laboratory (ORNL). The goal of this work was to help develop, demonstrate, and document an integration approach through Workbench to enable streamlined BISON/Dakota sensitivity analysis for metallic fuels. Workbench was coupled to BISON and Dakota to run sensitivity analyses on a remote cluster and visualize simulation results on a local machine. Issues were reported to the Workbench development team, and the process was documented to facilitate future analyses and code development.

ORNL funded UF for this work in support of the VTR project. This work involved actual or possible export-controlled information and export-controlled computer software (e.g., BISON). Researchers at UF involved in this work were responsible for and ensured that proper access controls were maintained for any such information and software. Furthermore, to ensure proper information security, no export-controlled information or software was provided by ORNL (or the VTR project) to UF for this work.

1.1 VTR

The VTR is a new fast spectrum test reactor that is currently being developed in the United States under the direction of DOE-NE [1]. The VTR mission is to enable accelerated testing of advanced reactor fuels and other materials required for advanced reactor technologies. The mission includes neutron irradiation capabilities that would support testing under alternate coolants, including sodium, molten salt, lead/lead-bismuth eutectic mixture, and gas. The VTR aims at addressing most of the needs of the various stakeholders, primarily composed of advanced reactor developers, as well as a number of other interested parties. Design activities are underway at multiple national laboratories, industry partners, and universities targeting a first-criticality date as early as 2026, depending upon funding support and future acquisition decisions. Current efforts are focused on all aspects of the VTR design. The current proposed

VTR concept is a 300 MWth sodium-cooled pool-type reactor with metallic uranium-plutonium-zirconium alloy fuel.

1.2 NEAMS

The NEAMS program is focused on developing new computational tools to analyze and optimize the performance and reliability of existing and advanced nuclear power plants. NEAMS is developing tools that allow researchers to investigate current problems and new ideas in unique ways that were previously considered impractical due to the high level of detail required. NEAMS can enhance understanding in areas ranging from changes in nuclear fuel materials to full-scale power plant operations [2]. The NEAMS mission is to develop, apply, deploy, and support state-of-the-art predictive modeling and simulation tools for design and analysis of current and future nuclear energy systems. This is accomplished by using computing architectures ranging from laptops to leadership-class computing facilities. The tools developed by NEAMS will enable transformative scientific discoveries and insights otherwise not attainable or affordable and will accelerate solution of existing problems as well as deployment of new designs for current and advanced reactors [2,3].

2. BACKGROUND

BISON and Dakota were initially set up on a cluster via a passwordless secure shell (SSH) through NEAMS Workbench using a Windows operating system. Later, the authors installed Workbench on a Mac operating system to compare Windows and Mac capabilities and to enhance security.

2.1 NEAMS WORKBENCH

The NEAMS Workbench is being developed to facilitate the transition from conventional to high-fidelity computational tools by providing a common user interface for model creation, review, execution, output review, and visualization for integrated codes [2].

The NEAMS Workbench can use common user input, including engineering-scale specifications, that are expanded into application-specific input requirements using customizable templates. The templating process enables multifidelity analysis of a system from a common set of input data. Additionally, the common user input processor can provide an enhanced alternative application input that is more convenient than the native input, especially for legacy codes. The various integrated codes and application templates available in NEAMS Workbench broaden the user community and facilitate system analysis and design [2].

Ultimately, the NEAMS Workbench improves usability and streamlines the process of utilizing the BISON tool, locally and remotely [4]. The software consists of an advanced text editor that incorporates input validation as well as an auto-complete tool. Data plotting is also available to assist the user with analysis of input and output data through the graphical user interface. Through these capabilities the NEAMS Workbench provides a common analysis environment that aids in accelerating the user's adoption of advanced computational tools such as BISON.

2.2 BISON

The BISON fuel performance code is a finite element method-based software tool being developed by various organizations and led by the Idaho National Laboratory (INL) for DOE-NE. BISON is based on INL's Multiphysics Object Oriented Simulation Environment (MOOSE). The code can simulate a host of fuel and cladding compositions and geometries, including light-water reactor fuel rods, TRISO (tri-structural isotropic) particle fuel, metallic fuel, and plate fuel. Depending on the problem type and

complexity level, varying dimensionality can be used in BISON, including 1D, 1.5D, 2D axisymmetric, 2D radial-azimuthal, and 3D models. Visualizations can be created through third-party software such as Paraview or VisIt using the results stored in output Exodus files. They may also be viewed through MOOSE's native visualization program, Peacock. BISON calculations include tensor mechanics, temperature- and burnup-dependent thermal properties, fission product swelling, thermal and irradiation creep, fission gas production and release, irradiation-induced swelling, and other phenomena [5].

2.3 DAKOTA

Dakota (Design Analysis Kit for Optimization and Terascale Applications) is a tool kit developed by Sandia National Laboratories. Dakota allows users to apply sensitivity analysis and uncertainty quantification techniques to their codes. Analyses are done by varying selected inputs in several simulations. The uncertainty is then determined by measuring the variation in the simulation outputs [6]. More specifically, Dakota is a wrapper program that creates input files and runs BISON on its own. Several uncertainty quantification methods are available. Monte Carlo uncertainty quantification is the simplest because it does not require the user to know the governing equations, but it is relatively computationally expensive.

For the first analysis, BISON and Dakota were coupled to analyze a U-19Pu-10Zr* ternary metallic fuel alloy based on the IFR-1 experiment. The IFR-1 BISON input file is currently available in the BISON repository. The second and third analyses were performed on the X430 experiment's T651 and T654 fuel pins. The T651 fuel pin was a U-10Zr alloy; the T654 pin was a U-19Pu-10Zr alloy.

2.4 IFR-1 EXPERIMENT

The IFR-1 experiment included irradiation of three fuel compositions: U-10Zr, U-8Pu-10Zr, and U-19Pu-10Zr. All were clad in 20% cold-worked D9 steel. Of the 169 fuel pins in the IFR-1 experiment, 18 were U-19Pu-10Zr, 19 were U-8Pu-10Zr, and 132 were U-10Zr. The experiment began irradiation in 1985 at the Fast Flux Test Facility (FFTF). IFR-1 fuel pins had an active fuel column height of 91.4 cm and contained axial blankets of depleted U-10Zr that were 16.5 cm long and were situated above and below the main fuel column [7]. Post-irradiation examinations demonstrated similarities between the IFR-1 fuel pins and the shorter fuel pins (34.3 cm active fuel column height) used in experiments conducted in Experimental Breeder Reactor II (EBR-II) [8].

2.5 X430 EXPERIMENT

The X430 experiment included irradiation of the following fuel compositions: U-10Zr, U-19Pu-10Zr, U-22Pu-10Zr, and U-26Pu-10Zr. All of them were clad in HT9 steel. The fuel assembly contained 37 fuel pins with an active fuel column height of 34.3 cm. This experiment was conducted at EBR-II in the late 1980s and early 1990s [9].

3. SETUP

3.1 WORKBENCH SETUP

The NEAMS Workbench was installed on a local Windows machine and the UF HiperGator cluster to allow simulations to be run remotely. A Windows zip file was downloaded from the NEAMS website to the local Windows machine and unzipped to set up and run Workbench. The Linux tar file for Workbench was downloaded from the same website and untarred on the HiperGator cluster. The Workbench setup

*All compositions in this work are expressed in weight percent unless noted otherwise.

instructions in the provided help documentation were utilized to configure the SSH to the cluster. The best practice is to follow the instructions listed in the Workbench Help Documentation and to use the Windows cmd terminal. Configuration files for MOOSE, Dakota, and BISON were generated using the Workbench auto-complete tool. The configuration files were used to set up Workbench so that it can identify and differentiate between each code (i.e., MOOSE, Dakota, and BISON).

To run BISON and MOOSE on Windows, an application script must be used to prevent Windows-to-Linux “dos2unix” errors. An example of a MOOSE script can be seen in Figure 1.

```
#!/bin/bash
for I in $@; do if [ ${i##*.} == I ]; then dos2unix $i; fi; done
MOOSE=~/projects/moose/modules/combined/combined-opt
if [ -z "$PBS_NUM_PNN" ]
then
$MOOSE "$@"
else
mpirun -np ${PBS_NUM_PNN} $MOOSE "$@"
fi
```

Figure 1. MOOSE dos2unix setup script.

The cluster scheduler specifications must be input in the Workbench setup file. The specifications shown in Figure 2 were included in the setup file because HiperGator uses a SLURM based scheduler.

```
scheduler_type = pbs
scheduler_specs {
    submit_path = "/opt/slurm/bin/sbatch"
    status_path = "/opt/slurm/bin/squeue"
    delete_path = "/opt/slurm/bin/scancel"
    hold_path = "/opt/slurm/bin/scontrol hold"
    release_path = "/opt/slurm/bin/scontrol release"
}
Scheduler_header [
    "#!/bin/bash"
]
```

Figure 2. Workbench job scheduler specifications particular to the UF HiperGator cluster.

Communication paths also needed to be specified for SSH configurations (Figure 3).

```
communication_type = ssh
ssh_specs {
    ssh_path = "C:\Windows\System32\OpenSSH\ssh.exe"
    ssh_options = "-T"
    scp_path = "C:\Windows\System32\OpenSSH\scp.exe"
    scp_options = "-pCqr"
}
```

Figure 3. Communication type for Windows SSH.

A minor bug was found in the interface between Python and Workbench. An existing PYTHONHOME alias pointed to version 3.6.5. This bug may appear if the user’s Python version is not up to date or if Python aliases are defined on the system. To fix the bug, the commands “unset PYTHONPATH” and “unset PYTHONHOME” were added to the entry.sh file located on the cluster at ~/Workbench-Linux/rte/entry.sh (see Figure 4).

```
#!/bin/bash
#https://stackoverflow.com/questions/59895/getting-the-source-directory-of-a-bash-
script-from-within
unset PYTHONPATH
unset PYTHONHOME

SOURCE="${BASH_SOURCE[0]}"
while [ -h "$SOURCE" ]; do # resolve $SOURCE until the file is no longer a symlink
  SELFDIR="$( cd -P "$( dirname "$SOURCE" )" && pwd )"
  SOURCE="$(readlink "$SOURCE")"
  [[ $SOURCE != /* ]] && SOURCE="$DIR/$SOURCE" # if $SOURCE was a relative symlink,
  we need to resolve it relative to the path where the symlink file was located
```

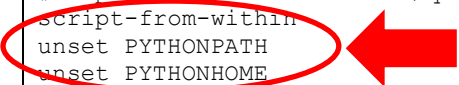


Figure 4. Updated entry.sh file.

When working on a Mac device, it is important to note that on occasion security parameters can prevent the simulation from running. This was the main obstacle faced after the transition from Windows to Mac. The work-around for this issue is to open a terminal window and navigate to the home directory and execute the following command: “sudo xattr -dr com.apple.quarantine PATH/TO/WORKBENCH”.

Passwordless SSH setup is simple to configure in the Mac operating system when following the Workbench Help Documentation instructions.

3.2 MOOSE AND DAKOTA COMBINED SETUP

MOOSE and Dakota were installed together on the cluster. The only obstacle faced with the MOOSE and Dakota setup was updating the username in the `moose_submit.sh` script, which is located in the `moose_dakota_test.zip` file provided by UF (see Figure 5).

```
#!/bin/sh
#SBATCH --job-name=moose_test.          #Job name
#SBATCH --ntasks=1                     #Number of tasks/MPI Ranks
#SBATCH --cpus-per-task=1               #Number of CPU per Task
#SBATCH --mem=1gb                       #Memory (120 gig/server)
#SBATCH --time=1:00:00                 #Walltime days-hh:mm:ss
#SBATCH --output=job-%j.out             #Output and error log
#SBATCH --account=michael.tonks         #Allocation group name, add -b for burst job

srun --mpi=pmix_v2 /home/USERNAME/projects/moose/modules/combined/combined-opt -i
moose.i > moose.out
```

Figure 5. Updating the username in moose_submit.sh.

The software installation was verified by comparing MOOSE predictions to an analytical solution for a simple 1D heat conduction equation. The MOOSE prediction was accurate.

3.3 BISON SETUP

BISON was installed from INL’s Gitlab repository. Similar procedures for the MOOSE-Dakota setup were followed.

4. NEAMS WORKBENCH-DAKOTA INTEGRATION

After completion of the setup, the integration of Workbench with BISON and Dakota was streamlined. Running Dakota simulations is simplified through Workbench, even when coupling with BISON, because

Workbench interfaces with both codes to run simulations on the remote cluster and makes the results available for visualization on the local machine. The general workflow consists of developing a Dakota input file (.in), a Dakota driver file (.drive), a BISON input file (.i), and a template of that BISON input file (.tmpl). Step-by-step instructions for how to run Dakota coupled with BISON through the NEAMS Workbench are provided in A-1.

4.1 IFR-1 BENCHMARK

Sensitivity and uncertainty analyses were performed using Dakota coupled with BISON's IFR-1 input file and then compared with previous analyses by Greenquist et al., which were performed without using Workbench [10]. Comparing the analyses in this work to those of Greenquist et al. helps to verify the Workbench implementation and setup.

A simplified correlation matrix was successfully generated using Dakota to analyze the correlations between the defined energy value and the calculated burnup value for the IFR-1 benchmark BISON input file. Dakota performed a multidimensional parameter study using 1 partition, 300 means, and 10 standard deviations. The matrix was created first on Windows (Figure 6) and later on Mac (Figure 7). Comparison of the two matrices verified that Workbench ran effectively on both operating systems.

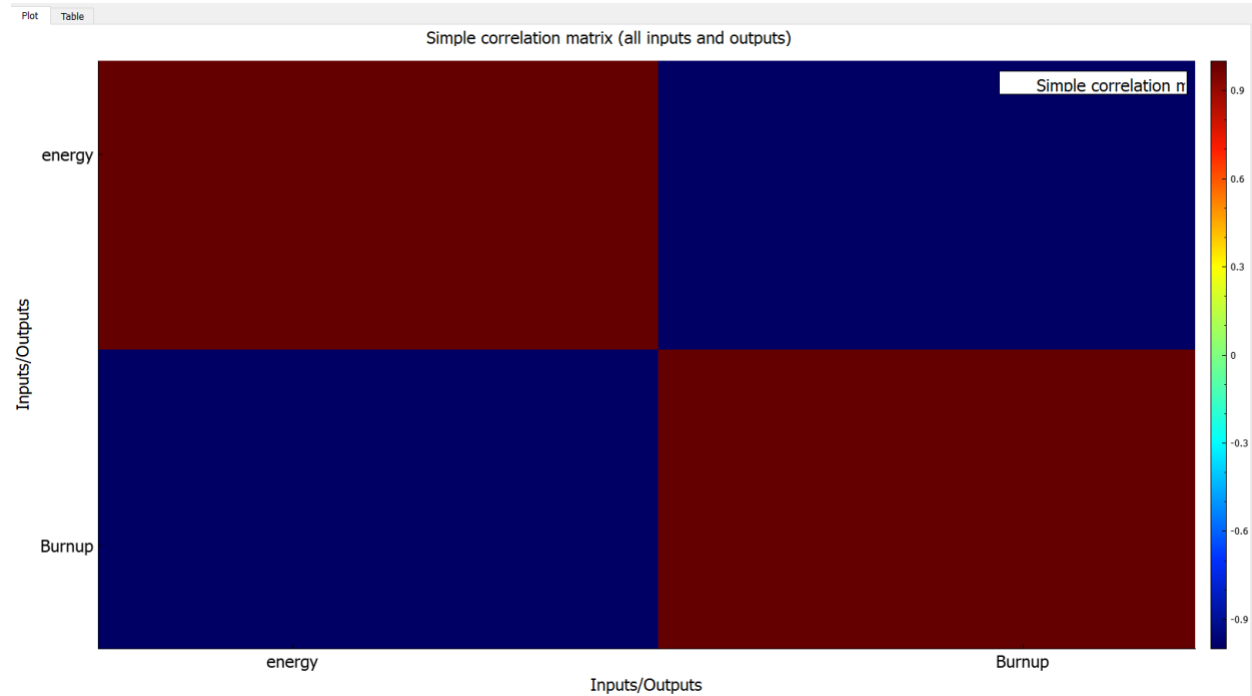


Figure 6. Energy per fission versus burnup simple correlation matrix generated by NEAMS Workbench on Windows.

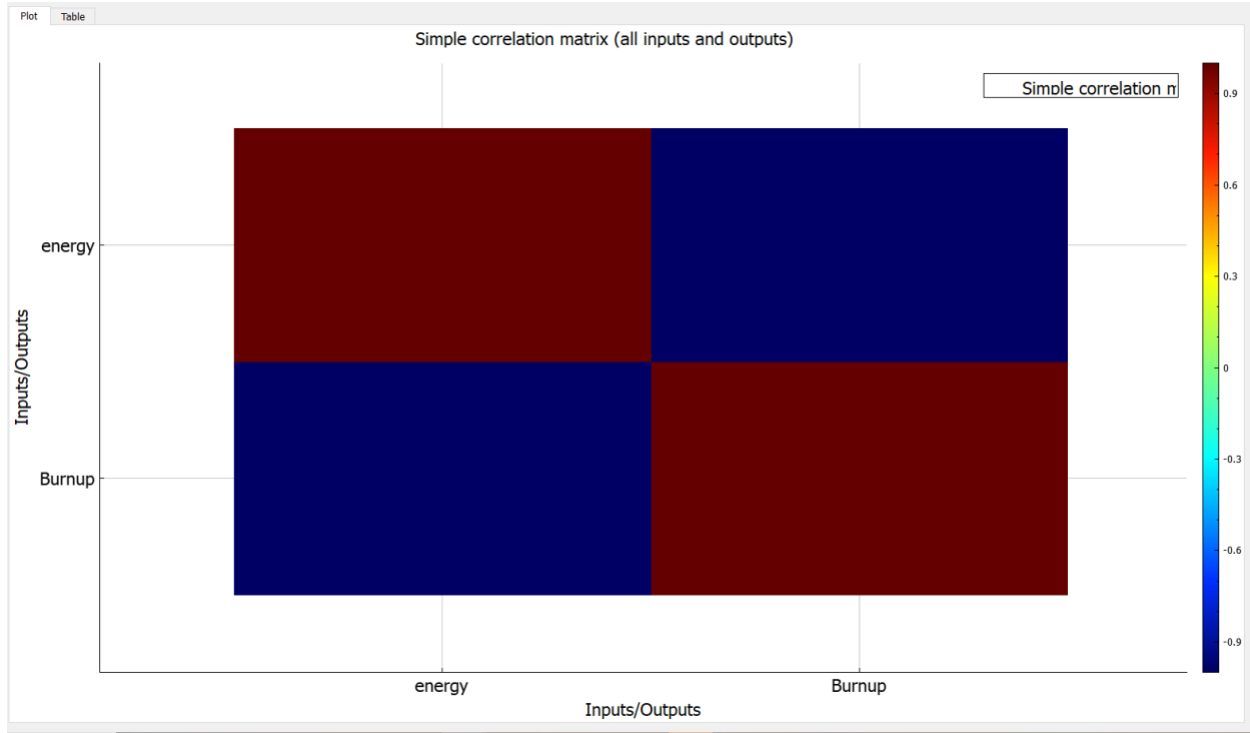


Figure 7. Energy per fission versus burnup simple correlation matrix generated by NEAMS Workbench on Mac.

Figures Figure 6 and Figure 7 illustrate Workbench's post-processor capability to generate a visualization of the simplified correlation matrix produced by Dakota. The dark red illustrates a correlation coefficient of 1; the dark blue illustrates a correlation coefficient of -1 , corresponding to a negative correlation. The test was intended to verify that Workbench, BISON, and Dakota were successfully coupled and running together. It also demonstrated that the Workbench visualization capability was working properly. However, the results were not intended to be interpreted physically because only two data points were collected. The purpose of including the figures is to illustrate the visualization capabilities that Workbench is equipped with. Both figures are identical, thus illustrating the capabilities of Workbench across operating systems.

Next, to further illustrate the Dakota-BISON integration and visualization features of the NEAMS Workbench, a sensitivity analysis was performed to evaluate the impacts of energy variations on peak cladding temperature. The analysis was used as additional verification (see Figure 8). The results indicate that variations in energy per fission within the range examined do not have a significant impact on predicted cladding temperatures, correlating correctly with the results of Greenquist et al. [10].

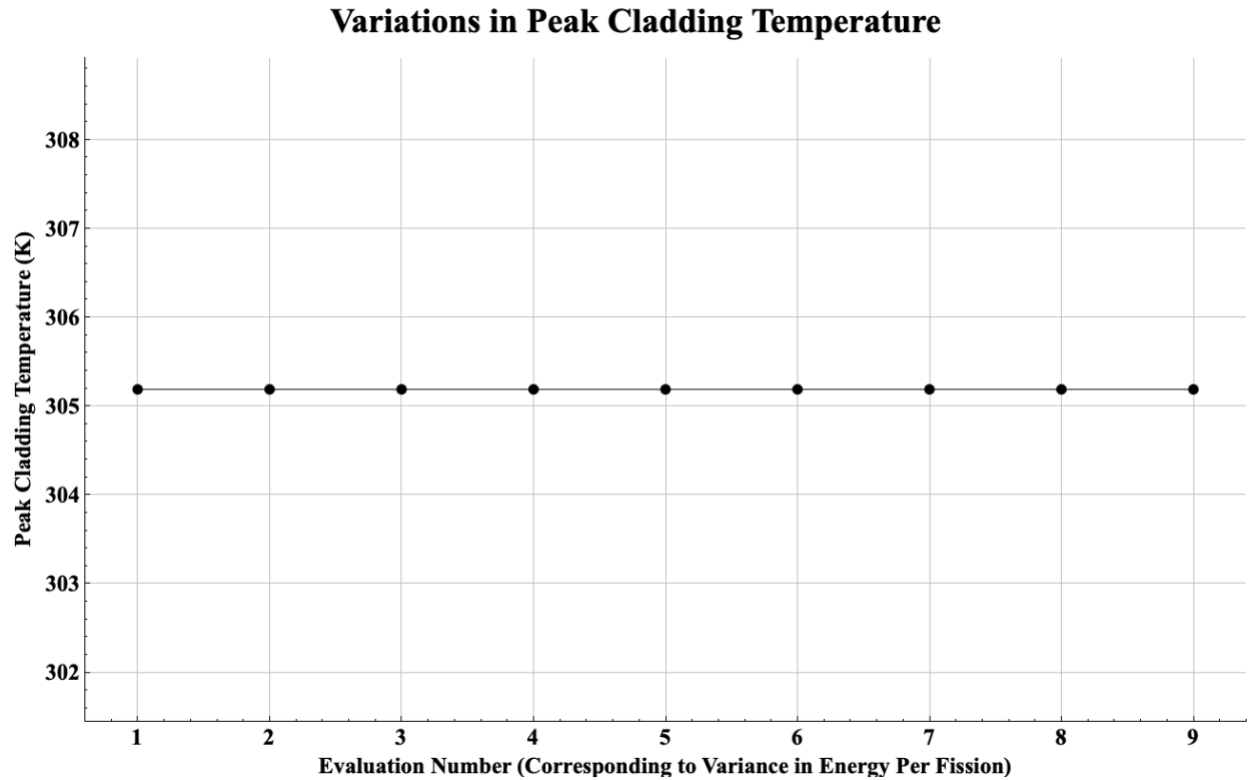


Figure 8. Peak cladding temperature (K) versus evaluation number (corresponding to different energy per fission values). Figure generated by Workbench.

The Workbench integration method was verified by comparing another of the analyses performed by Greenquist et al. to a generated output to study the impact of varying the mean of the fission gas release fraction on peak outlet temperature. The results matched up with the results of the previously performed analysis using alternative methods and demonstrated the lack of an impact fission gas release has on peak coolant outlet temperature (see Figure 9).

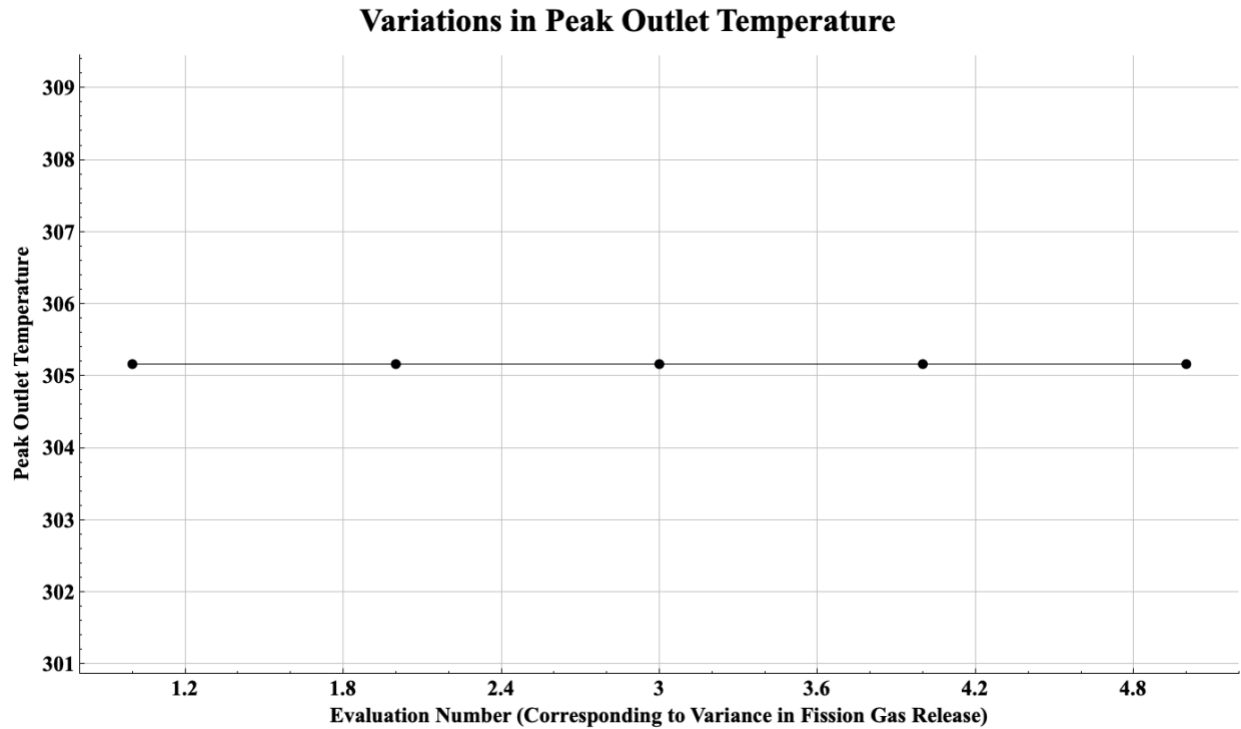
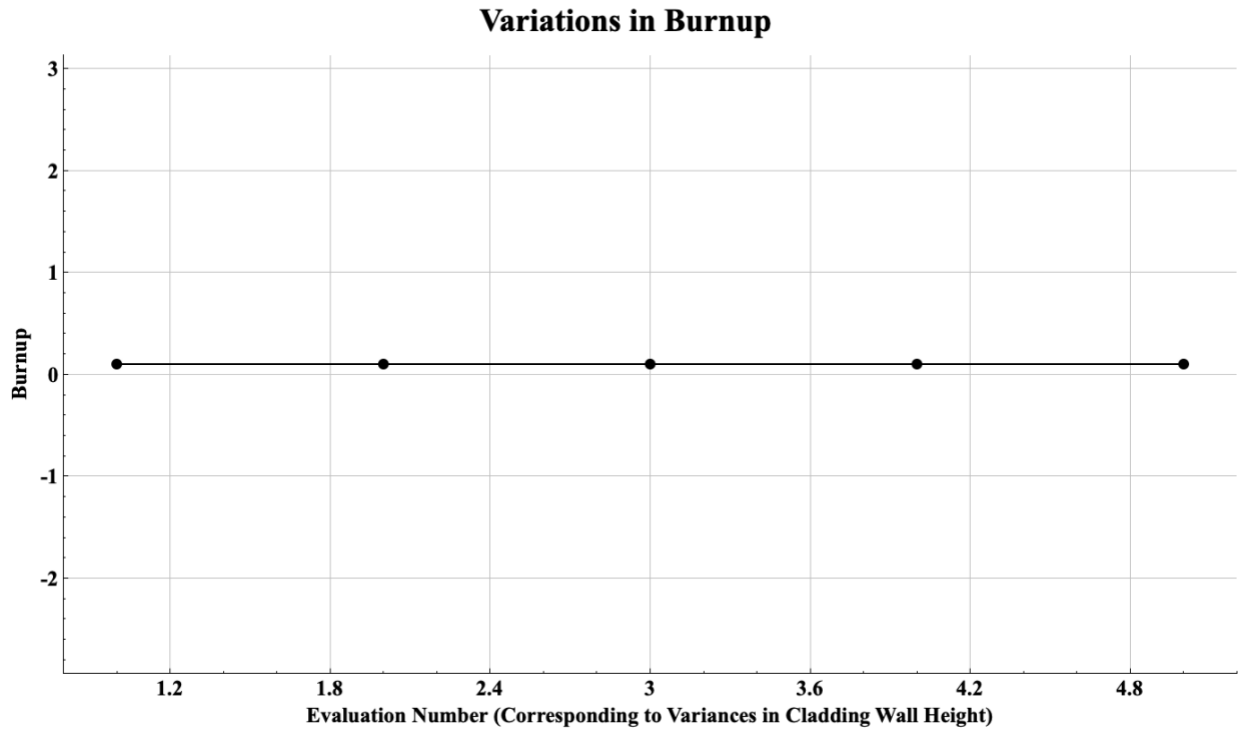


Figure 9. Peak end of life outlet temperature (K) versus evaluation number (corresponding to different fission gas release values). Figure generated by Workbench.

A study was then performed to analyze the impacts of variations in cladding height, which could occur during manufacturing. To start, cladding height variations were compared with end of life burnup values. The results are show in Figure 10.



**Figure 10. Burnup (at. %) versus evaluation number (corresponding to different cladding wall height values).
Figure generated by Workbench.**

Evidently, the cladding height had no impact on end of life burnup, as it stays consistently at ~0.09 across each variation.

Last, cladding wall height was varied to evaluate the impact on cladding creep strain. Figure 11 shows the results. For the few evaluations that were performed, there was some variation. Cladding creep strain generally decreased as cladding wall height increased. The variations in the results shown in Figure 11 may be due to differences in numerical convergence in the simulations.

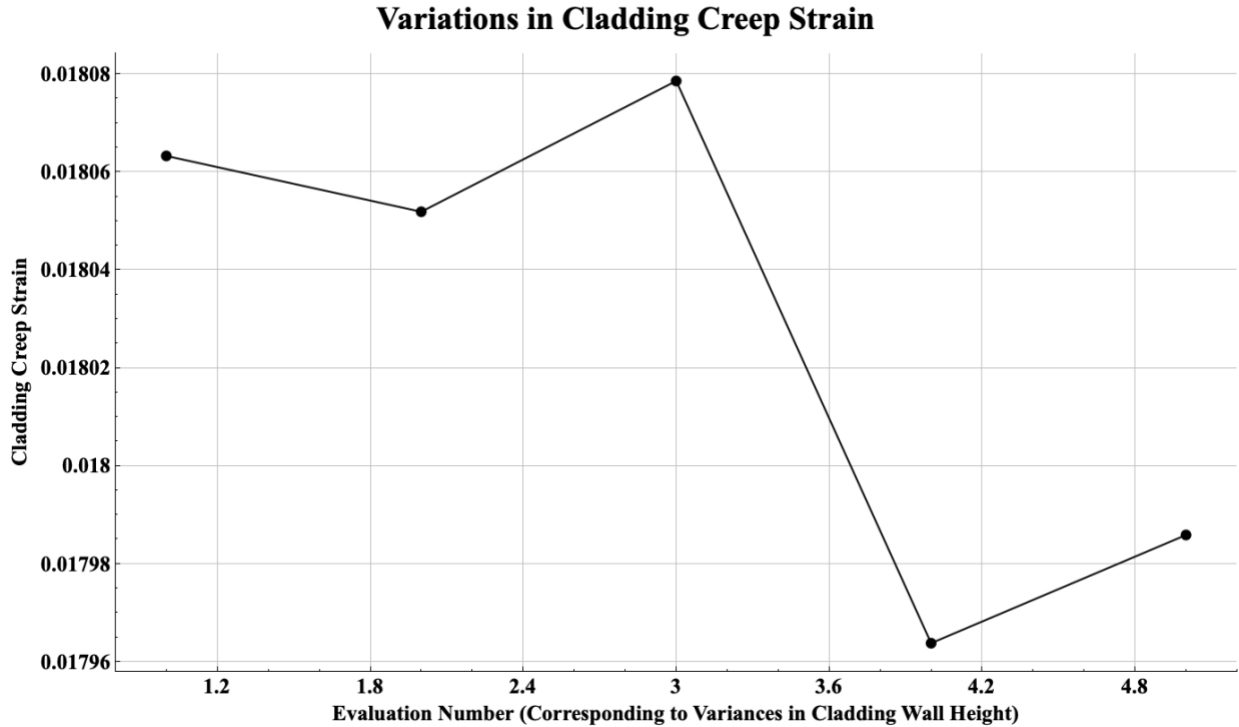


Figure 11. Cladding creep strain versus evaluation number (corresponding to different cladding wall height values). Figure generated by Workbench.

4.2 OBSTACLES

Workbench developers are currently investigating a bug where the last status the Workbench client stores is “running,” even after it is disconnected from the completed job. As a result, the Workbench program thinks the job is still running.

The easiest workaround is to delete the `{FILENAME}.i.status*` files. A `{FILENAME}_dakota.in_{SERVER}.status.json` file contains the last status, which causes the Workbench to attempt to resume status. Killing all processes, removing that file, and rerunning the input avoids this problem. The `.status.json` file will reappear until the file stops running or until the user right-clicks on the messages panel and selects either “terminated” or “killed.” Also, the amount of memory and time required to run the file must be specified in the Dakota driver file (see Figures A.7 and A.8 in APPENDIX A) so it can be passed to the remote scheduler. Without specifying the scheduler header to the corresponding cluster type, the scheduler header will cause the file to crash.

Moreover, it is essential to specify the working directory and to tag the directory for the Dakota files to run in. Including “file_save” ensures the directory will be saved and will be accessible after the Dakota jobs finish (see Figure 12).

```

1 # DAKOTA INPUT FILE
2
3 environment
4     tabular_data
5     tabular_data_file = "ifr_1.1_dakota.dat"
6 method
7     sampling
8     probability_levels = 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 0.95
9     num_probability_levels = 10
10    sample_type lhs
11    samples = 1
12    seed = 3487
13
14 variables
15 normal_uncertain = 1
16     means = 300
17     std_deviations = 10
18     descriptors 'energy'
19
20 interface
21 asynchronous evaluation_concurrency = 1
22     analysis_drivers = '/home/kayleecunningham/Workbench-Linux/rte/entry.sh dakota_driver.py'
23     fork
24         parameters_file = 'params.in'
25         results_file = 'results.out'
26     file_save file_tag
27     work_directory replace directory_tag named = "ifr1_work" directory_save
28     copy_files = "dakota_driver.py" # automatically generated
29                 "ifr_1.1.tmpl" # required to create each application analysis
30                 "ifr_1.1_dakota.drive" # how to pre- and post-process and execu
31

```

Figure 12. Depiction of Dakota input file.

4.3 X430 EXPERIMENT ANALYSIS

A mesh refinement study was performed on X430 benchmark simulations for the T654 and T651 pins using Workbench. After the scheduler header was adapted, Dakota was employed to produce a fuel slug mesh refinement study analyzing the impact of changes in mesh refinement on fission gas release, burnup, and cladding hoop strain predictions. It was a multidimensional parameter study with two partitions. Simulations were run using fuel meshes discretized using 5, 7, and 9 radial elements and 250, 255, and 260 axial elements. Simulations that used fewer than five radial elements had convergence problems and crashed, illustrating the minimum level of radial mesh refinement needed to avoid convergence problems for this benchmark simulation. Variations in mesh refinement within the ranges evaluated had no significant impacts on predicted fission gas release, burnup, or cladding hoop strain.

Another preliminary mesh refinement study was performed on the T654 fuel pin to analyze the impact of axial fuel mesh refinement on predictions of maximum fuel temperature, radial cladding growth, and axial fuel growth. Simulations with fewer than 150 axial mesh elements failed to converge and crashed, pointing toward the minimum level of axial mesh refinement needed to avoid convergence problems for this benchmark simulation. Simulations run with 150, 200, and 250 axial elements produced no significant variations in the 3 parameters evaluated.

5. CONCLUSION

The NEAMS Workbench enables the coupling of BISON and Dakota on both Windows and Mac operating systems. The work performed illustrates Workbench's capabilities in streamlining the coupling of the two codes, creating a convenient platform for sensitivity and uncertainty analysis of nuclear fuel.

Through each analysis performed with data from the IFR-1 experiment, BISON and Dakota's capabilities were verified and demonstrated.

When evaluated, Workbench results produced variations in peak cladding temperature and variations in peak outlet temperature that corresponded to results of Greenquist et al. [10]. Variations in cladding creep strain exhibited some noise, which may have been due to differences in the numerical convergence of the simulations. Preliminary mesh refinement studies helped to identify the minimum amounts of radial and axial mesh refinement needed to avoid convergence problems and to determine ranges of mesh refinement within which certain BISON predictions are relatively stable. Additional analyses would be necessary to fully characterize the effects of mesh refinement on BISON predictions. This work successfully demonstrated integration of BISON and Dakota with NEAMS Workbench and exercised many of its features, laying the groundwork for more analyses like these.

6. REFERENCES

1. K. Pasamehmetoglu, “Versatile Test Reactor Overview,” Advanced Reactors Summit VI, San Diego, California, Jan. 29–31 (2019).
2. R. A. Lefebvre et al., *NEAMS Workbench Status and Capabilities*, ORNL/TM-2019/1314, Oak Ridge National Laboratory, Oak Ridge, Tennessee, September 2019.
3. K. M. Cunningham, J. J. Powers, and R. A. Lefebvre, *Modeling the IFR-1 Experiment: A BISON Metallic Fuel Benchmark*, ORNL/TM-2019/1270, Oak Ridge National Laboratory, Oak Ridge, Tennessee, July 2019.
4. K. M. Cunningham, R. A. Lefebvre, *NEAMS Workbench and Bison Fuel Performance Remote Application Configuration*, ORNL/TM-2019/1208, Oak Ridge National Laboratory, Oak Ridge, Tennessee, September 2019.
5. R. L. Williamson et al., “Multidimensional Multiphysics Simulation of Nuclear Fuel Behavior,” *Journal of Nuclear Materials* 423 (2012) 149–163.
6. B. M. Adams et al., *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 User's Manual*, Sandia National Laboratories, Albuquerque, New Mexico, 2014.
7. A. L. Pitner and R. B. Baker, “Metal Fuel Test Program in the FFTF,” *Journal of Nuclear Materials* 204 (1993) 124–130.
8. D. L. Porter and H. Tsai, *Full-Length Metallic Fast Reactor Fuel Pin Test in FFTF (IFR-1)*, INL/LTD-11-21062, Idaho National Laboratory, Idaho Falls, Idaho, March 2011.
9. A. M. Yacout, *Long-Life Metallic Fuel for the Super Safe, Small and Simple (4S) Reactor*, AFT-2008-000056, Nuclear Regulatory Commission, Washington, DC, June 2008.
10. I. T. Greenquist et al., *A Metallic Fuel Performance Benchmark Problem Based on the IFR-1 Experiment*, ORNL/TM-2020/1534, Oak Ridge National Laboratory, Oak Ridge, Tennessee, June 2020.

APPENDIX A. BISON-DAKOTA INTEGRATION THROUGH NEAMS WORKBENCH TUTORIAL

APPENDIX A. BISON-DAKOTA INTEGRATION THROUGH NEAMS WORKBENCH TUTORIAL

The purpose of this appendix is to provide a comprehensive tutorial of how to execute a Design Analysis Kit for Optimization and Terascale Applications (Dakota) input coupled with BISON through the Nuclear Energy Advanced Modeling and Simulation (NEAMS) Workbench.

This tutorial assumes the user is remotely utilizing Secure Shell to connect to a cluster. It is also assumed that Workbench has already been preconfigured for remote access to said cluster, and that the cluster already has Dakota and BISON installed. It is also assumed the user is familiar with BISON and Dakota. The user must have prepared a BISON input file.

For installation and remote setup instructions, see the Workbench Help Documentation via the button at the top left-hand corner of the interface: Help > Help Documentation (see Figure A.1).

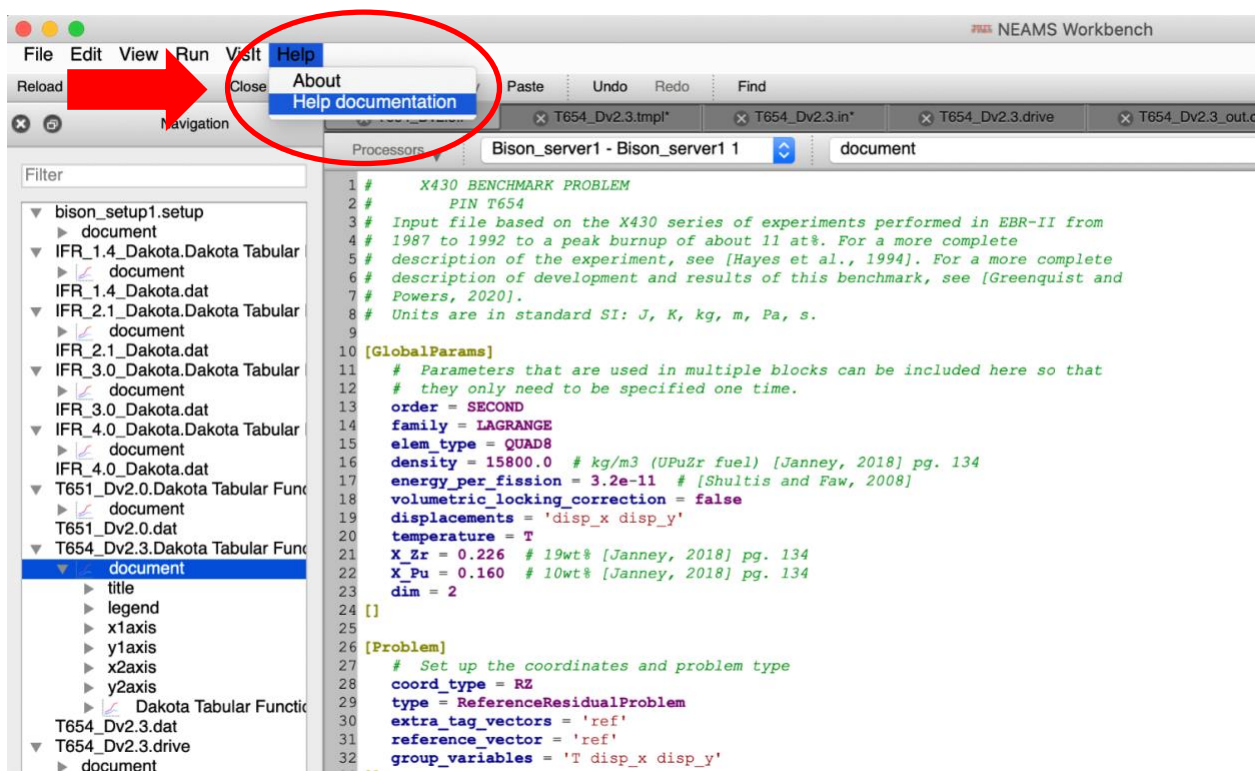


Figure A.1. How to access NEAMS Workbench Help Documentation.

For instructions on how to configure Workbench for remote BISON access, see Reference [A.1].

This tutorial uses the X430 experiment T654 fuel pin input file and a corresponding Dakota input file. The Dakota input file and the Dakota driver file are attached at the end of this appendix.

Start by launching the NEAMS Workbench. Open the BISON input file by selecting File > Open File. The user will need to create a template from the BISON input file. Workbench can generate it.

Right-click on the input file in the Navigation panel. Next, select Create Template (see Figure A.2.)

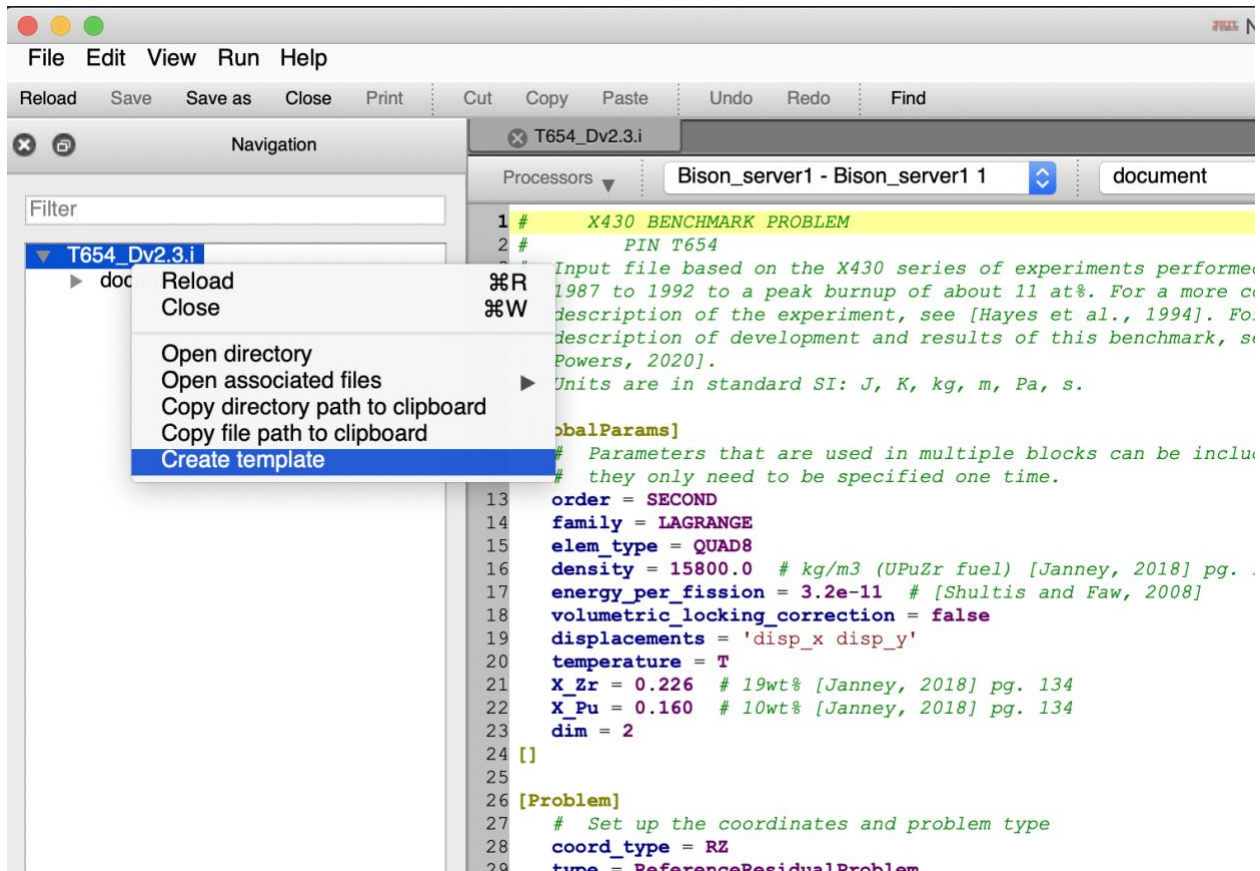


Figure A.2. Selecting “Create Template.”

Workbench will request that the user name the template and select the file’s location. It should be named the same as the BISON input file for simplification purposes later on. The template must be stored in the same directory as the BISON input file to ensure the file is uploaded to the cluster when running.

Next, the user will need to select the parameters to vary from the input file in the Dakota study. In this example, the number of axial elements used to discretize the fuel mesh was varied. Once the parameter is selected, it must be templated by changing the value in the template file to a variable name in angle brackets (see Figure A.3).

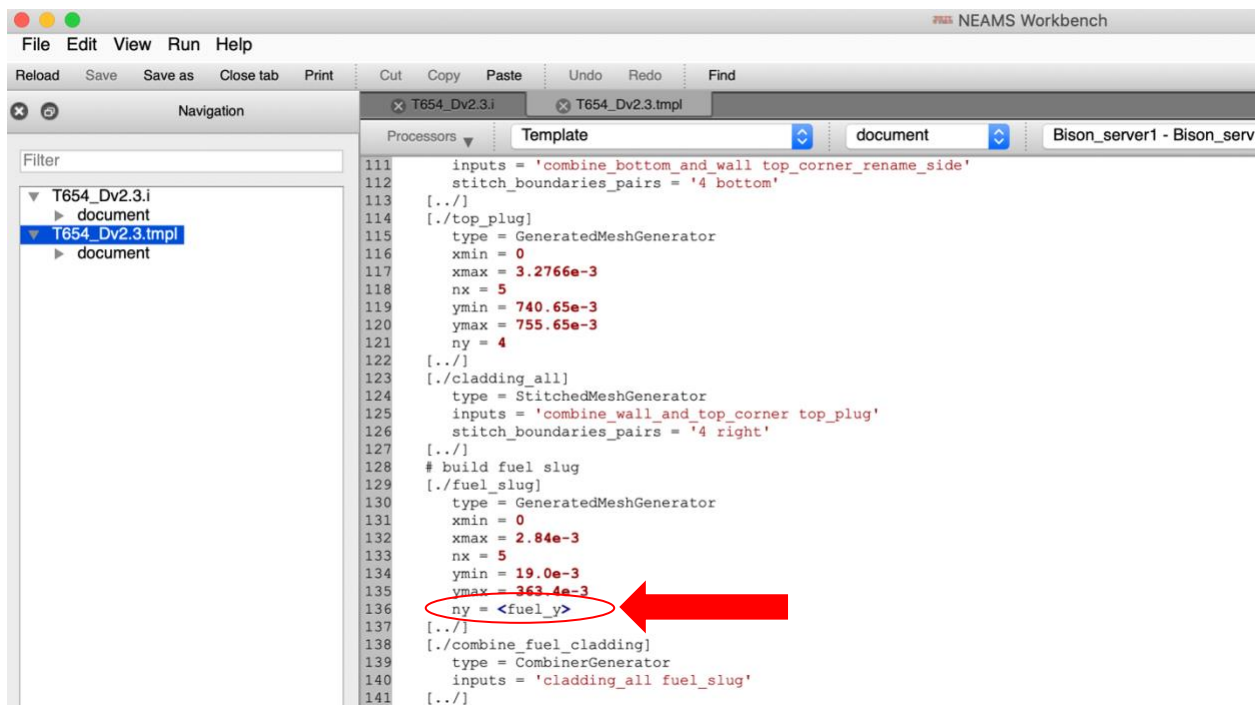


Figure A.3. Line 136 templated using angle brackets.

As illustrated in Figure A.3, line 136 has been templated with `<fuel_y>`.

Next, the Dakota input file must be created. This can be done by selecting `File > New File` and naming the new file the same as the BISON input and template file. For instance, in this example, the Dakota input file is “T654_Dv2.3.in”. Note that the input must be saved as “.in”.

The input file is to be written as a standard Dakota input file. More information on this can be found in Dakota documentation [A.2].

Workbench’s auto-complete tool and validation features can be useful when you are writing the input file. To use auto-complete, place the cursor at the beginning of a new line and press `CTRL+SPACE`. This is the same on both Mac and Windows machines (see Figure A.4).

The input validation feature operates automatically. If any syntax errors are made, the validation panel (select via the bottom right corner, see Figure A.4) will highlight them so they can be quickly and easily corrected.

For these features to work properly, both drop-down tabs at the top of the screen must have “Dakota” selected (see Figure A.4).

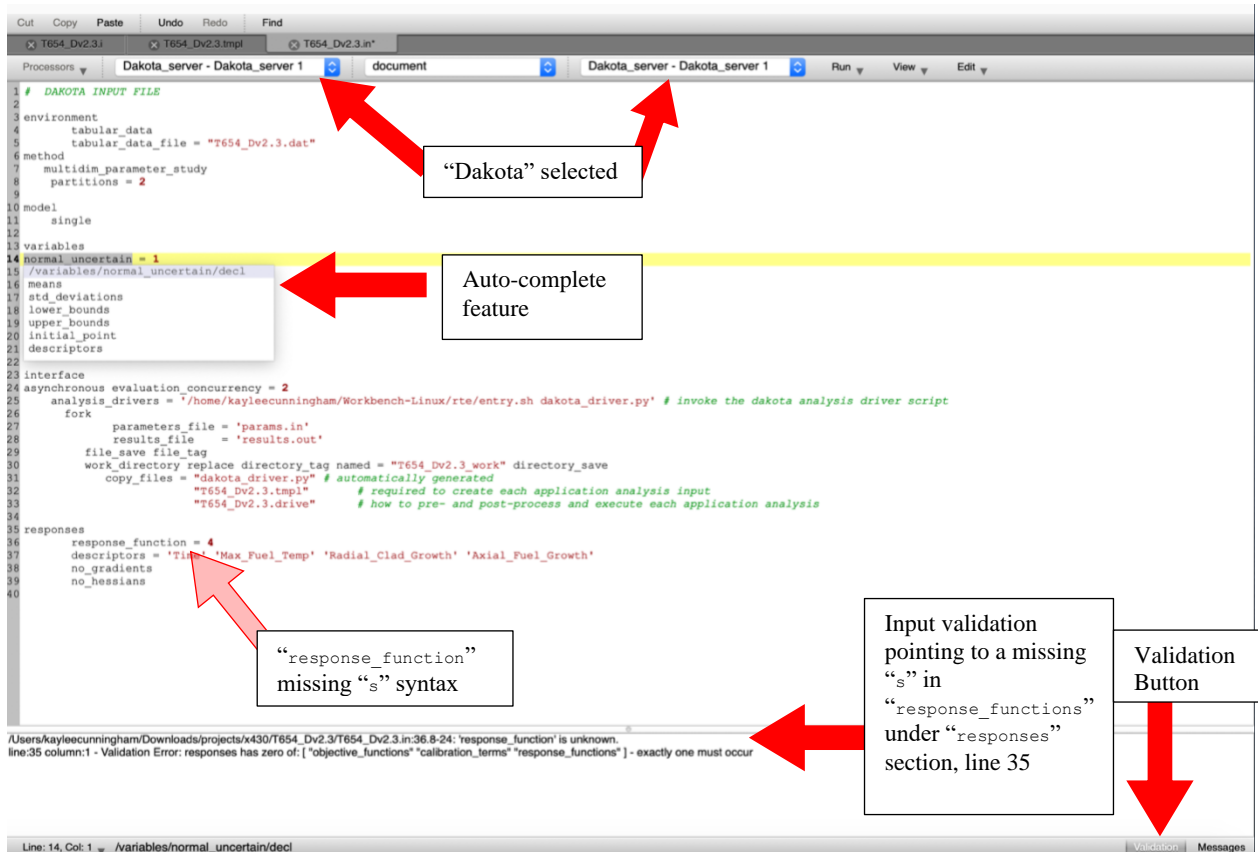


Figure A.4. Illustration of Dakota selected in drop-down bars, auto-complete feature, and input validation feature.

T654_Dv2.3.in is shown in Figure A.5, and a generic version is included in Figure A.6. The following discussions reference both figures.

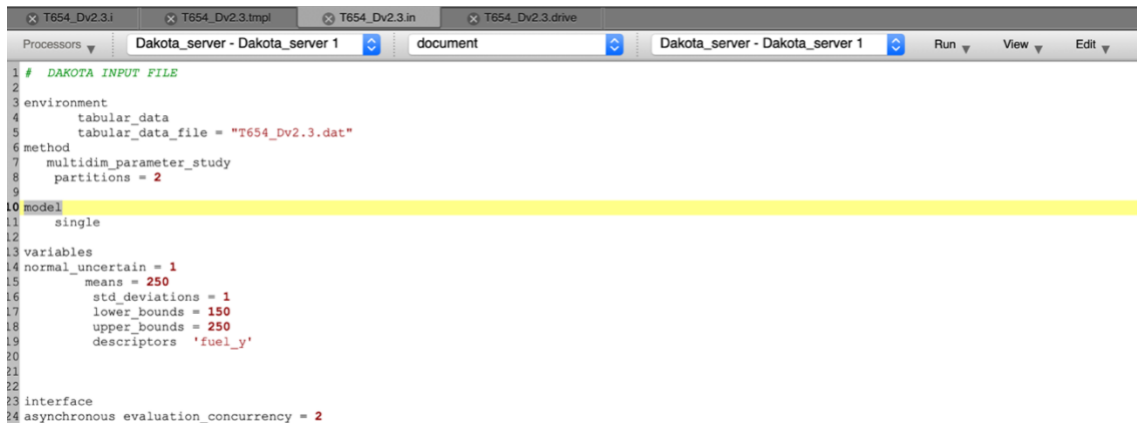


Figure A.5. Launch the Dakota input by selecting "Run."

```

1  # DAKOTA INPUT FILE
2
3  environment
4      tabular_data
5      tabular_data_file = "FILE_BASENAME.dat"
6  method
7      multidim_parameter_study
8      partitions = 2
9
10 model
11     single
12
13 variables
14     normal_uncertain = 1
15     means = 250
16     std_deviations = 1
17     lower_bounds = 150
18     upper_bounds = 250
19     descriptors 'TEMPLATE_VARIABLE'
20
21
22
23 interface
24     asynchronous_evaluation_concurrency = 2
25     # invoke the Dakota analysis driver script
26     analysis_drivers = '/PATH/TO/WORKBENCH/rte/entry.sh dakota_driver.py'
27     fork
28     parameters_file = 'params.in'
29     results_file = 'results.out'
30     file_save file_tag
31     work_directory replace directory_tag named = "FILE_BASENAME_work"
32     directory_save
33     copy_files = "dakota_driver.py" # automatically generated
34                 "FILE_BASENAME.tmpl" # used to create each analysis input
35                 "FILE_BASENAME.drive" # pre- and post-processing and execution
36
37 responses
38     response_functions = 4
39     descriptors = 'OUTPUT_VAR1' 'OUTPUT_VAR2' 'OUTPUT_VAR3' 'OUTPUT_VAR4'
40     no_gradients
41     no_hessians

```

Figure A.6. Generic Dakota input file.

In line 5, the tabular data file should be named the same as the input file name, ending in .dat. Lines 3–22 are specific to the type of analysis the user chooses to run. The variable(s) chosen to template should be included under

```
Variables > descriptors > "variable_name_omit_angle_brackets"
```

Lines 23 through 34 are specific to Workbench. The “asynchronous evaluation_concurrency” (line 24) allows the user to specify the number of BISON inputs that should be run concurrently. The file path on line 26 should be the user’s path to workbench:

```
`PATH/TO/WORKBENCH/rte/entry.sh dakota_driver.py'
```

Lines 33–34 should specify the template and driver file. This is where using the same base name for each file simplifies the input.

Lines 36–41 are specific to the type of analysis the user has chosen. In this example, no Hessians and no gradients are specified. Four responses specific to the BISON output have been chosen and are listed in the “descriptors” in line 39. They correspond directly to columns in the BISON-produced csv file and will be further specified next in the driver file.

The Dakota driver file must be created as well. The Dakota driver file is shown in Figure A.7, and a generic version is included in Figure A.8. The driver file specifies the application path to BISON, the name of the BISON input and template files, the BISON output csv file, and each column in which the response functions are located. The title of each column specified in the driver file must match exactly the descriptor in the Dakota input file. The scheduler header and submit path are each specific to the cluster the user is launching jobs on.

```

1 application '/home/kayleecunningham/projects/bison/bison-opt.sh -i T654_Dv2.3.i'
2   input_file 'T654_Dv2.3.i'
3   input_tmpl 'T654_Dv2.3.tmpl'
4
5 extract_from 'T654_Dv2.3_out.csv' find last_line 1
6   column 1 delimiter ',' as "Time"
7   column 4 delimiter ',' as "Max_Fuel_Temp"
8   column 13 delimiter ',' as "Radial_Clad_Growth"
9   column 14 delimiter ',' as "Axial_Fuel_Growth"
10
11 scheduler
12   header = "#!/bin/bash"
13           "#SBATCH --cpus-per-task=3"
14           "#SBATCH --ntasks=3"
15           "#SBATCH --mem=6gb"
16           "#SBATCH --time=40:00:00"
17
18
19
20 submit_path = '/opt/slurm/bin/sbatch'

```

Figure A.7. An example of a Dakota driver file.

```

1 application '/PATH/TO/BISON/bison/bison-opt.sh -i FILE_BASENAME.i'
2   input_file 'FILE_BASENAME.i'
3   input_tmpl 'FILE_BASENAME.tmpl'
4
5 extract_from 'FILE_BASENAME_out.csv' find last_line 1
6   column 1 delimiter ',' as "OUTPUT_VAR1"
7   column 2 delimiter ',' as "OUTPUT_VAR2"
8   column 3 delimiter ',' as "OUTPUT_VAR3"
9   column 4 delimiter ',' as "OUTPUT_VAR4"
10
11 scheduler
12   header = "#!/bin/bash"
13           "#SBATCH --cpus-per-task=3"
14           "#SBATCH --ntasks=3"
15           "#SBATCH --mem=6gb"
16           "#SBATCH --time=40:00:00"
17
18
19
20 submit_path = '/opt/slurm/bin/sbatch'

```

Figure A.8. Generic Dakota driver file (note that the scheduler header and submit path are specific to a SLURM based scheduler).

Once each of the four files (BISON input `.i`, BISON template `.tmpl`, Dakota input `.in`, and Dakota driver `.drive`) is prepared, the user should return to the Dakota input file to launch the job by clicking the “run” button (see Figure A.9).



Figure A.9. Launch the Dakota input by selecting “Run.”

As the files are running, updates about the jobs will be listed in the “Messages” panel (button next to Validation button, see Figure A.4).

Once the file is finished, the user can right-click on the file in the navigation panel and select “Open associated files” to view outputs (see Figure A.10).

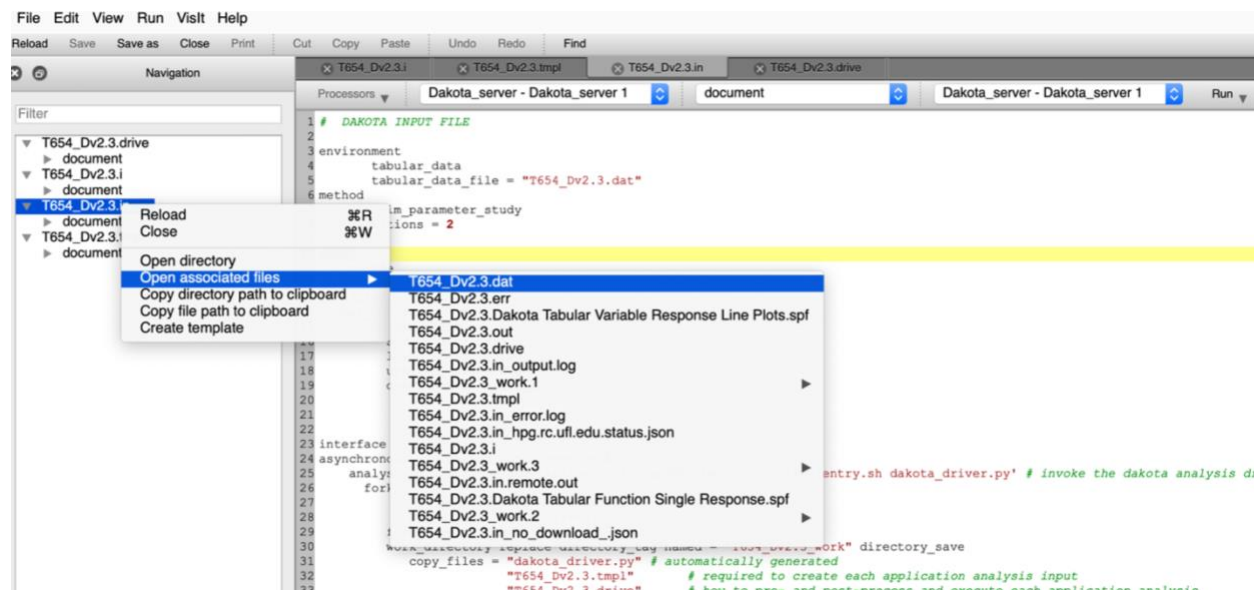


Figure A.10. Demonstration of “Open associated files”.

When an output file, for example `T654_Dv2.3.dat`, is selected, the Processors tool (Figure A.11) can be used to generate figures.

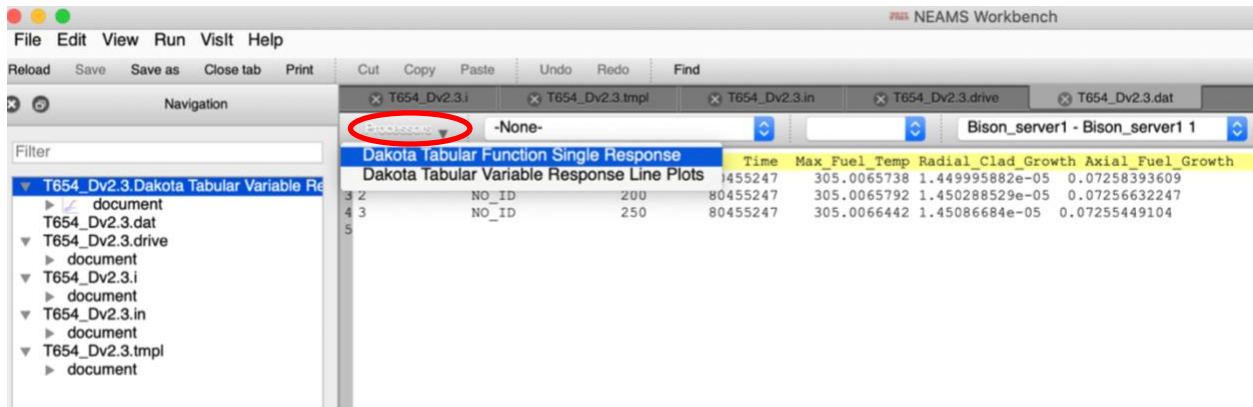


Figure A.11. Using Processors tool to generate figures.

Figure A.12 illustrates the automatically generated plot. By right-clicking and selecting Plot Options (shown in the figure), the user can customize the figure details (e.g., axis titles, the data visible, the font, the line type, the plot type).

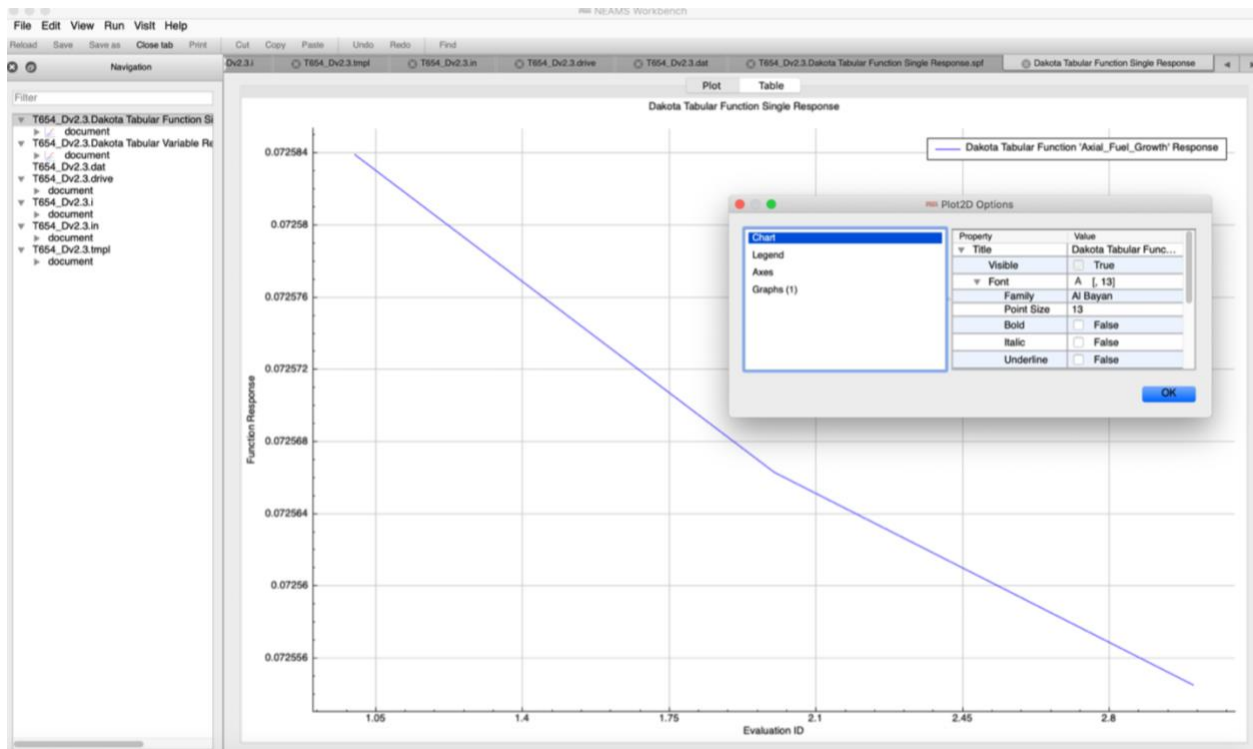


Figure A.12. Workbench plotter tool.

This concludes the BISON-Dakota Workbench integration tutorial.

REFERENCES

- A.1. K. M. Cunningham, R. A. Lefebvre, *NEAMS Workbench and Bison Fuel Performance Remote Application Configuration*, ORNL/TM-2019/1208, Oak Ridge National Laboratory, Oak Ridge, Tennessee, September 2019.
- A.2. *Dakota Documentation*, Sandia National Laboratories, Albuquerque, New Mexico, <https://dakota.sandia.gov/documentation.html> (accessed May 12, 2021).