ORNL/TM-2015/531

An Update on Improvements to NiCE Support for PROTEUS. (MS-15OR0401039)

Andrew Bennett
Alexander J. McCaskey
Jay Jay Billings

September 2015

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2015/531

Computer Science and Mathematics Division

An Update on Improvements to NiCE Support for PROTEUS. (MS-15OR0401039)

Andrew Bennett
Alexander J. McCaskey
Jay Jay Billings

Date Published: September 2015

Prepared by
OAK RIDGE NATIONAL LABORATORY
P.O. Box 2008
Oak Ridge, Tennessee 37831-6285
managed by
UT-Battelle, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

EXECUTIVE SUMMARY

The Department of Energy Office of Nuclear Energy's Nuclear Energy Advanced Modeling and Simulation (NEAMS) program has supported the development of the NEAMS Integrated Computational Environment (NiCE), a modeling and simulation workflow environment that provides services and plugins to facilitate tasks such as code execution, model input construction, visualization, and data analysis. This report details the developement of workflows for the reactor core neutronics application, PROTEUS. This advanced neutronics application (primarily developed at Argonne National Laboratory) aims to improve nuclear reactor design and analysis by providing an extensible and massively parallel, finite-element solver for current and advanced reactor fuel neutronics modeling. The integration of PROTEUS-specific tools into NiCE is intended to make the advanced capabilities that PROTEUS provides more accessible to the nuclear energy research and development community.

This report will detail the work done to improve existing PROTEUS workflow support in NiCE. We will demonstrate and discuss these improvements, including the development of flexible IO services, an improved interface for input generation, and the addition of advanced Fortran development tools natively in the platform.

1. Introduction

The NEAMS Integrated Computational Environment (NiCE) aims to enhance the usability of modeling and simulation codes and related software for both users and developers. Customized user interfaces and tools for input creation, job launching, and output analysis make end-to-end workflows a clear, well-defined set of steps. This process has been successfully applied to several high-profile codes, including the neutronics simulator developed at Argonne National Laboratory, PROTEUS.

The PROTEUS Core Neutronics application is primarily developed at Argonne National Laboratory (ANL) as part of the NEAMS Reactors Product Line. It is written in Fortran 90 with C preprocessor definitions, and uses an even-parity discrete ordinates approximation (SN2ND) to solve the steady-state neutron transport equation [2]. It runs on a variety of architectures, and scales from a couple cores on a laptop, to over 10^5 cores on a machine such as Mira at ANL. It relies heavily on PETSc matrices and vectors to acheive this parallelism and provide such scalability. Simulation capabilities include cross section generation, radiation transport, and fuel cycle modeling. This report details the efforts in increase support levels for PROTEUS within NiCE.

2. The Previous State of the PROTEUS Plugins

Prior to the work outlined in this document the PROTEUS plugins found in NiCE were oriented towards advanced users and developers of PROTEUS. During the model development phase of the PROTEUS workflow many options were populated with default values. In order for many of these options to be changed the user would have to know all of the valid settings allowable by PROTEUS.

3. IO Handlers for the INI File Format

In order to make the PROTEUS workflow more user friendly a new set of capabilities has been developed. Among these capabilities is a new IO service that allows for the reading and writing of generalized INI-formatted files. INI files are broken into sections, each containing some number of keys and associated values. Comments are usually supported via the # or ; characters. A typical INI file may look like the following:

```
# A typical INI file may look something like this
[Section1]
variable1 = value1
variable2 = value2

# INI files may have many sections.  This one has two
[Section2]
variable3 = value3
variable4 = value 4
```

These new INI services are configurable, making them highly reusable by other NiCE components that make use of the INI format. They are able to handle different comment characters, assignment operators, heading styles, indentation schemes, and are able to handle customized templates. The templating system allows for rapid development of intuitive plugins for codes that use INI formats. The INI reader can be optionally given a template file to use as a basis for building the resulting NiCE user interface (UI). Templates themselves follow an INI format. An example can be found below.

```
[Resolutions]
Refinement = 1.0 ; -100.0, 100.0 ; Continuous
Timestep = 1.0E-6 ; ; Undefined
Tolerance = 5.0E-6 ; ; Undefined

[Input/Output]
Input = input.mesh ; ; Undefined
Use_refinement = NO ; NO, YES ; Discrete
Output_format = ascii ; ascii, binary ; Discrete
```

Each line in a section of the template has four pieces. First and second are the variable name and the default value, separated by the '=' character. The next piece specifies applicable ranges. This piece can be left blank if there are no ranges or lists to select from. The final piece of a template entry is the type of UI element that NiCE will display. Undefined allows for free-form entry, continuous allows for a floating point entry within the range specified, and discrete allows for a selection amongst the list of options given.

Loading new INI files into NiCE with an accompanying template file will automatically build a custom-tailored UI into NiCE, complete with error checking and form validation. The resulting UI will be broken into sections according to the template file. If no template file is given NiCE will fall back to the default mode, with all entries being set to free form mode.

4. Improved PROTEUS Model Plugin

The PROTEUS model's improved UI makes it easy to quickly change settings. Advantages of the new UI include reduced risk of errors in input generation as well as improved discoverability of options. When a new PROTEUS model is created via the template all of the options are clearly labeled, with their respective input types drawn in an intuitive way. Those unfamiliar with advanced options within PROTEUS can easily see their availability, reducing the learning curve. The templating system also allows for new input files to be created or modified with less manual input, reducing the risk of typos.

5. PROTEUS Developer Support

In addition to the improvements made to the PROTEUS model plugin work has been done to increase the amount of support that PROTEUS developers have for working within NiCE. The Photran development environment for Eclipse has been integrated into NiCE, allowing for seamless development of PROTEUS, a Fortran based code. Photran [1] comes bundled as a part of the Eclipse Parallel Tools Platform (PTP), which provides a comprehensive suite of tools for developing parallel applications. This set of tools closes the gap for end-to-end project support for PROTEUS. Developers can now modify code, compile, generate input, launch jobs, and visualize output from NiCE.

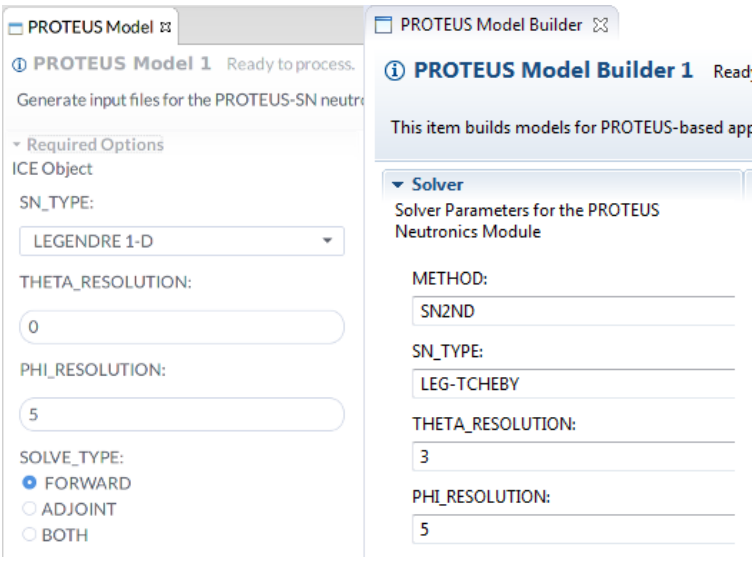

Fig. 1. The new PROTEUS model (left) makes options that have discrete values discoverable, while the old PROTEUS model was undocumented. Use of drop-down menus, radio buttons, and other forms of custom UI make it easier and less error prone to create new input.

6. Tutorial - Using PROTEUS with NiCE

This section is designed to outline the basic steps of setting up and using PROTEUS within NiCE. This guide assumes that you have PROTEUS installed on either a local or remotely accessible system. There are two different tasks for the input generation and launching of PROTEUS within NiCE:

- **PROTEUS Model Builder** - Generates a custom input file necessary to launch a PROTEU job.
- **PROTEUS Launcher** - Initiates the PROTEUS run on a local or remote system using the files generated from the *PROTEUS Model Builder*.

Follow the instructions at the Getting NiCE article on our wiki `https://wiki.eclipse.org/Getting_ICE` to download and install the latest version of NiCE on your system.

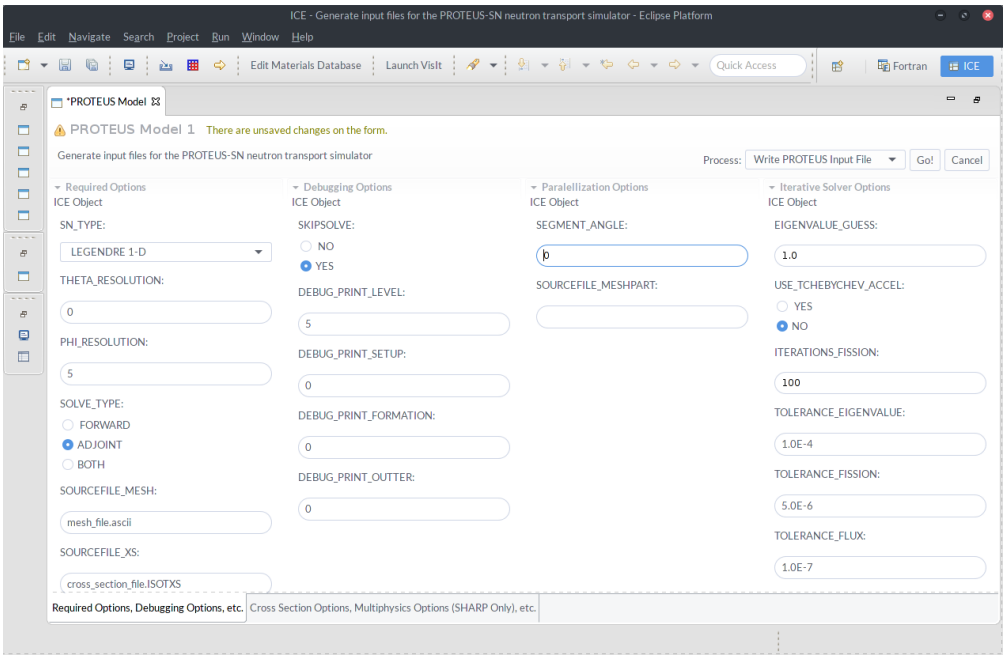

Fig. 2. The PROTEUS Model defines all of the information needed to run PROTEUS.

To begin with, create a new PROTEUS Model item. This can be accomplished within the NiCE perspective by clicking the 'green plus' button on the left hand panel. From the list select *PROTEUS Model*. The form that will populate in the center panel contains all of the information necessary to generate a PROTEUS input file. Customize the file with the parameters and filenames for your run, then on the top right of the form hit the *Go!* button besides *Write PROTEUS Input File*. This will create the input file in a format useable for PROTEUS.

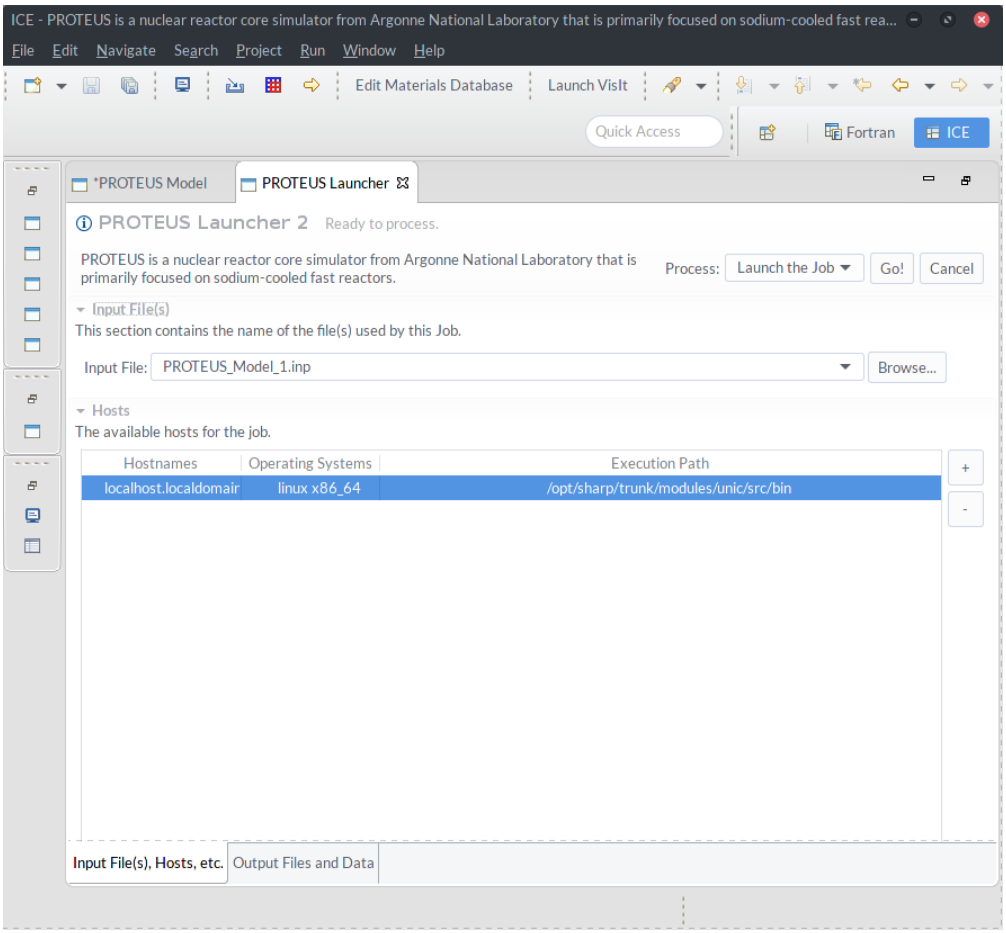

Fig. 3. A view of the PROTEUS Launcher.

Now, in a similar way that the PROTEUS Model was created, create a new PROTEUS Launcher. This item will allow the newly generated file to be passed to PROTEUS. Fill out the form so that the *PROTEUS_Model.inp* file is selected and the path to the PROTEUS binary is filled out. Then hit *Go!* to launch the job.

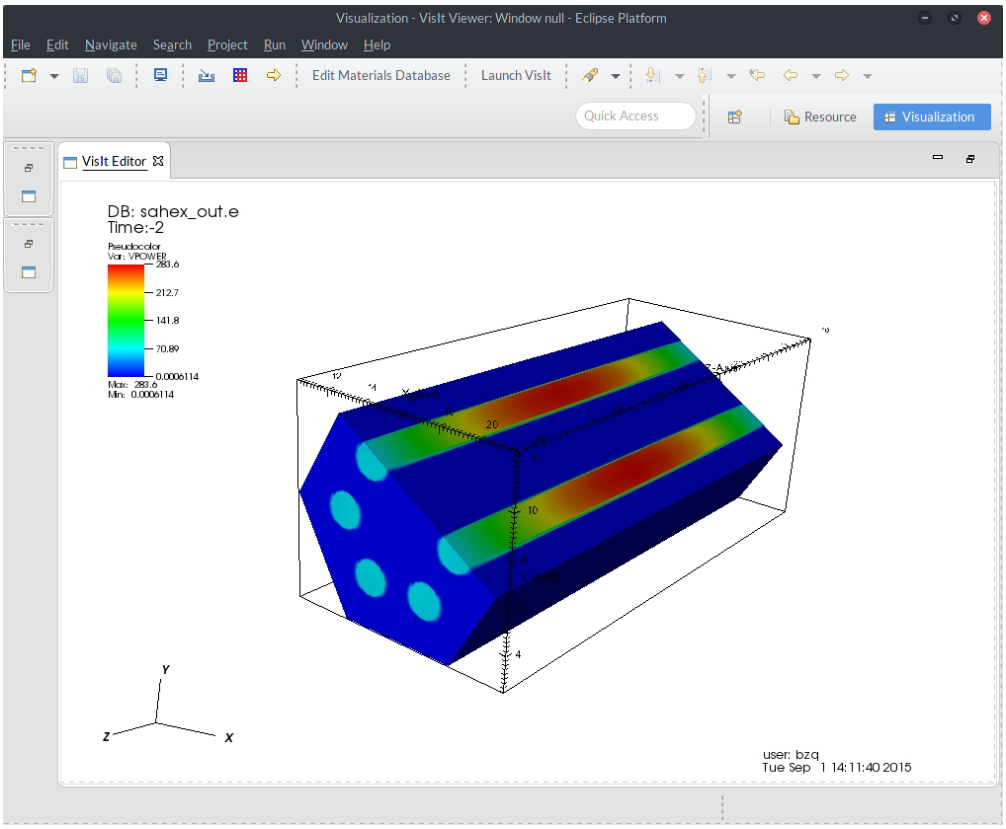

Fig. 4. A view of the Sahex sample PROTEUS output

When the job has finished the output will be transferred to a location that NiCE can access. To visualize this output you can use NiCE's native VisIt client. To use this feature you will need to have a copy of VisIt installed either locally or on a remotely accessible machine. Click the *Launch VisIt* button on the top bar in NiCE to begin. On the left side of the form will be two panels that are used to manipulate the output. Click the 'green plus' on the top panel to select a file to visualize. Once the file is open, highlight it in the list below and hit the 'green plus' on the lower panel to select the variables.

For more advanced visualization the VisIt Python interface can be used by clicking on the button next to the 'green plus' on the lower panel.

7. Acknowledgements

The NiCE team would like to acknowledge the PROTEUS team at Argonne National Laboratory, specifically Emily Shemon, who served as subject matter experts in making this work possible. Their input and advice greatly advanced this work, and will continue to do so in the future. Additionally, the NiCE team would like to acknowledge the funding agency for this work, the U.S. Department of Energy Office of Nuclear Energy's Nuclear Energy Advanced Modeling and Simulation (NEAMS) program and the Advanced Modeling and Simulation Office (AMSO) within DOE-NE.

REFERENCES

[1] Mariano Mendez. Photran 7.0 User's Guide. Sept 2011.

[2] E. R. Shemon, M. A. Smith, C. H. Lee, and A. (Nuclear Engineering Division) Marin-Lafleche. *PROTEUS-SN User Manual*. Aug 2014.