ORNL/TM-2014/699

An Update on Improvements to NiCE Support for RELAP-7 (MS-15OR040103)

Alexander J. McCaskey
Anna Wojtowicz
Jordan H. Deyton
Taylor Patterson
Jay Jay Billings

December 2014

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

ORNL/TM-2014/699

Computer Science and Mathematics Division

AN UPDATE ON IMPROVEMENTS TO NICE SUPPORT FOR RELAP-7

Alexander J. McCaskey
Anna Wojtowicz
Jordan H. Deyton
Taylor Patterson
Jay Jay Billings

Date Published: December 2014

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

1. Introduction

The Multiphysics Object-Oriented Simulation Environment (MOOSE) is a framework that facilitates the development of applications that rely on finite-element analysis to solve a coupled, nonlinear system of partial differential equations. RELAP-7 represents an update to the venerable RELAP-5 simulator that is built upon this framework and attempts to model the balance-of-plant concerns in a full nuclear plant.

This report details the continued support and integration of RELAP-7 and the NEAMS Integrated Computational Environment (NiCE). RELAP-7 is fully supported by the NiCE due to on-going work to tightly integrate NiCE with the MOOSE framework, and subsequently the applications built upon it. NiCE development throughout the first quarter of FY15 has focused on improvements, bug fixes, and feature additions to existing MOOSE-based application support. Specifically, this report will focus on improvements to the NiCE MOOSE Model Builder, the MOOSE application job launcher, and the 3D Nuclear Plant Viewer. This report also includes a comprehensive tutorial that guides RELAP-7 users through the basic NiCE workflow: from input generation and 3D Plant modeling, to massively parallel job launch and post-simulation data visualization.

2. Scope

During the first quarter of FY15, the NiCE development team focused on improving support for MOOSE-based applications, including RELAP-7. These changes were facilitated through NiCE's two MOOSE-specific plugins: the MOOSE Model Builder for generating input files, and the MOOSE Launcher for launching problems.

Some of these changes included routine bug fixes, as well as a significant expansion to the list of available features for RELAP-7 users. This includes usability enhancements, such as a new automatic YAML and action syntax generator, additional utilities for the 3D Plant Viewer, and a complete overhaul of the MOOSE Model Builder's Property Table (parameter editing tool) for a more intuitive and customizable model building workflow. Additionally, NiCE now includes a fully-customizable 2D CSV Plot Viewer which allows RELAP-7 PostProcessor data to be plotted and visualized "straight out of the box".

This document provides an update for NiCE's RELAP-7 users and is not intended as an introduction or overview of NiCE. This report assumes the reader is familiar with NiCE's MOOSE-specific plugins. To learn about NiCE and how to use its MOOSE-specific plugins, please visit our wiki at `https://wiki.eclipse.org/ICE`.

2.1 MOOSE Eclipse Perspective

NiCE supports numerous plugins from different areas of science, and as a result it can sometimes be difficult for new users to make sense of all the different windows, panels and tabs in the NiCE workbench. To address this issue for RELAP-7 and other MOOSE users, NiCE now includes a *MOOSE Perspective* which pares down UI components to only those that are necessary for MOOSE-based plugins. While it's not necessary to switch to the MOOSE Perspective to use MOOSE-based plugins in NiCE, it has been included for the convenience of users.

2.2 YAML and Action Syntax Generator

One of the most-requested features by MOOSE users this year was an automatic YAML and Action syntax generator. YAML and Action syntax files are necessary for any GUI-based MOOSE environment as a means to discern the appropriate rules of creating input for codes such as RELAP-7.

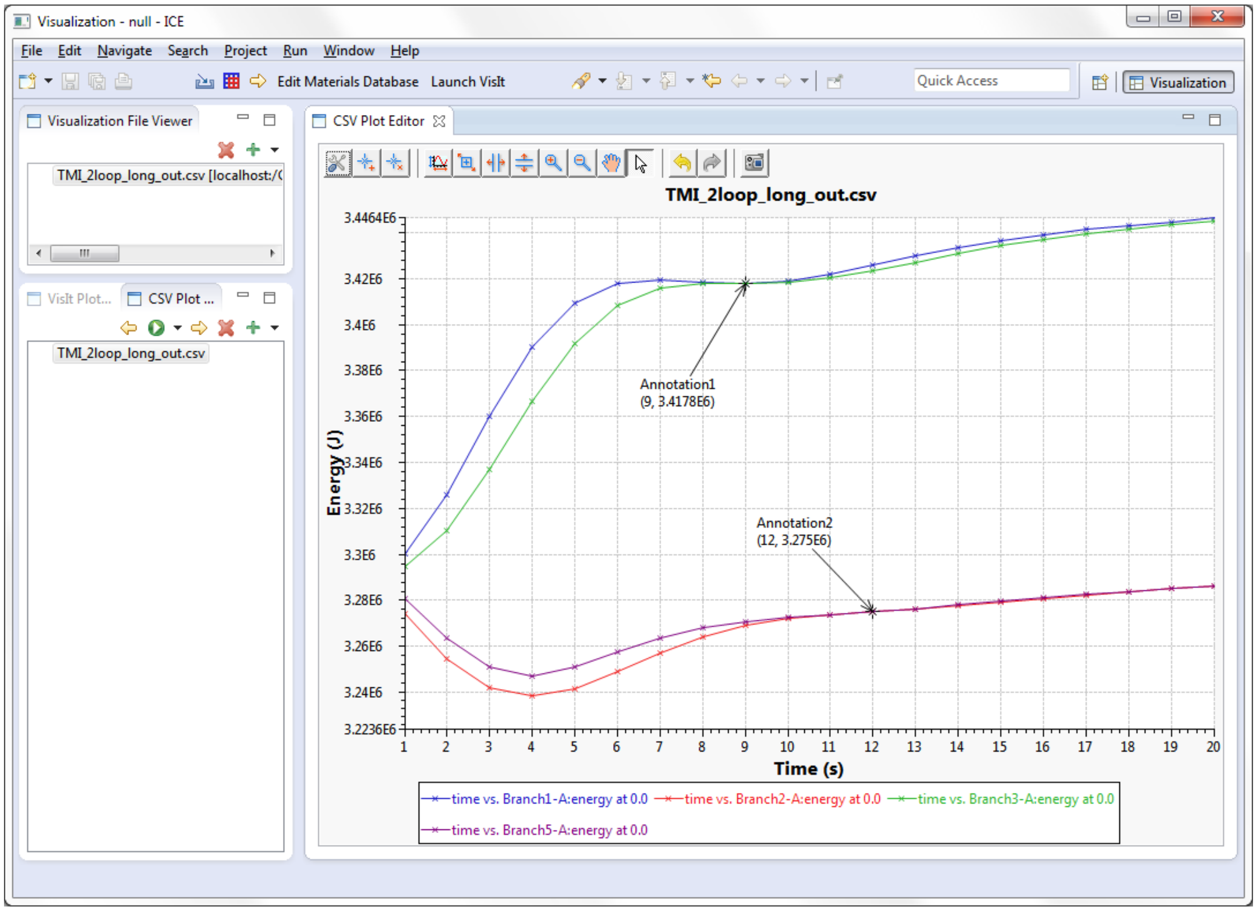
Previously, the NiCE team provided instructions on how to manually generate and modify these files for use with NiCE; however, since then, the team has implemented a utility which will do all of this automatically. All that is required of the user is to specify where their RELAP-7 installation is located (either locally or remotely), and any login credentials (if necessary for remote machines).

Running this utility will prompt NiCE to interact with RELAP-7 to produce YAML and action syntax files, download them to the user's local machine (if generated remotely), remove extraneous header/footer data, and place them in the appropriate directory in NiCE's local workspace. This feature is currently included as part of the MOOSE Launcher plugin, with plans to implement it as its own stand-alone utility in the near future.

2.3 CSV Plotting Support

Over the course of the summer, NiCE introduced a new 2D CSV plot viewer, capable of plotting PostProcessor data from RELAP-7 and other MOOSE-based codes. The CSV plot viewer features a myriad of high-level plotting capabilities, including multi-series plotting, time series plotting, and four plot types: line, scatter, bar and contour.

Fig. 1. The NiCE CSV Plot Viewer visualizing RELAP-7 PostProcessor data.


Visualization - null - ICE
File Edit Navigate Search Project Run Window Help
Edit Materials Database Launch VisIt
Quick Access
Visualization
Visualization File Viewer
TMI_2loop_long_out.csv [localhost:/(
VisIt Plot...
CSV Plot ...
TMI_2loop_long_out.csv
CSV Plot Editor
TMI_2loop_long_out.csv
Energy (J)
Time (s)
Annotation1
(9, 3.4178E6)
Annotation2
(12, 3.275E6)
time vs. Branch1-A:energy at 0.0
time vs. Branch2-A:energy at 0.0
time vs. Branch3-A:energy at 0.0
time vs. Branch5-A:energy at 0.0

The CSV Plot Editor pictured in Figure 1 includes many tools that enable the user to fully customize their plot. This list includes (but is not limited to) options to change the size, color, format, borders, line style and fonts of the graph labels, axes, grids, traces and more. Additionally, there are manual and auto-scale tools to change the perspective, options to add/remove custom annotations to data points, and a screenshot utility to save an image of the plot.

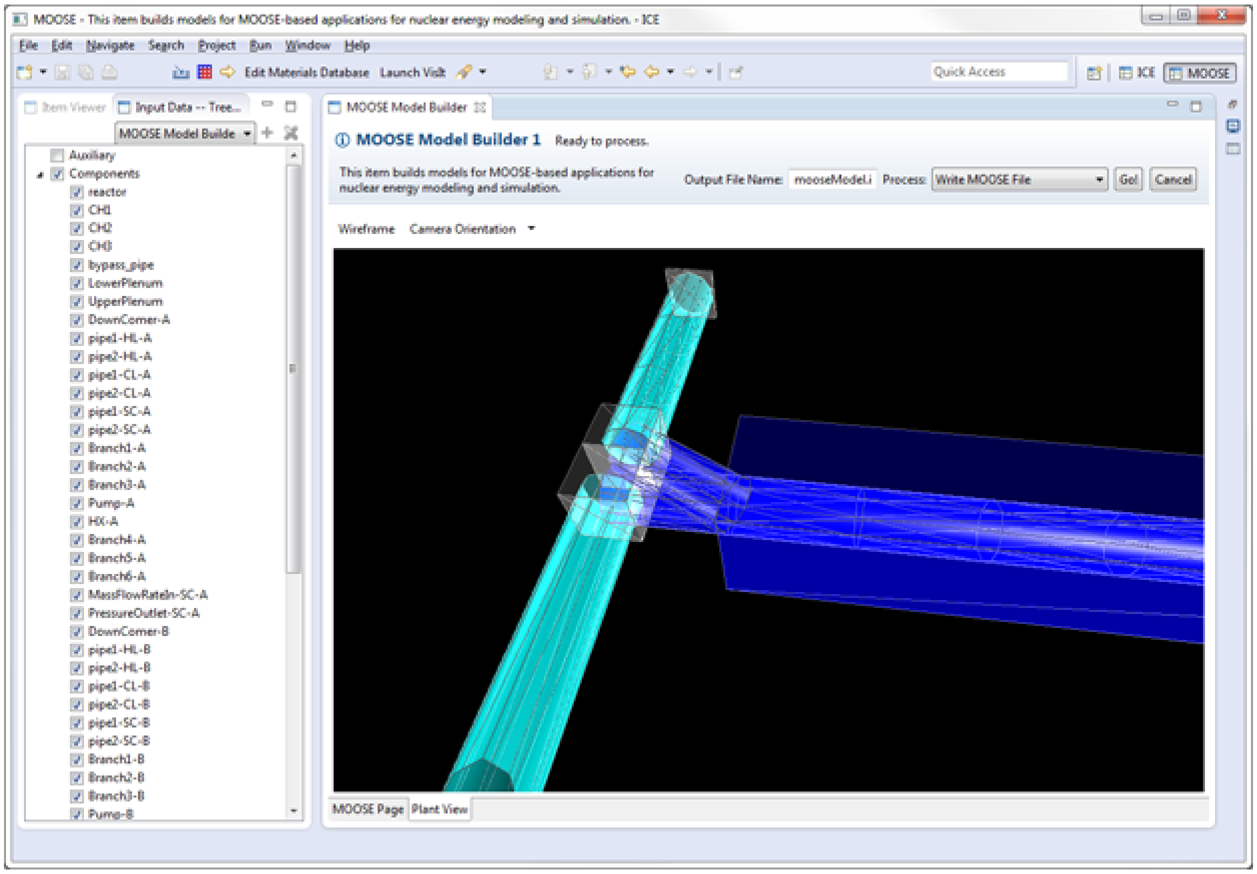
2.4 3D Plant View Enhancements

The MOOSE Model Builder features an interactive 3D plant view for RELAP-7 problems, which has been improved and refined since its initial release in NiCE in early 2014.

2.4.1 Wireframe Toggling

By clicking the *Wireframe* button in the Plant View's ToolBar, the user can now toggle wireframe mode for all rendered plant components. This lets you see how the meshes for the rendering engine are constructed (Figure 2). In certain cases, this may be useful for verifying the plant model before running a simulation. For instance, pipe components in RELAP-7 have a property “n_elems,” or the number of slices along the pipe’s length. By switching to wireframe mode, the user can see the same number of cylindrical sections that stacked together represent the pipe.

Fig. 2. The NiCE RELAP-7 Plant View shown in the new Wireframe mode.



MOOSE - This item builds models for MOOSE-based applications for nuclear energy modeling and simulation. - ICE

MOOSE Model Builder 1 Ready to process.

This item builds models for MOOSE-based applications for nuclear energy modeling and simulation.

Output File Name: mooseModel.i Process: Write MOOSE File

2.4.2 Camera Re-orientation

The MOOSE Model Builder’s Plant View now provides the ability to rotate the camera in any direction. To navigate the Plant View, use the W, S, A, D, space, and C keys to move forward, backward, left, right, up, and down, respectively. The viewing angle can now be rotated by left-clicking and dragging with the mouse, or by using the arrow keys. In addition to these controls, the camera can roll left or right by using the Q and E keys.

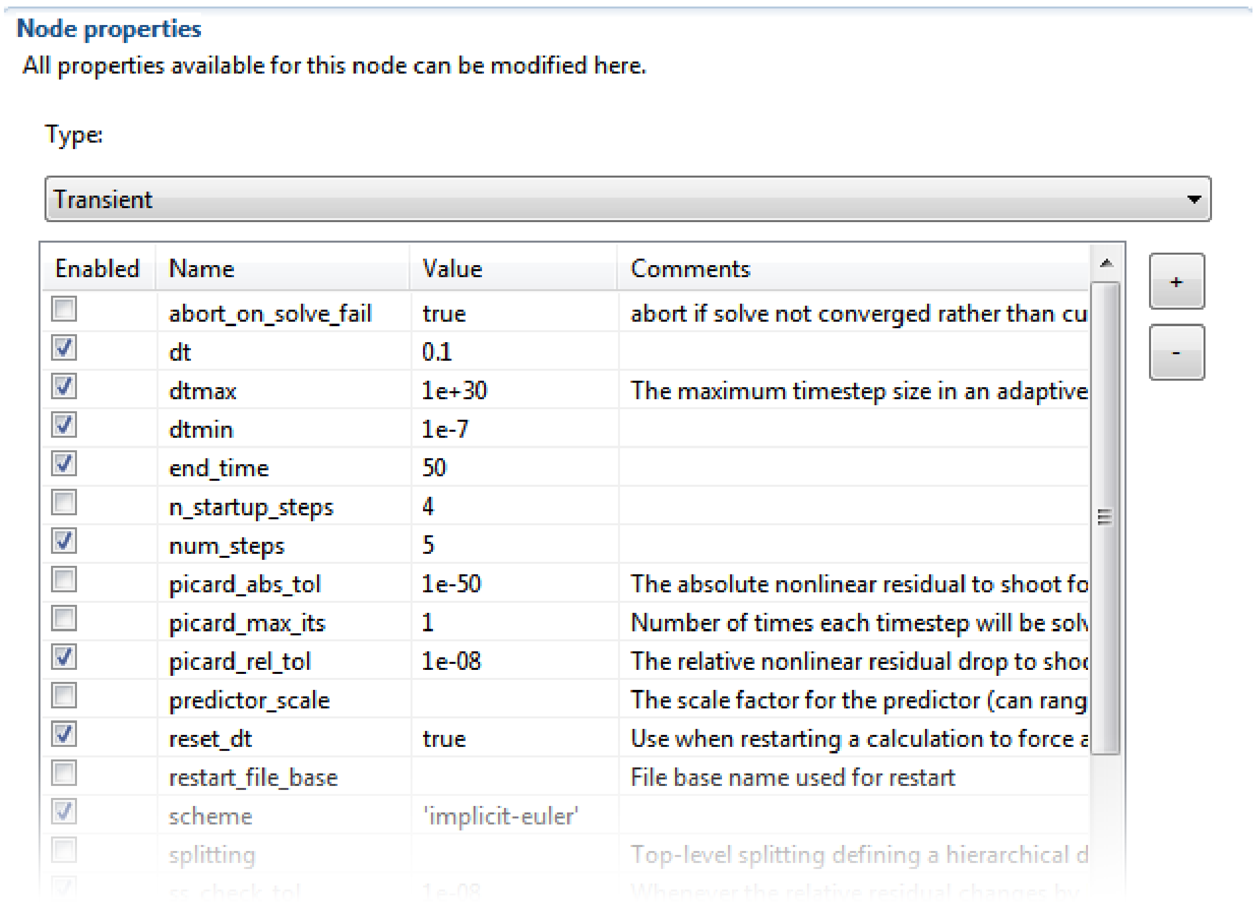
Additionally, the user can now reset the position of the Plant View’s camera by opening the *Camera Orientation* menu in the Plant View’s ToolBar and selecting *Reset to current default*. The initial default orientation “YZ” has the Y axis increasing to the right, the Z axis increasing upward, and the camera looking at the YZ-plane from the positive X axis. In the same *Camera Orientation* menu, we provide two alternative default orientations: the XY orientation shows the XY plane from the positive Z axis, and the ZX orientation shows the ZX plane from the positive Y axis.

2.5 Modifying MOOSE Block Parameters

The NiCE development team implemented a handful of enhancements related to parameter creation and modification, including a new parameter table, the ability to add in-line comments, comment out entire parameters/lines, and parameters with discrete sets of values. All of the features mentioned in the following section occur in the "Property View" tab of NiCE (Figure 3).

When working with a MOOSE Model Builder Item, selecting a block in the Tree View will prompt a list of associated parameters (if any) to appear in the Property View tab. This tab presents the parameters in a simple and intuitive table with four columns: Enabled, Name, Value and Comment.

Fig. 3. The RELAP-7 Parameter Table.



Node properties

All properties available for this node can be modified here.

Type:

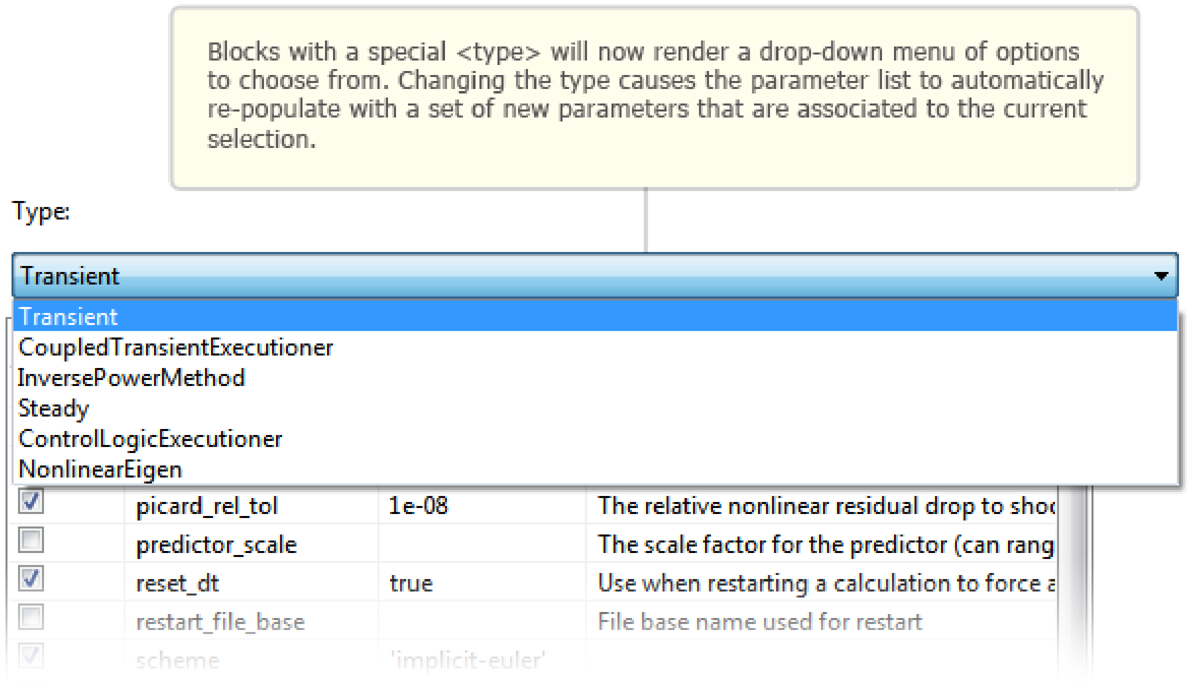
Transient

Enabled	Name	Value	Comments
☐	abort_on_solve_fail	true	abort if solve not converged rather than cu
✓	dt	0.1	
✓	dtmax	1e+30	The maximum timestep size in an adaptive
✓	dtmin	1e-7	
✓	end_time	50	
☐	n_startup_steps	4	
✓	num_steps	5	
☐	picard_abs_tol	1e-50	The absolute nonlinear residual to shoot fo
☐	picard_max_its	1	Number of times each timestep will be solv
✓	picard_rel_tol	1e-08	The relative nonlinear residual drop to shoo
☐	predictor_scale		The scale factor for the predictor (can rang
✓	reset_dt	true	Use when restarting a calculation to force a
☐	restart_file_base		File base name used for restart
✓	scheme	'implicit-euler'	
☐	splitting		Top-level splitting defining a hierarchical d
✓	ss_check_tol	1e-08	Whenever the relative residual changes by

2.5.1 *Type* Parameter Toggling

Some blocks such as the Executioner block have a special "type" property which determines the list of parameters associated to the block. NiCE will now automatically determine which blocks contain this special property, and expose it to the user as a drop-down menu above the parameter table. Changing the selection of this drop-down menu will cause the parameters list to change

Fig. 4. The RELAP-7 Parameter Type Selection Widget.



Blocks with a special <type> will now render a drop-down menu of options to choose from. Changing the type causes the parameter list to automatically re-populate with a set of new parameters that are associated to the current selection.

✓	picard_rel_tol	1e-08	The relative nonlinear residual drop to sho
☐	predictor_scale		The scale factor for the predictor (can rang
✓	reset_dt	true	Use when restarting a calculation to force a
☐	restart_file_base		File base name used for restart
✓	scheme	'implicit-euler'	

2.5.2 Enabling and Disabling Parameters

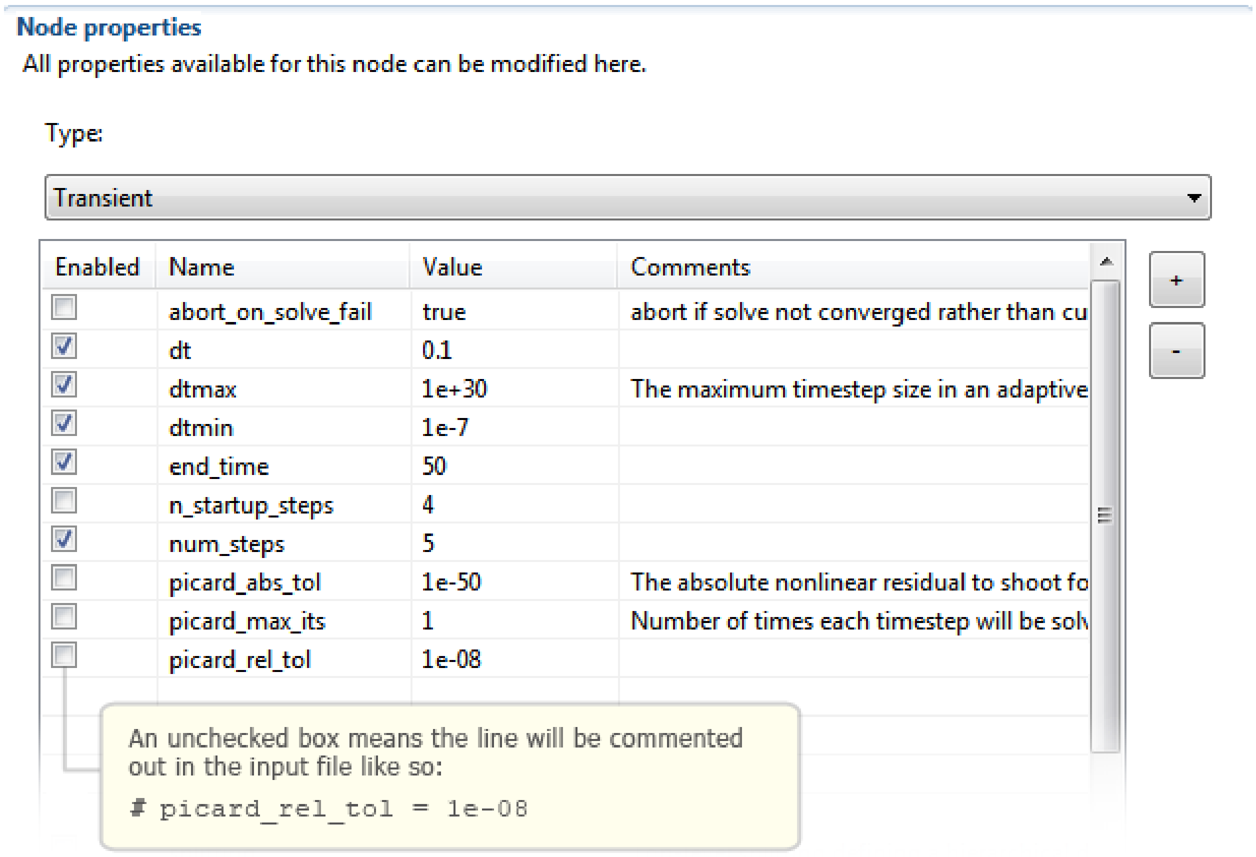
The "Enabled" column is a new feature which displays a checkbox next to each parameter row. A checked box means that the parameter in question will be written to file normally; an unchecked box means the parameter will still be written, however, the whole line will be commented out, and will thusly not be used during RELAP-7 execution. This is particularly useful when the user might want to keep track of parameters used in previous executions, without having to delete them from the current input file.

In addition to commenting-out parameters when an input file is created, NiCE also will correctly parse commented-out parameters when importing data from an already-existing file (insensitive to whitespaces). Previously, NiCE would skip over commented parameters when importing data. However, now all parameters are read in, and their appropriate states are reflected by the Enabled checkbox.

2.5.3 Adding In-line Comments

In addition to commenting-out entire parameters, NiCE also offers the user the option of adding an in-line comment at the end of a parameter via the "Comment" column of the Property View table.

Fig. 5. The RELAP-7 Parameter Table showing how to disable a parameter.



Node properties

All properties available for this node can be modified here.

Type: Transient

Enabled	Name	Value	Comments
☐	abort_on_solve_fail	true	abort if solve not converged rather than cu
✓	dt	0.1	
✓	dtmax	1e+30	The maximum timestep size in an adaptive
✓	dtmin	1e-7	
✓	end_time	50	
☐	n_startup_steps	4	
✓	num_steps	5	
☐	picard_abs_tol	1e-50	The absolute nonlinear residual to shoot fo
☐	picard_max_its	1	Number of times each timestep will be solv
☐	picard_rel_tol	1e-08	

An unchecked box means the line will be commented out in the input file like so:

```
# picard_rel_tol = 1e-08
```

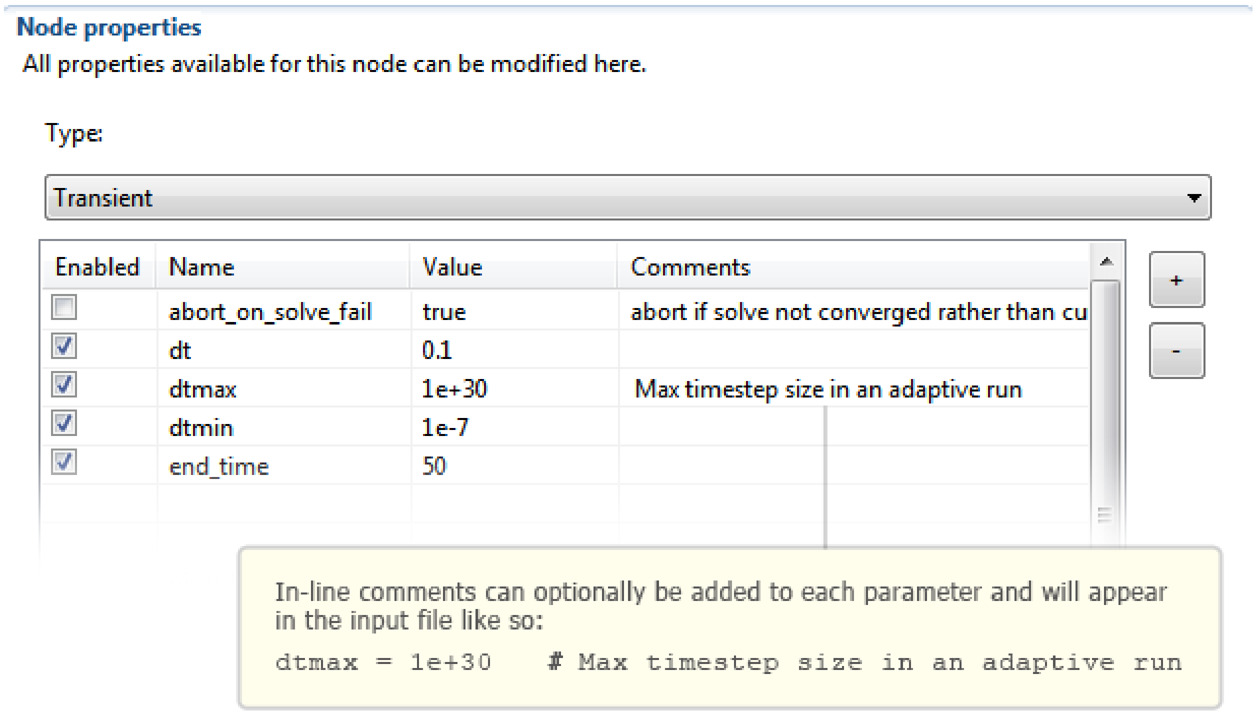
The resulting behavior is that the parameter is still used during RELAP-7 execution, however, offers the user a convenient place store a note on the parameter if they chose to, such as units, a list of changes, or a short explanation of what the parameter does.

Similar to how NiCE will parse commented-out lines when importing an already-existing file, NiCE will also parse in-line comments in a similar fashion (whitespace insensitive), reflecting them in the “Comments” column.

2.5.4 Parameters with Discrete Values

The last new feature in NiCE related to parameter modification is the ability for some parameters to have discrete sets of values. By default, parameters in the Property View have a text box where the user can enter any value they wish to assign. However, some parameters in RELAP-7 and other MOOSE codes are intended to only have a specific value chosen from a discrete set of options. For example, a Pipe component in RELAP-7 can have a parameter named “controlled” whose value should be only one of 11 pre-determined values. To accommodate this convention, NiCE now determines at load time which parameters have discrete sets of values, and renders a drop-down menu in place of where the “value”

Fig. 6. The RELAP-7 Parameter Table showing how to add or view a comment.


Node properties

All properties available for this node can be modified here.

Type: Transient

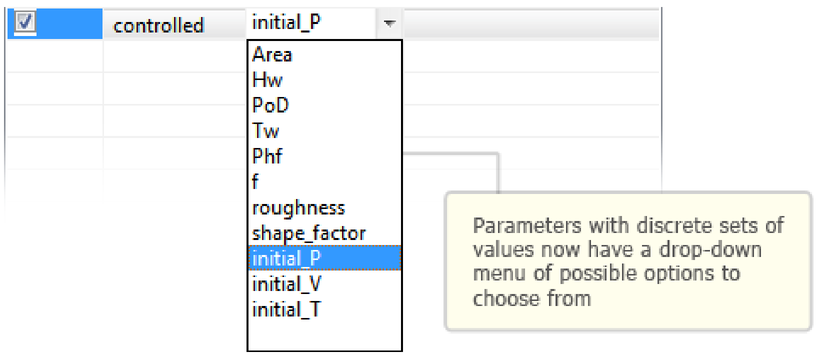
Enabled	Name	Value	Comments
☐	abort_on_solve_fail	true	abort if solve not converged rather than cu
✓	dt	0.1	
✓	dtmax	1e+30	Max timestep size in an adaptive run
✓	dtmin	1e-7	
✓	end_time	50	

In-line comments can optionally be added to each parameter and will appear in the input file like so:

```
dtmax = 1e+30    # Max timestep size in an adaptive run
```

textbox is normally located. This prevents users from attempting to set invalid values which could potentially halt the problem execution later on.

Fig. 7. The RELAP-7 Parameter Table showing MOOSE-defined discrete parameters..



Parameters with discrete sets of values now have a drop-down menu of possible options to choose from

2.6 Comprehensive List of Bug Fixes

A myriad of bugs in NiCE's MOOSE plugins were fixed since the last RELAP-7 report, however, they are not directly addressed here as their technical and specific nature are beyond the scope of this report. For completeness though, we will now list all the bugs fixed, and provide a link for each that gives a more detailed description of the bug.

1. 2014-09-19 - *MOOSE Model Builder only writing out "required" parameters*: The MOOSE Model Builder only wrote certain parameters marked as "required" and skipped other parameters, including custom parameters. Now, non-required parameters with valid and non-empty values will also be written to the input file.

   https://sourceforge.net/p/niceproject/moosebugs/9/

2. 2014-09-22 - *Unchecking parent blocks should uncheck all child blocks too*: Unchecking blocks in the MOOSE tree did not uncheck the child blocks; this has been fixed. Unchecking a parent block unchecks all of its child blocks. Checking a parent block does not check any of its child blocks.

   http://sourceforge.net/p/niceproject/moosebugs/3/

3. 2014-09-22 - *Remove BlankBlocks at root level*: Previously, you could add BlankBlocks to the root level of the MOOSE tree. The ability to add blocks to the root level has now been removed.

   http://sourceforge.net/p/niceproject/moosebugs/4/

4. 2014-09-22 - *Enable parameter deletion for all blocks/parameters*: The user was not allowed to delete all parameters. Now, users may delete any parameter.

   http://sourceforge.net/p/niceproject/moosebugs/5/

5. 2014-09-22 - *Disable renaming of root nodes*: Any block, including the root level blocks, could be renamed. This has been changed so that root level blocks cannot be renamed.

   http://sourceforge.net/p/niceproject/moosebugs/11/

6. 2014-09-22 - *Adding parameters to BlankBlocks does not work*: Parameters could not be added to BlankBlocks. This bug has been fixed.

   http://sourceforge.net/p/niceproject/moosebugs/12/

7. 2014-09-22 - *mpirun not working for parallel execution*: The command issued for parallel execution has been changed from mpirun to mpiexec.

   https://sourceforge.net/p/niceproject/moosebugs/14/

8. 2014-09-23 - *Not all nodes in tree have an active DataComponent*: A block's parameters are stored in a NiCE DataComponent. However, to display them in the user interface (UI) requires the DataComponent to be marked as active. A fallback to display them was already available in the UI code, but the underlying DataComponents are now automatically marked as active.

   https://sourceforge.net/p/niceproject/moosebugs/10/

9. 2014-10-08 - *YAML reading should skip header/footer lines*: Reading action syntax files required manual removal of header and footer lines before use in NiCE. The file parsers in NiCE have been updated to remove these automatically.

https://sourceforge.net/p/niceproject/moosebugs/2/

10. 2014-10-27 - *Persisted MOOSE Launcher has incorrect title*: When persisted, a MOOSE Launcher would have an incorrect title, e.g., instead of "MOOSE Launcher 1", the title would be "MOOSE Launcher Launcher". This has been corrected.

    https://sourceforge.net/p/niceproject/moosebugs/21/

11. 2014-10-27 - *RELAP-7 and MARMOT take too long to load in Model Builder*: Performance while loading RELAP-7 and MARMOT trees in the MOOSE Model Builder was poor. This was due to logic added to handle custom top-level blocks and conversion from old block naming conventions to newer ones (ex. "Output" to "Outputs"). This has been fixed, and performance for loading these two MOOSE applications in NiCE has improved.

    https://sourceforge.net/p/niceproject/moosebugs/18/

12. 2014-10-27 - *MOOSE Tree doesn't clear when no MOOSE Model Builder is active*: When all MOOSE Model Builders were closed, the supported tree view would not clear. This would cause null pointer exceptions. This has been fixed.

    https://sourceforge.net/p/niceproject/moosebugs/20/

13. 2014-11-03 - *Saving Form or setting Input file resets File entries*: When updating and saving the MOOSE Launcher, the other selections in the MOOSE Launcher (mesh, power history, etc.) would reset. This has been fixed.

    https://sourceforge.net/p/niceproject/moosebugs/25/

14. 2014-11-07 - *Select parent node(s) when child node is selected*: When the child of an unchecked block is checked, all of the child block's ancestor blocks should also be checked. This bug has been fixed.

    https://sourceforge.net/p/niceproject/moosebugs/27/

15. 2014-11-07 - *YAML/action syntax launchJob.sh not working locally on Mac*: Executing the launchJob.sh script locally via NiCE would fail with bash errors, while running it from the command line was error free. This bug was due to escape characters invalid for running on OS X. This bug has been fixed.

    https://sourceforge.net/p/niceproject/moosebugs/26/

16. 2014-11-10 - *Simultaneous 3D views not supported*: Previously, opening multiple 3D views in NiCE would lead to exceptions, ultimately preventing all 3D views from working until NiCE is restarted. This included the RELAP-7 3D plant view available in the MOOSE Model Builder. Support for multiple 3D views has since been integrated into NiCE.

    https://sourceforge.net/p/niceproject/development-backlog/345/

17. 2014-11-10 - *Plant view needs to be rotatable or have alternate axes specified*: Previously, the view could not be rotated around the x- or z-axis. Instead, the view was locked so that the y-axis was always up. This meant the RELAP-7 plants, which sit in the yz-plane, were often sideways. The camera is now fully rotatable using the Q and E keys.

    https://sourceforge.net/p/niceproject/bugs/277/

18. 2014-12-05 - *Plant View (RELAP-7) doesn't update when Components are unselected*: When unchecking plant components in a RELAP-7 tree, the unchecked components would still be rendered in the MOOSE Model Builder?s 3D plant view. Unchecking and checking the components now removes and adds the plant components from the plant view.

    https://sourceforge.net/p/niceproject/moosebugs/30/

19. 2014-12-09 - *NullPointerException when rendering EntryComposite*: When trying to use the MOOSE Launcher with no input files in the ICE default directory, the launcher would crash. This was due to a UI bug and has since been fixed.

    https://bugs.eclipse.org/bugs/show_bug.cgi?id=454640

3. Future Work

There are a number of improvements that can be made to the 3D plant view, including descriptive textures for the various types of plant components, tips for plant components when the user hovers a mouse over them, rotating 3D axes in the foreground, and camera controls for rotation around a pivot point. In addition to these features, based on user experience, the 3D plant view could also be extended for data visualization: for instance, simulation output could be integrated into the 3D graphics for pipes, core channels, pumps, and other plant components to help the user investigate variations in temperature or flow across the simulated plant.

4. Availability

All of the capabilities described herein are available for free at the NiCE website, http://niceproject.sourceforge.net, as binary releases and source code. Detailed tutorials and documentation on how to use NiCE, including the tutorials in this document, are also available at that page.

The capabilities in this document are brand new and while we assert that they work, that may not be the case on all systems for all users. Problems should be reported directly to the NiCE team by sending an email to the project list, ice-dev <at> eclipse.org, or following the instructions to report bugs at https://bugs.eclipse.org/bugs/buglist.cgi?component=Core&product=Ice&resolution=---.

5. Tutorial - Using RELAP-7 with NiCE

The capabilities in this document are brand new and while we assert that they work, that may not be the case on all systems for all users. Problems should be reported directly to the NiCE team by sending an email to the project list, `ice-dev <at> eclipse.org`, or following the instructions to report bugs on our Bugzilla.

5.1 Introduction

This section is designed to outline the basic steps of setting up and using the MOOSE plug-ins in NiCE. NiCE currently supports four MOOSE-based applications: MARMOT, BISON, RELAP-7 and RAVEN. There are two different tasks for the input generation and launching of MOOSE products within NiCE:

- **MOOSE Model Builder** - Generates a custom input file necessary to launch a MARMOT, BISON, RELAP-7 or RAVEN job.
- **MOOSE Launcher** - Initiates the MARMOT, BISON, RELAP-7 or RAVEN codes to run on a local or remote system using the files generated from the *MOOSE Model Builder*. Also includes a utility for generating YAML and Action syntax files necessary to use the *MOOSE Model Builder*.

5.2 Installation and Configuration

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.

5.2.1 Prerequisites

You should have the MOOSE environment installed, either your local or remote machine. Instructions for installing MOOSE can be found at `http://mooseframework.org/getting-started/`. Additionally, you should have MARMOT, BISON, RELAP-7 or RAVEN compiled and ready to run. Contact the development teams of these projects on the rules and regulations for obtaining these codes.

5.2.2 MOOSE Perspective

NiCE supports numerous plugins from different areas of science, and as a result it can sometimes be difficult for new users to make sense of all the different windows, panels and tabs in the workbench. To address this issue for MOOSE users, NiCE now includes a *MOOSE Perspective* which pares down UI components to only those that are necessary for MOOSE-based plugins.

To access the *MOOSE Perspective*, use the the NiCE toolbar at the top and navigate to: *Window* > *Open Perspective* > *Other...* Select *MOOSE* in the window that pops up and click *OK*. Alternatively, you can also access the same pop-up menu by clicking the *Open Perspective* button in the upper right-hand corner of the NiCE workbench. Once the MOOSE Perspective opens, you should notice the workbench now contains fewer UI components, and should resemble something like this: While it's not necessary to switch to the MOOSE Perspective to use MOOSE-based plugins, it has been included for user convenience.

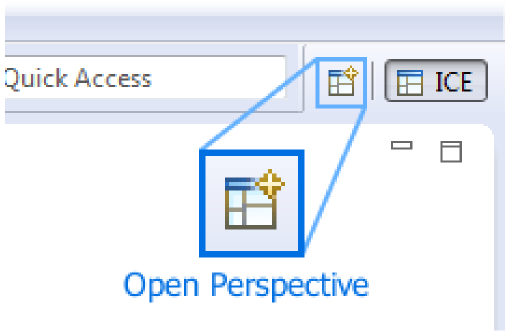
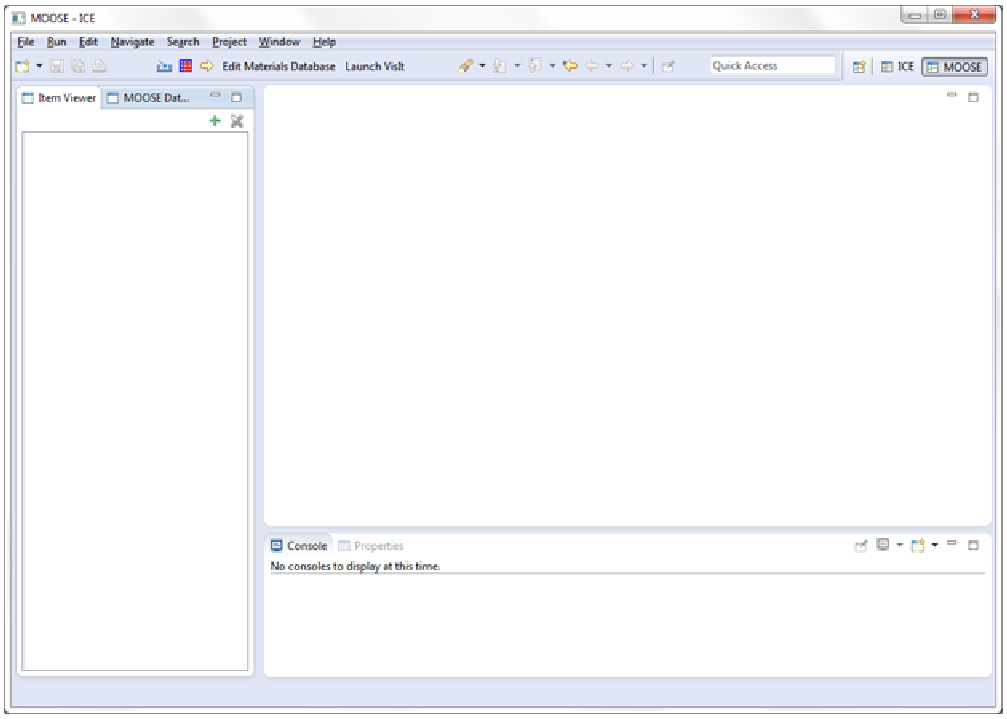


Fig. 8. The NiCE Open Perspective button will allow you to switch to the MOOSE Perspective.


MOOSE - ICE
File Run Edit Navigate Search Project Window Help
Edit Materials Database Launch VisIt
Quick Access
ICE
MOOSE
Item Viewer
MOOSE Dat...
Console
Properties

No consoles to display at this time.

Fig. 9. A view of the NiCE MOOSE Perspective.

5.3 Generating YAML and Action Syntax Files

NiCE relies on generated YAML and Action syntax files to specify the rules of creating MOOSE-based input files. To simplify this task, NiCE includes a utility that will generate these files (either locally or remotely), clean them up, and place them in the appropriate local directory for NiCE to use. All the user has to specify is the hostname of the machine where the MOOSE applications are installed, and the path to those MOOSE applications. This task must be completed at least once for every installation of NiCE before the *MOOSE Model Builder* can be used. It's also a good idea to periodically re-run this utility to keep NiCE's YAML and Action syntax files in sync with any updates to your MOOSE-based codes.

To use this utility, begin by clicking the green "+" button in the *Item Viewer*, located on the left-hand side of the NiCE workbench. This will prompt a window to pop up; select the *MOOSE Launcher* Item and

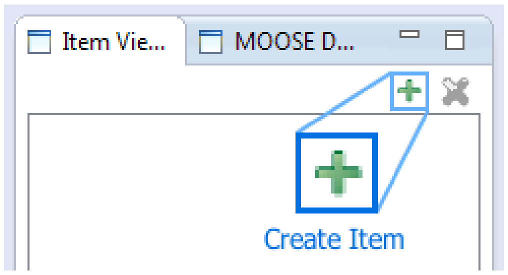
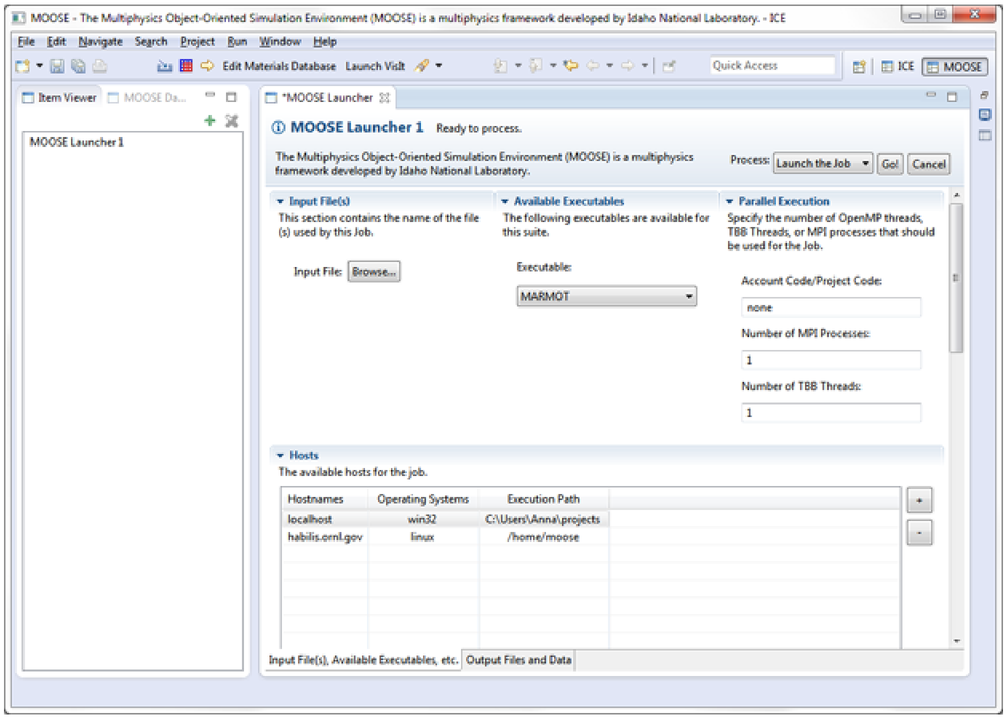


Fig. 10. Creating a new MOOSE Launcher is simple: just click the green *Create Item* button.

click *Finish*. The *MOOSE Launcher* will load in the main workbench area (Figure 11). From the list of


MOOSE - The Multiphysics Object-Oriented Simulation Environment (MOOSE) is a multiphysics framework developed by Idaho National Laboratory. - ICE
File Edit Navigate Search Project Run Window Help
Edit Materials Database Launch VisIt
Quick Access
ICE
MOOSE
Item Viewer
MOOSE Da...
MOOSE Launcher 1
*MOOSE Launcher
Input File(s), Available Executables, etc.
Output Files and Data

MOOSE Launcher 1 Ready to process.

The Multiphysics Object-Oriented Simulation Environment (MOOSE) is a multiphysics framework developed by Idaho National Laboratory.

Process: Launch the Job Go! Cancel

Input File(s)
This section contains the name of the file (s) used by this Job.
Input File: Browse...

Available Executables
The following executables are available for this suite.
Executable: MARMOT

Parallel Execution
Specify the number of OpenMP threads, TBB Threads, or MPI processes that should be used for the Job.
Account Code/Project Code: none
Number of MPI Processes: 1
Number of TBB Threads: 1

Hosts
The available hosts for the job.

Hostnames	Operating Systems	Execution Path
localhost	win32	C:\Users\Anna\projects
habilis.ornl.gov	linux	/home/moose

Fig. 11. The MOOSE Launcher Item view.

Available Executables, select “Generate YAML/Action syntax files”. Next, specify where your

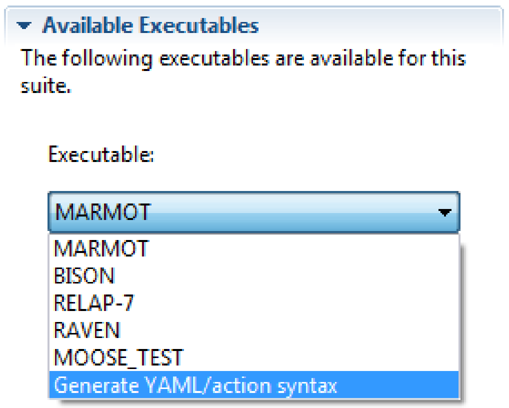
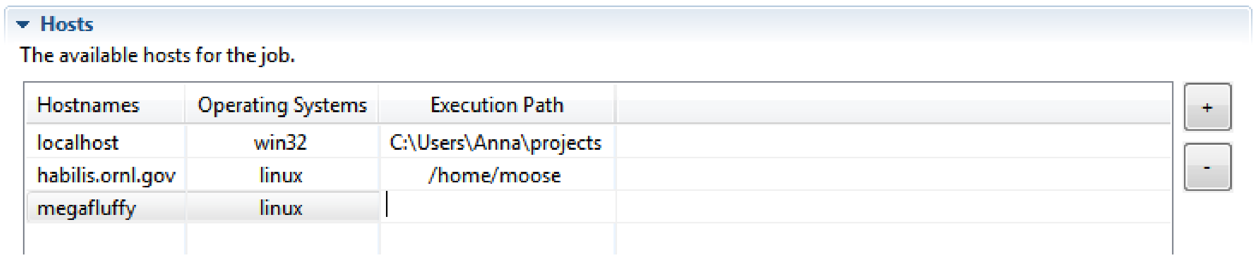

Available Executables
The following executables are available for this suite.
Executable:
MARMOT
MARMOT
BISON
RELAP-7
RAVEN
MOOSE_TEST
Generate YAML/action syntax

Fig. 12. To generate YAML and Action Syntax files, select the appropriate Executable from the Available Executables drop-down menu.

MOOSE-based codes are installed, either locally or remotely. A list of hosts used at ORNL is displayed by default, however, additional hosts can be added by clicking the “+" button to the right of the *Hosts* table.



Hostnames	Operating Systems	Execution Path
localhost	win32	C:\Users\Anna\projects
habilis.ornl.gov	linux	/home/moose
megafluffy	linux	

Fig. 13. The NiCE Hosts Table, where users can specify the local or remote machine to launch a MOOSE application or generate YAML/Action syntax files.

When adding hosts, set the *Execution Path* to the trunk directory of the machine’s MOOSE installation. For example, if I have BISON on a machine with the following structure:

```
/home/user/trunk/bison/bison-opt
```

then the execution path in NiCE should be set to:

```
/home/user/trunk
```

If you are launching on a remote machine, be sure that you have appropriate privileges for the execution path. Once you are finished, save your settings by clicking the floppy-disk icon in the upper-left hand corner of the NiCE workbench (or *Ctrl+S*) as shown in Figure 14. If you make any subsequent changes to the *MOOSE Launcher* form, you will have to re-apply them by saving the form in the same way.

Finally, use the *Process* menu in the upper right-hand corner; select the *Launch the Job* task from the drop-down menu and click the *Go!* button (see Figure 15). This step will launch a script on your designated machine that checks which MOOSE-based codes you have installed, generates the appropriate

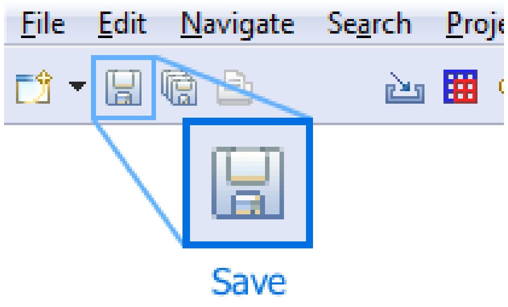


Fig. 14. The NiCE Save button

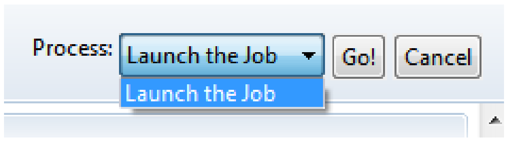


Fig. 15. The NiCE Process drop-down.

files, removes any extraneous header/footer text, and places them in your *$ HOME/ICEFiles/ default/MOOSE* directory.

5.4 Creating Input

To create an input file for a MOOSE problem, there are two ways you can get started: creating an input file from scratch, or importing an already existing file to modify.

5.4.1 Creating an Item

If you'd like to start from scratch, begin by clicking the green "+" button in the *Item Viewer*, located on the left-hand side of the NiCE workbench (Figure 10). This will prompt a window to pop up; select the *MOOSE Model Builder* Item, and click *Finish*. Alternatively, if you'd like to import an already existing `*.i` file to modify, click the yellow item import arrow located at the top of the NiCE workbench. A wizard will pop up prompting you to specify two things: the `*.i` file you'd like to import from your filesystem, and what kind of *Item* you'd like to import it into (Figure 17). Use the *Browse...* button to select your input file, and set the item type to *MOOSE Model Builder*. Click *Finish* when you're done and NiCE will import the file data.

Regardless of how you decide to begin creating your input file, once the *MOOSE Model Builder* loads, you will see a workbench that looks like Figure 18. Make note of the two tabs highlighted: the *Tree View* and *Properties* tabs, as we'll be using them quite a bit in the following section. Note that all the panels, tabs and windows in NiCE can be re-sized and moved around for the configuration that best suits your needs.

5.4.2 Constructing the Tree

Before beginning the construction of a problem, we must first specify which MOOSE application that will be launched with this input problem. To do this, use the drop-down menu on the *Tree View* and select

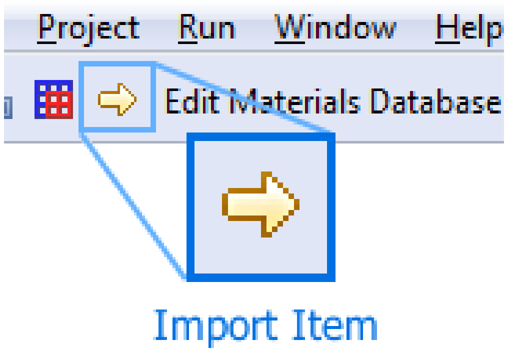
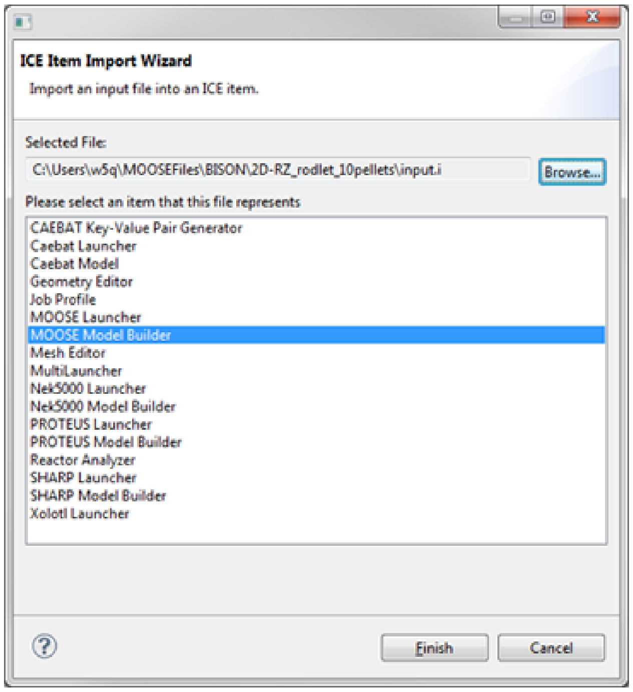


Fig. 16. The NiCE Toolbar button for importing an existing Item.


ICE Item Import Wizard

Import an input file into an ICE item.

Selected File:

C:\Users\w5q\MOOSEFiles\BISON\2D-RZ_rodlet_10pellets\input.i

Please select an item that this file represents

CAEBAT Key-Value Pair Generator
Caebat Launcher
Caebat Model
Geometry Editor
Job Profile
MOOSE Launcher
MOOSE Model Builder
Mesh Editor
MultiLauncher
Nek5000 Launcher
Nek5000 Model Builder
PROTEUS Launcher
PROTEUS Model Builder
Reactor Analyzer
SHARP Launcher
SHARP Model Builder
Xolotl Launcher

Fig. 17. The NiCE Import Existing Item Wizard.

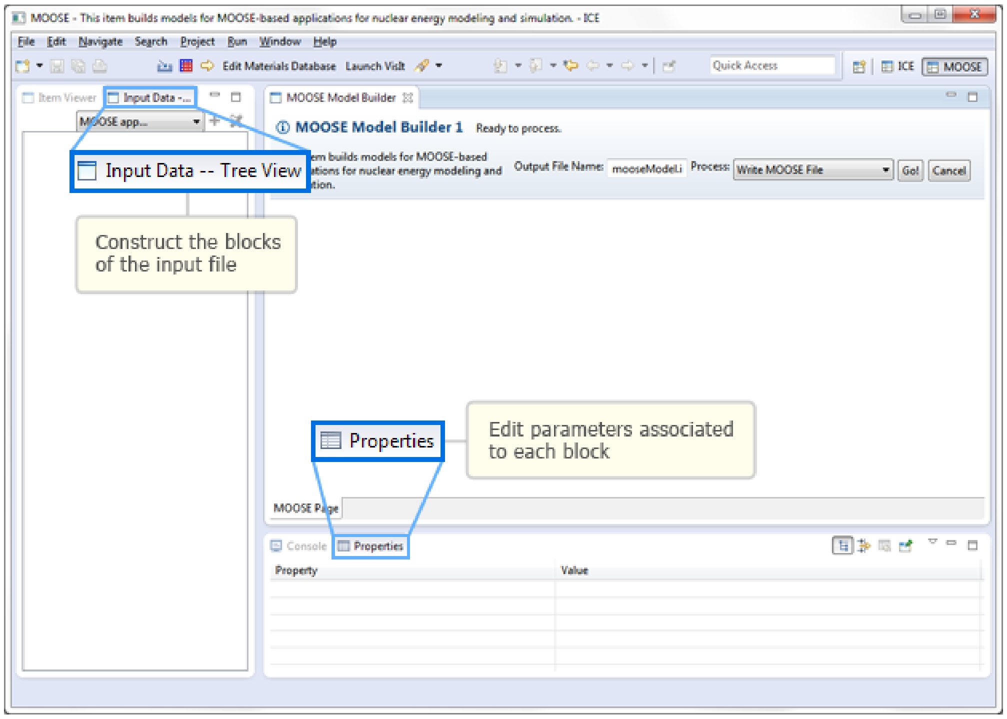


Fig. 18. The NiCE MOOSE Model Builder Item for creating MOOSE block-based input files.

one of the options.

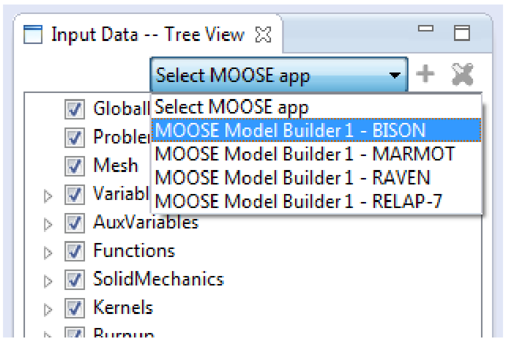
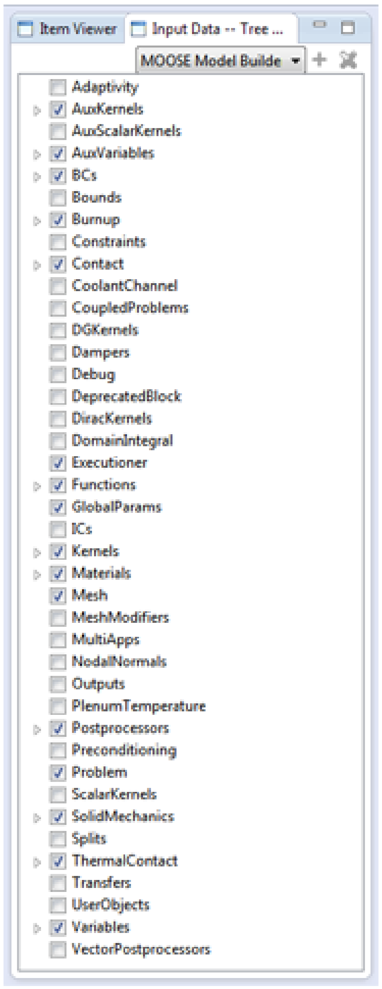


Fig. 19. The NiCE MOOSE App drop-down menu.

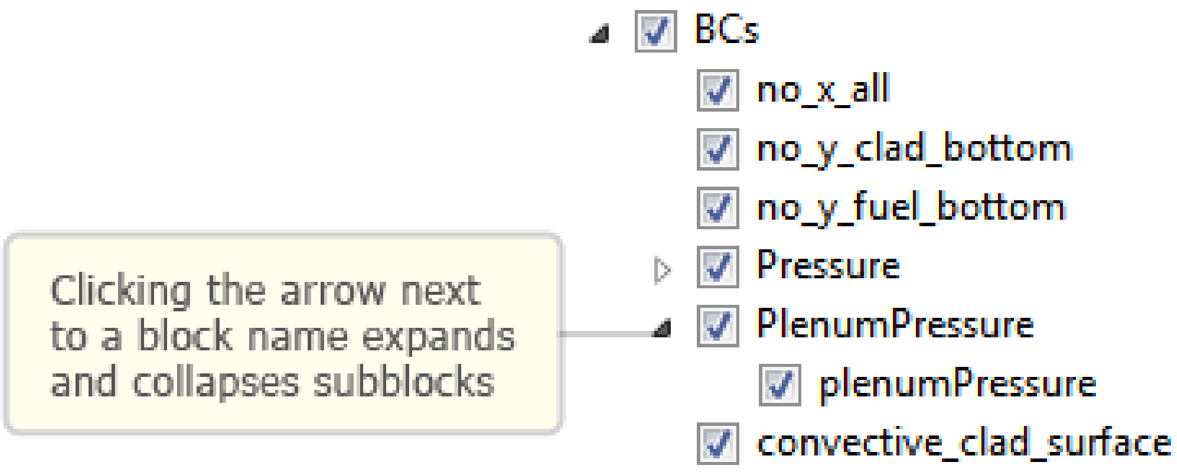
At this point, a fully loaded set of blocks should appear in the *Tree View*. If you imported your data from an already existing file, you'll notice the data has been loaded into any blocks that are checked off. If you started from scratch, none of the blocks will be checked off yet. We're now ready to begin using the *Tree View* to add, delete and modify blocks.

5.4.2.1 Expanding and Collapsing Subblocks If you imported data from an existing input file, you will likely notice some block names with a small arrow next to them. This indicates that the block contains subblock structures beneath it. To access these subblocks, simply click the small arrow and the subblocks will expand. Clicking this arrow again will collapse the subblocks.



- ☐ Adaptivity
- ✓ AuxKernels
- ☐ AuxScalarKernels
- ✓ AuxVariables
- ✓ BCs
- ☐ Bounds
- ✓ Burnup
- ☐ Constraints
- ✓ Contact
- ☐ CoolantChannel
- ☐ CoupledProblems
- ☐ DGKernels
- ☐ Dampers
- ☐ Debug
- ☐ DeprecatedBlock
- ☐ DiracKernels
- ☐ DomainIntegral
- ✓ Executioner
- ✓ Functions
- ✓ GlobalParams
- ☐ ICs
- ✓ Kernels
- ✓ Materials
- ✓ Mesh
- ☐ MeshModifiers
- ☐ MultiApps
- ☐ NodalNormals
- ☐ Outputs
- ☐ PlenumTemperature
- ✓ Postprocessors
- ☐ Preconditioning
- ✓ Problem
- ☐ ScalarKernels
- ✓ SolidMechanics
- ☐ Splits
- ✓ ThermalContact
- ☐ Transfers
- ☐ UserObjects
- ✓ Variables
- ☐ VectorPostprocessors

Fig. 20. The NiCE MOOSE Model input file block tree. Each node represents a block in the input file.



- ✓ BCs
  - ✓ no_x_all
  - ✓ no_y_clad_bottom
  - ✓ no_y_fuel_bottom
  - ✓ Pressure
  - ✓ PlenumPressure
    - ✓ plenumPressure
  - ✓ convective_clad_surface

Clicking the arrow next to a block name expands and collapses subblocks

Fig. 21. The NiCE MOOSE model tree showing the ability to add and view sub-trees.

5.4.2.2 Adding and Deleting Blocks If you'd like to add a subblock, first select the parent block you'd like to add it to. Next, click the green "+" button located at the top right-hand corner of the *Tree View* (Figure 22). Alternatively, you can also right-click the parent block, and select *Add Child* from the context menu that appears. Doing this will prompt a pop-up dialog to appear. This dialog contains a list of all

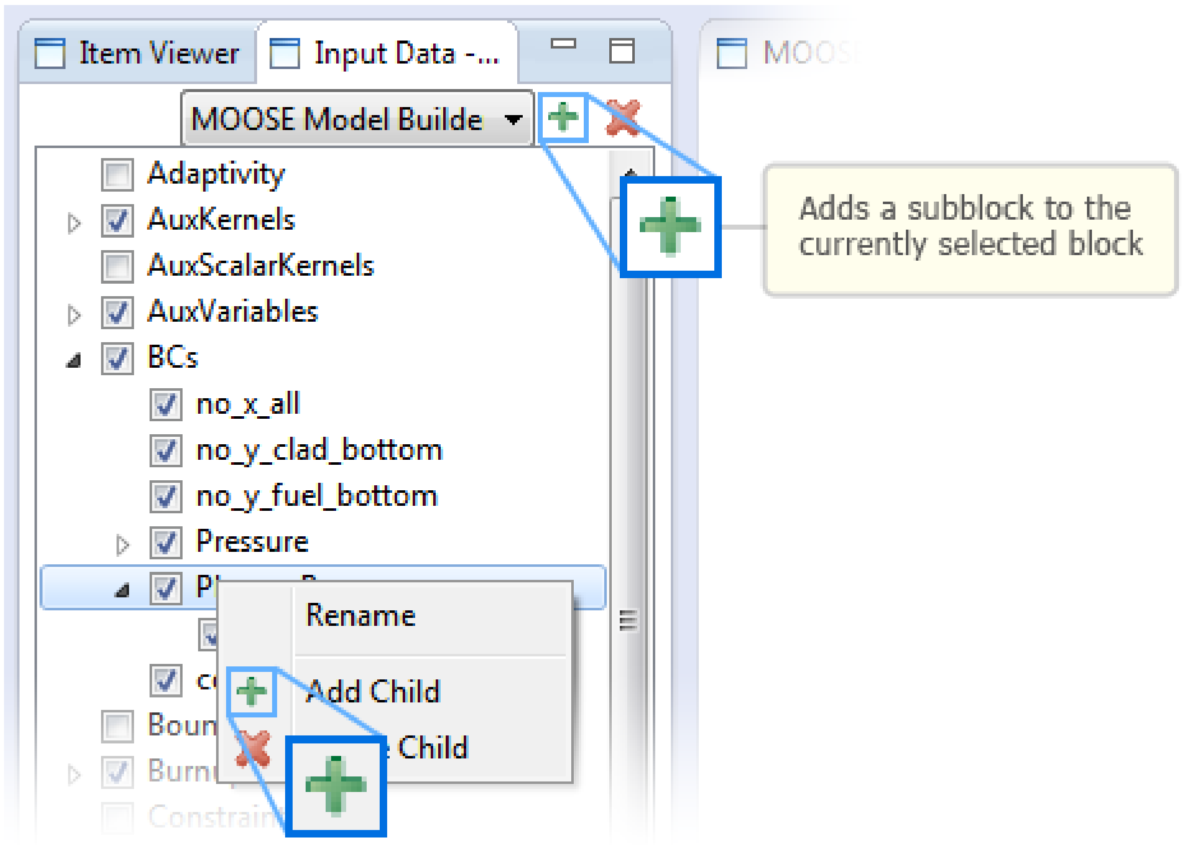
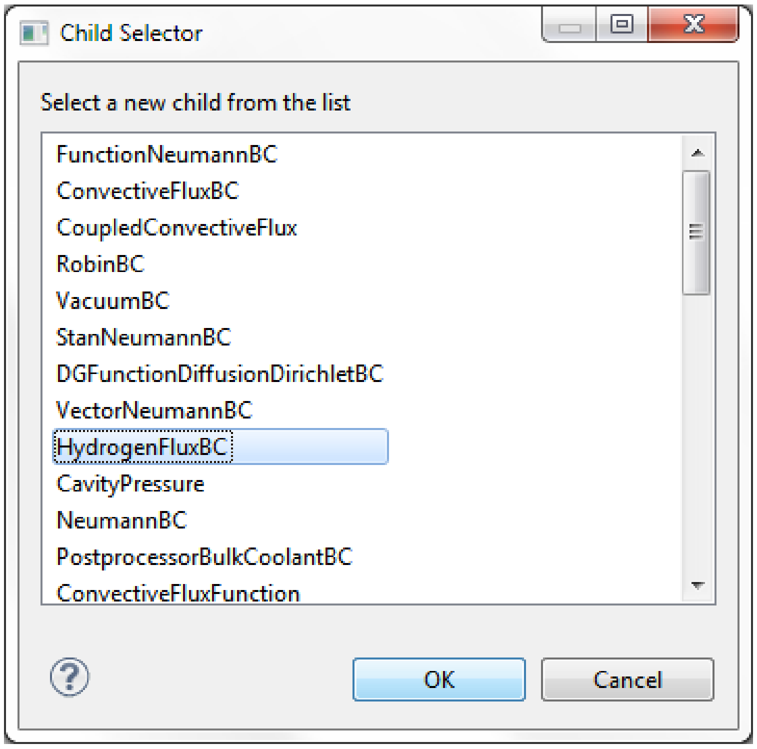


Fig. 22. The NiCE MOOSE tree can be modified by the addition or removal of sub-blocks.

possible subblocks that can be added, according to rules outlined in the YAML files generated earlier (Figure 23). Each block that appears in the list has its own unique set of parameters, with the exception of `BlankBlocks`, which are—as their name suggests—blocks that are totally devoid of any information. Once you've selected a subblock to add, click *OK*, and it will be added in the *Tree View*. Use the arrow next to the parent block to expand/collapse the list of subblocks.

Similarly, to delete a block, select the particular block you'd like to remove, and click the red "x" button located at the top right-hand corner of the *Tree View*. You can also delete a block by right-clicking on it and selecting *Delete Child* from the pop-up context menu. If the red "x" button is greyed-out for a particular block, this means that deletion has been disabled. This is true for all top-level blocks.

5.4.2.3 Renaming Blocks To rename a block, right-click on the block in question and select *Rename* from the context menu that appears. If the renaming option is greyed-out from the context menu, this means that renaming has been disabled for this block. Once again, this is true for all top-level blocks.



FunctionNeumannBC
ConvectiveFluxBC
CoupledConvectiveFlux
RobinBC
VacuumBC
StanNeumannBC
DGFunctionDiffusionDirichletBC
VectorNeumannBC
HydrogenFluxBC
CavityPressure
NeumannBC
PostprocessorBulkCoolantBC
ConvectiveFluxFunction

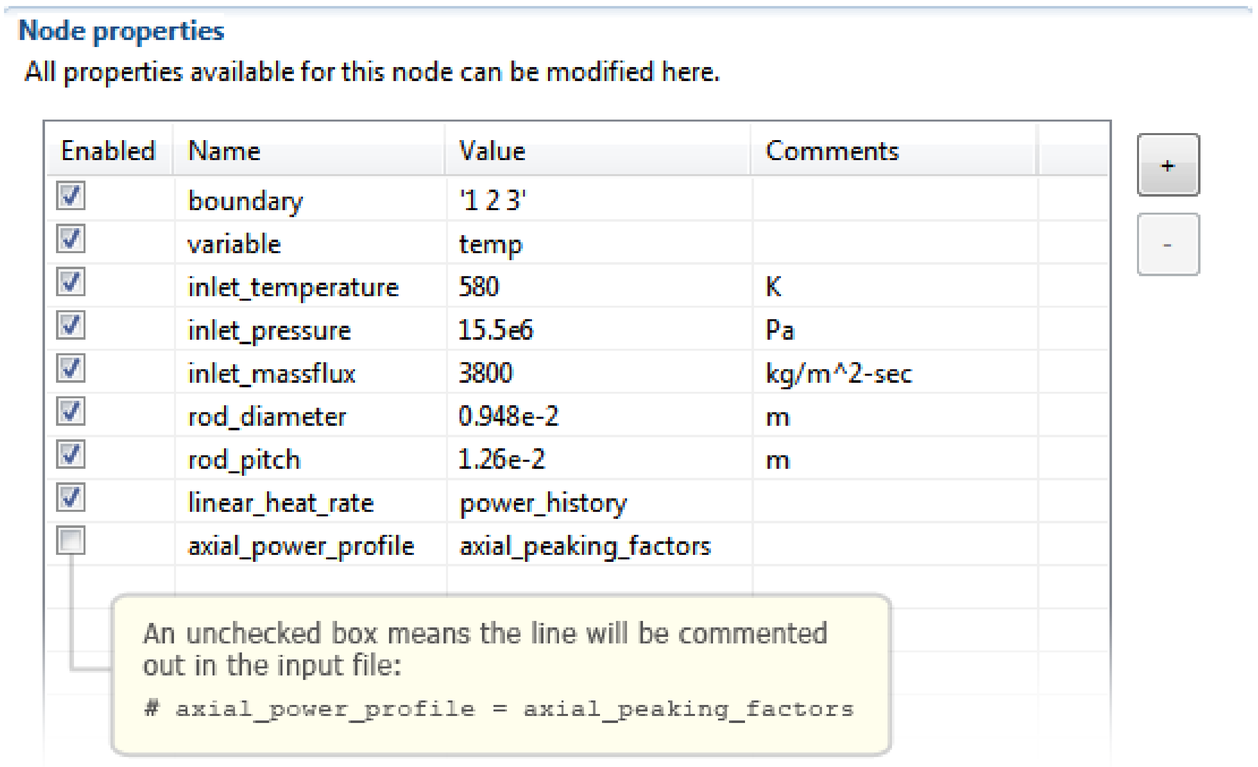
Fig. 23. When adding a new node, a YAML-specified list of possible sub-blocks appear in a convenient dialog.

5.4.3 Editing Parameters

Each block has a set of parameters associated with it. The default list of parameters for each block is drawn from the YAML file that was generated earlier. To add, remove or modify parameters associated to a block, first select a block from the *Tree View*. A table of parameters will then appear in the *Properties* tab referenced earlier (Figure 24). The parameters table displays a parameter name, value and the option for an in-line comment, which can all be edited directly in this table. When written out to file, each parameter will be written to one line in the form:

```
name = value     # comment
```

Additional parameters can be added by clicking the “+" button to the right of the parameter table; parameters can be similarly deleted by clicking the “-" icon. Next to each parameter row, you’ll also notice


Node properties
All properties available for this node can be modified here.
+
-

Enabled	Name	Value	Comments
✓	boundary	'1 2 3'	
✓	variable	temp	
✓	inlet_temperature	580	K
✓	inlet_pressure	15.5e6	Pa
✓	inlet_massflux	3800	kg/m^2-sec
✓	rod_diameter	0.948e-2	m
✓	rod_pitch	1.26e-2	m
✓	linear_heat_rate	power_history	
☐	axial_power_profile	axial_peaking_factors	

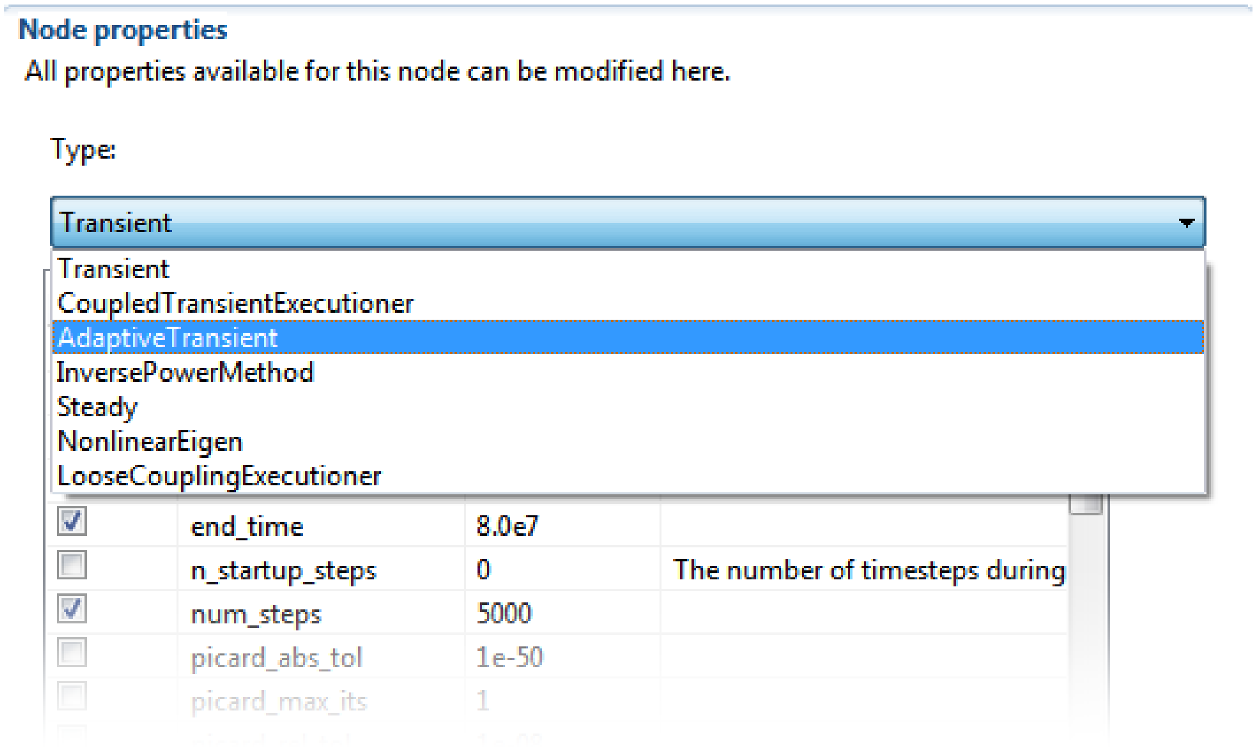
An unchecked box means the line will be commented out in the input file:

```
# axial_power_profile = axial_peaking_factors
```

Fig. 24. The NiCE Properties Tab containing a set of properties associated with the selected MOOSE tree node.

an *Enabled* checkbox. Checking this option means that the parameter associated to it will be written out normally in the file created. Unchecking this option off will cause the entire line to be commented out, and thus, won’t be used during the problem runtime. Note that if you created a *MOOSE Model Builder* by importing data from an already existing input file, NiCE automatically parses any in-line comments or commented out lines (non-sensitive to leading whitespaces). This will be reflected in the *Enabled* and *Comment* columns of the parameters table.

Lastly, some blocks contain a special `type` parameter, such as the Executioner block, which affects the list of parameters associated to the block. In these special instances, an additional drop-down menu will appear in the *Property* tab, above the parameters table. By setting the `type` of the block NiCE



Node properties

All properties available for this node can be modified here.

Type:

Transient

✓	end_time	8.0e7	
	n_startup_steps	0	The number of timesteps during
✓	num_steps	5000	
	picard_abs_tol	1e-50	
	picard_max_its	1	

Fig. 25. The Properties Tab when a selected MOOSE tree node has an adaptive type associated with it.

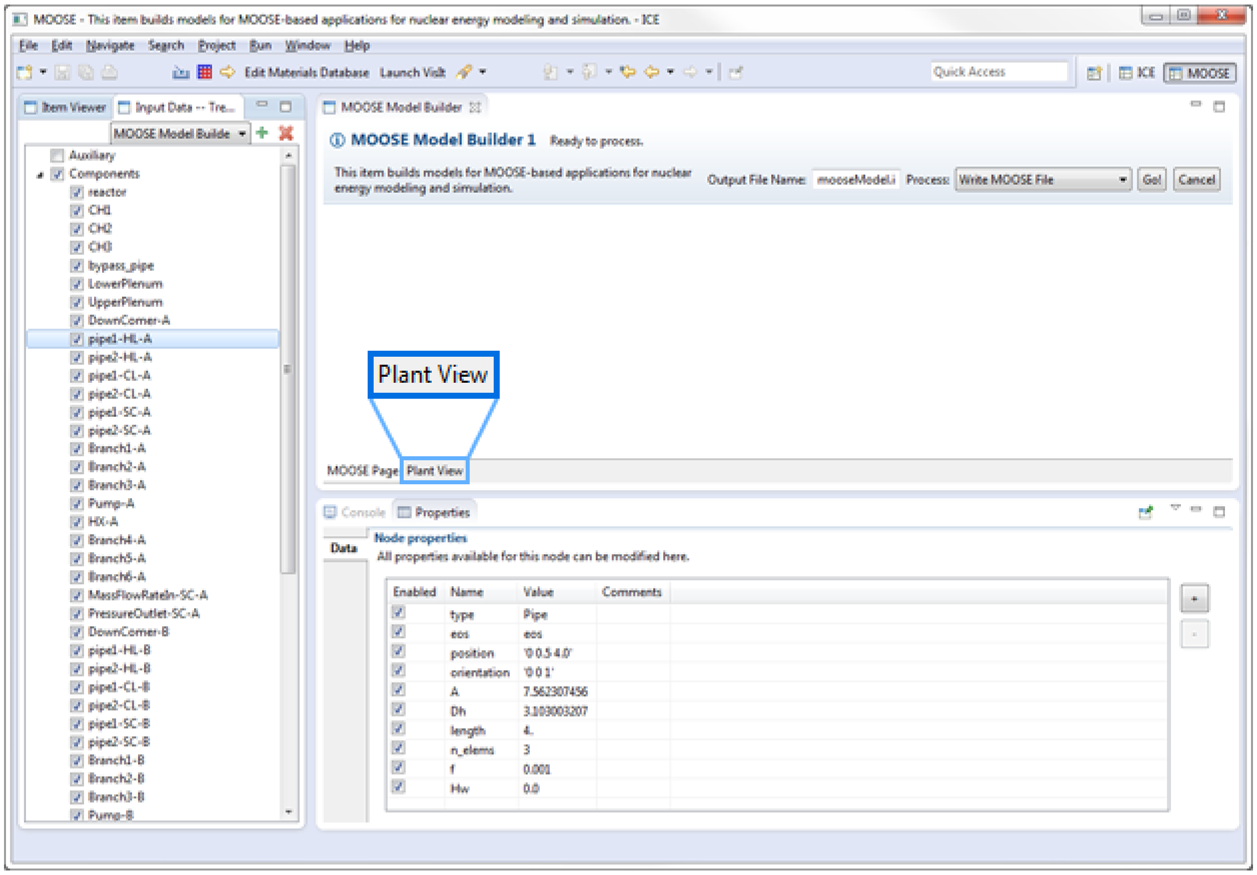
automatically repopulates the parameter table according to the rules defined by the YAML specification.

5.4.4 Viewing the 3D RELAP-7 Plant

For RELAP-7 problems, NiCE includes an interactive 3D *Plant View* to visualize the physical respresentation of the problem. To access the *Plant View*, click the *Plant View* tab located at the bottom of the main *MOOSE Model Builder* panel. Note that the *Plant View* tab is only visible if RELAP-7 has been selected as the MOOSE application in the *Tree View*; the *Plant View* will not be accessible for any other MOOSE applications.

This view draws data from the `Components` block in the *Tree View*; if you have valid components in the `Components` block, then they should be rendered. Before we go on, there are a few things to note about using the *Plant View*. First, only components that are currently enabled in the *Tree View* (i.e. have a checkmark) are rendered. This way, components can easily be turned on and off without having to delete them entirely. And secondly, the *Plant View* updates in real time; any changes that you make to components in the *Tree View* will be reflected immediately such as adding, removing, re-positioning or re-orienting components.

Once your plant components have rendered, you can move around the 3D space by using the following keyboard controls:

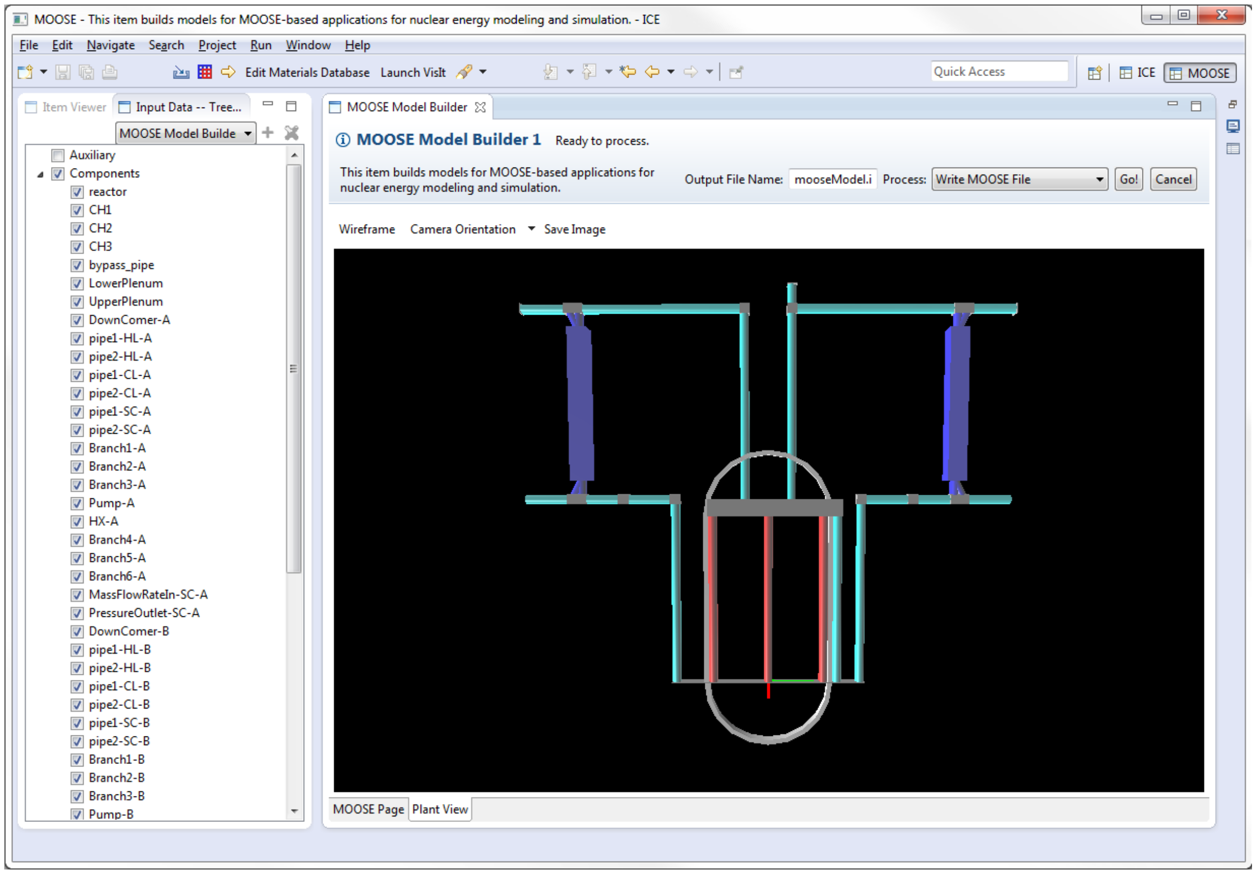


This item builds models for MOOSE-based applications for nuclear energy modeling and simulation.

All properties available for this node can be modified here.

Enabled	Name	Value	Comments
✓	type	Pipe	
✓	eos	eos	
✓	position	'0 0.5 4.0'	
✓	orientation	'0 0 1'	
✓	A	7.562307456	
✓	Dh	3.103003207	
✓	length	4.	
✓	n_elems	3	
✓	f	0.001	
✓	Hw	0.0	

Fig. 26. A view of the NiCE Plant View tab.



MOOSE Model Builder 1 Ready to process.

This item builds models for MOOSE-based applications for nuclear energy modeling and simulation.

Output File Name: mooseModel.i Process: Write MOOSE File

Fig. 27. A view of the NiCE Plant View.

Table 1. Camera Keyboard Controls

Movement	Key	Rotation	Key
Left	A	Rotate clockwise	Q
Right	D	Rotate counterclockwise	E
Up	C	Pivot left*	âĘŘ
Down	Space	Pivot right*	âĘŠ
Zoom in	W	Pivot up*	âĘŚ
Zoom out	S	Pivot down*	âĘŞ

The camera viewing-angle can also be pivoted by left-clicking and dragging.
Finally, the *Plant View* has 3 tools located in the toolbar above the 3D viewing window.

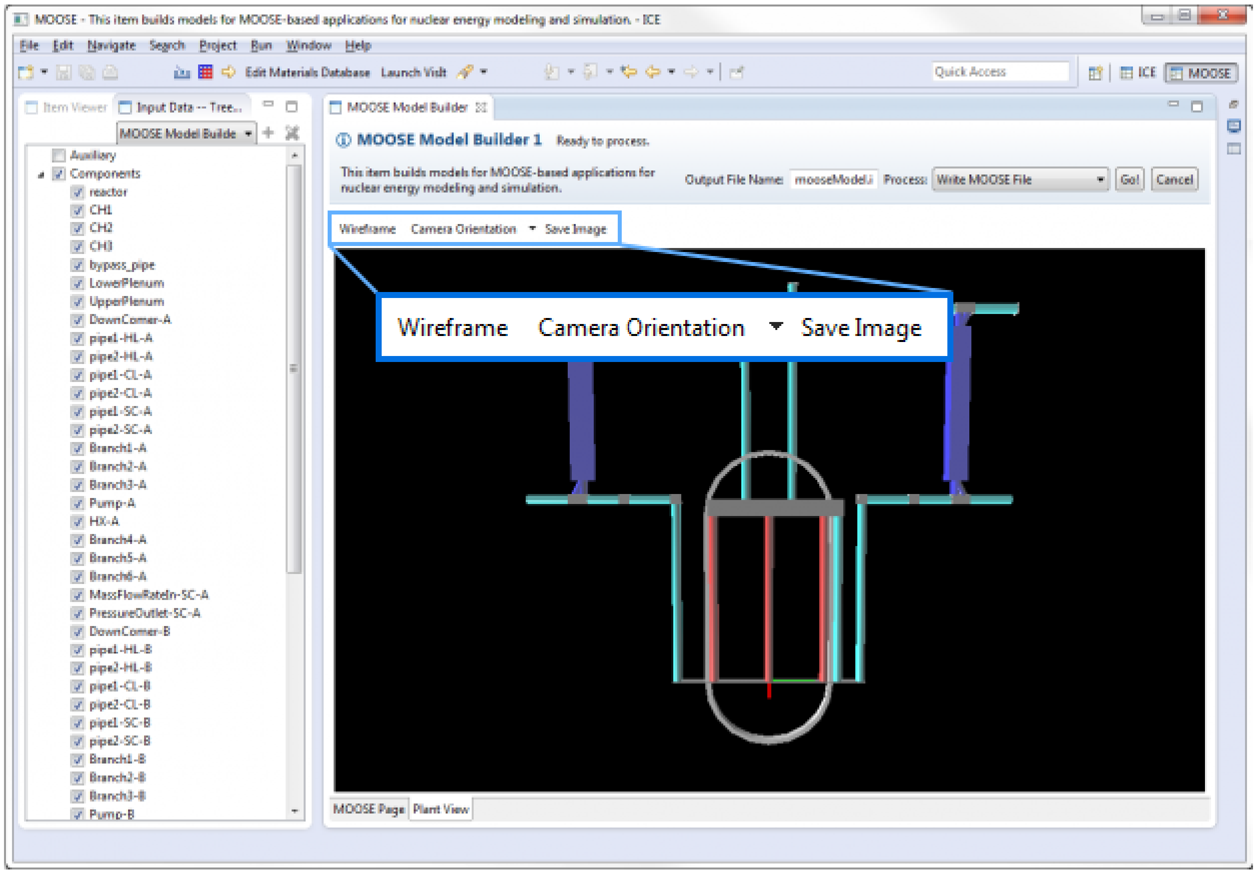
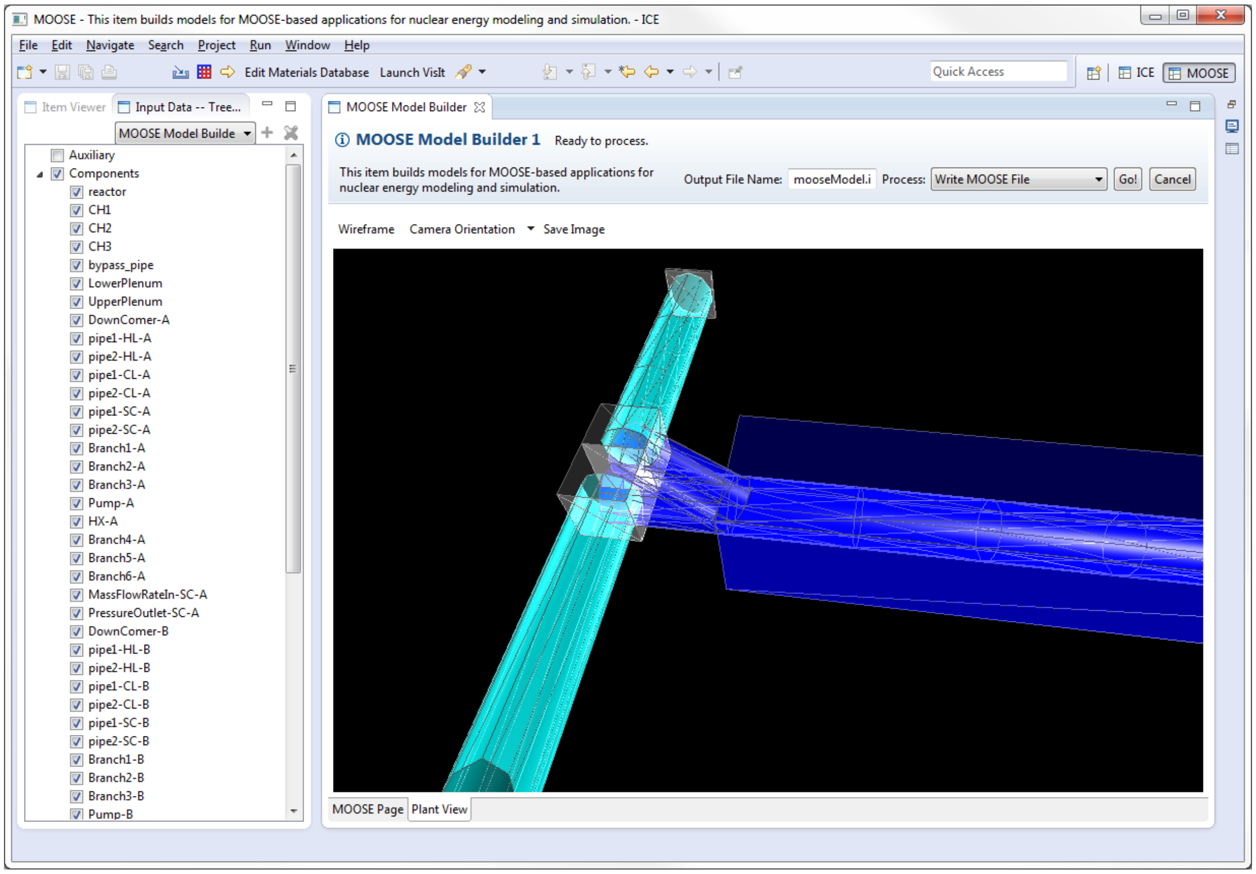


Fig. 28. A view of the NiCE Plant View toolbar buttons.

- **Wireframe** - Clicking this will toggle the plant's wireframe on and off; this allows you see how the meshes for the rendering engine are constructed. In certain cases, this may be useful for verifying the plant model before running a simulation. For instance, pipe components in RELAP-7 have a property called "n_elems" representing the number of cylindrical cross-section slices along the



MOOSE - This item builds models for MOOSE-based applications for nuclear energy modeling and simulation. - ICE

MOOSE Model Builder 1 Ready to process.

This item builds models for MOOSE-based applications for nuclear energy modeling and simulation.

Output File Name: mooseModel.i Process: Write MOOSE File

Fig. 29. The RELAP-7 NiCE Plant View with wireframe toggled on.

pipe's length. By switching to wireframe mode, the user can see the same number of cylindrical sections stacked together that represent the pipe's physical construction.

- **Camera Orientation** - You can re-orient the rendering engine's default camera view by clicking on the *Camera Orientation* button. The default orientation—YZ—is the standard physics orientation with the Y axis increasing to the right, the Z axis increasing upwards, and the camera looking at the YZ-plane along the positive X axis. In the same "Camera Orientation" menu, we provide two alternative default orientations: the XY orientation shows the XY-plane from along the positive Z axis, and the ZX orientation shows the ZX-plane from along the positive Y axis. Selecting *Reset to current default* will snap the camera back to the origin view should you ever get lost.

- **Save Image** - Clicking this will prompt a pop-up window to save a `.png` image of the current *Plant View* to your local filesystem.

5.4.5 Exporting Images

Users can now export the 3D plant view as a static image. To use this feature, press the "Save Image" button on the plant view's ToolBar. Currently, only the PNG file format is supported for exported plant view images. The 3D plant view. The wireframe toggle button is highlighted in green, the camera orientation settings in magenta, and an example tool tip in yellow. The save image button and the camera movement and rotation buttons are also highlighted.

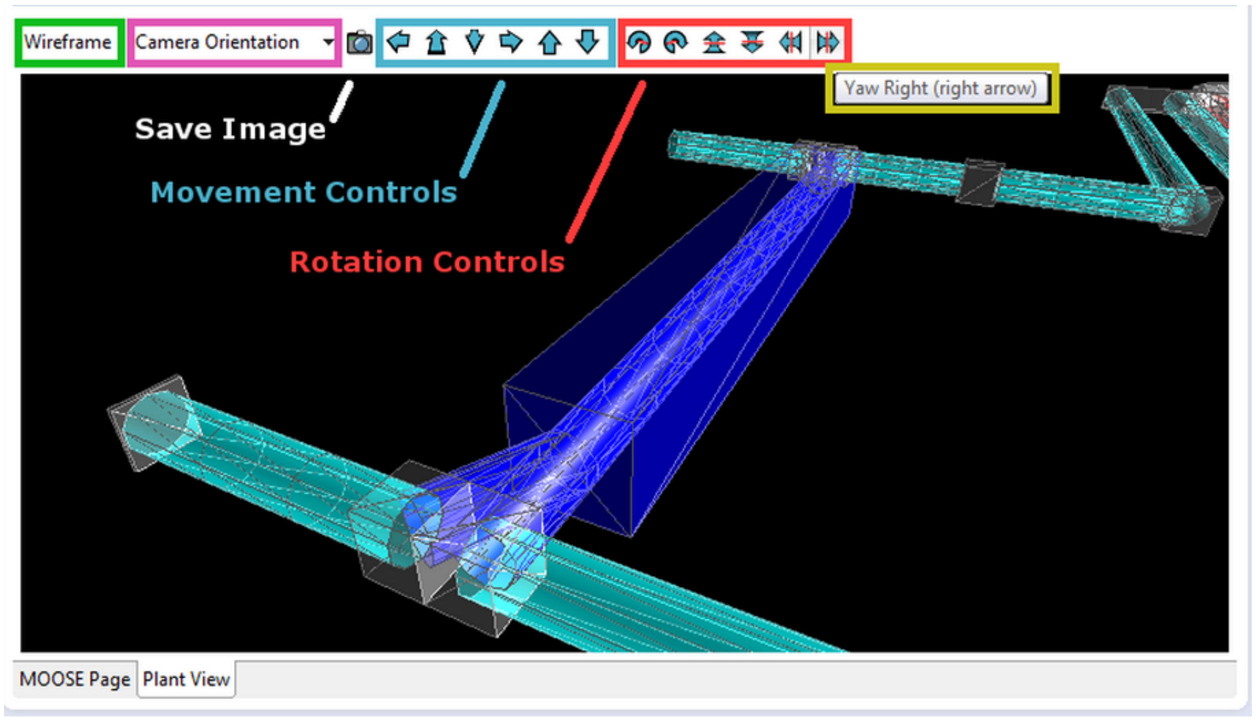


Fig. 30. A view of the newly added Plant-view controls.

5.4.6 Creating the File

Once you have edited your blocks and associated parameters to your liking, the last step is to write them to file. Any block in the *Tree View* with a checkmark next to it will be written to file, and any block without a checkmark will not. Ensure that you've correctly selected all the necessary blocks. Save your work by clicking the floppy-disk save icon (or *Ctrl+S*).

At the top of the *MOOSE Model Builder* tab, specify the name of the file you'd like to write, set the *Process* drop-down menu to "Write MOOSE File", and click the *Go!* button.

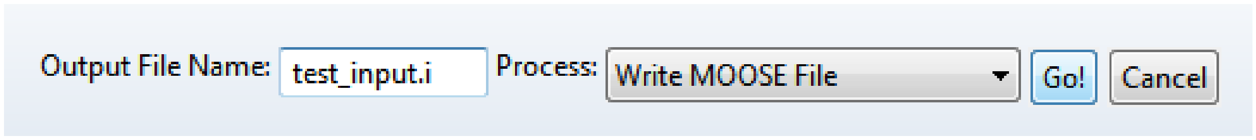

Output File Name: test_input.i
Process: Write MOOSE File

Fig. 31. ICE_MOOSEWriteFile.png

This will write the contents of your *Tree View* to the specified filename, and will be placed it in your `$HOME/ICEFiles/default` directory. If you wish to review the file before moving onto the next section, you can do so by using the NiCE toolbar and navigating to: *File > Open File...*

5.5 Launching a MOOSE Job

Once you've generated appropriate input files, launching a MOOSE job is a relatively simple task. To get started, click the green "+" button in the *Item Viewer* once more to create a new NiCE Item. Select *MOOSE Launcher* from the menu that pops up and click *Finish*. A form will appear in the main NiCE workbench area. This form contains the information necessary for launching a MOOSE problem.

5.5.1 Selecting the MOOSE Product

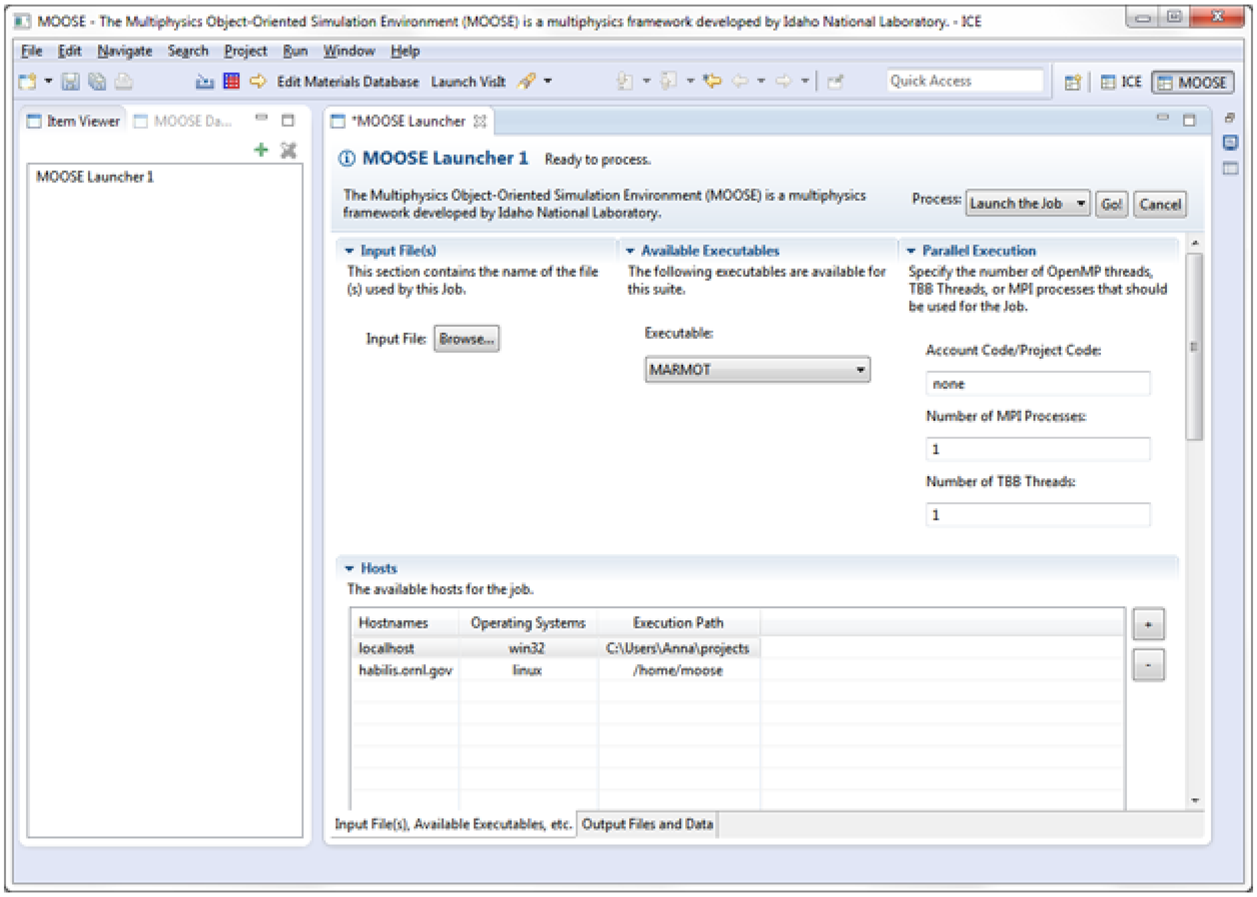
The first piece of necessary information is to set which MOOSE-based application you'd like to run. From the list of *Available Executables*, select an application, and apply your choice by clicking the save button in the upper left-hand corner of the workbench (*Ctrl+S*).

5.5.2 Selecting the Input File(s)

From the *Input File(s)* drop-down menu, select an appropriate MOOSE input file. This drop-down menu displays all `*.i` files that were in the `$HOME/ICEFiles/default` directory at the time the *MOOSE Launcher* was created. If you created your own input file in the previous step using the *MOOSE Model Builder*, this file should appear in the list of available files.

If you'd like to use an input file not found in this list, click the *Browse...* button; a file browser will pop up for you to locate the file you'd like to use. Once you've selected a file, it will be imported into the `$HOME/ICEFiles/default` directory.

If your problem requires additional files, such as a mesh, peaking factors, or power history data, do the same for these as well. The next step is to tell NiCE which machine your MOOSE code will be run on, either locally or remotely. A list of hosts used at ORNL is displayed by default, however, additional hosts can be added by clicking the "+" button to the right of the *Hosts* table (Figure 35. When adding hosts, set the *Execution Path* to the trunk directory of the machine's MOOSE installation. For example, if you are launching RELAP-7 on a machine with the follow structure:


MOOSE - The Multiphysics Object-Oriented Simulation Environment (MOOSE) is a multiphysics framework developed by Idaho National Laboratory. - ICE
File Edit Navigate Search Project Run Window Help
Edit Materials Database Launch VisIt
Quick Access
ICE MOOSE
Item Viewer MOOSE Da...
MOOSE Launcher 1
*MOOSE Launcher
Input File(s), Available Executables, etc. Output Files and Data

MOOSE Launcher 1 Ready to process.

The Multiphysics Object-Oriented Simulation Environment (MOOSE) is a multiphysics framework developed by Idaho National Laboratory.

Process: Launch the Job Go! Cancel

Input File(s)
This section contains the name of the file (s) used by this Job.
Input File: Browse...

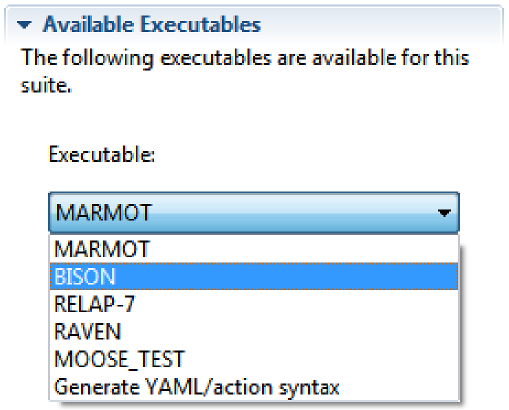
Available Executables
The following executables are available for this suite.
Executable: MARMOT

Parallel Execution
Specify the number of OpenMP threads, TBB Threads, or MPI processes that should be used for the Job.
Account Code/Project Code: none
Number of MPI Processes: 1
Number of TBB Threads: 1

Hosts
The available hosts for the job.

Hostnames	Operating Systems	Execution Path
localhost	win32	C:\Users\Anna\projects
habilis.ornl.gov	linux	/home/moose

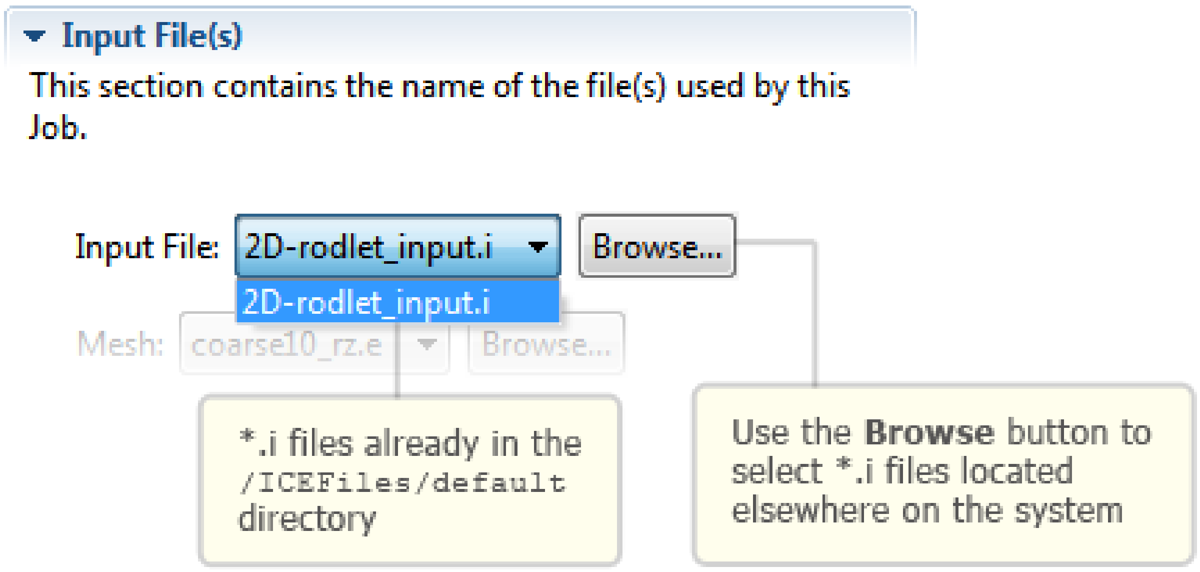
Fig. 32. A view of the NiCE MOOSE Launcher Item.



Available Executables
The following executables are available for this suite.

Executable: MARMOT

Fig. 33. Use the drop-down to select the available MOOSE executable you'd like to launch.



Input File(s)

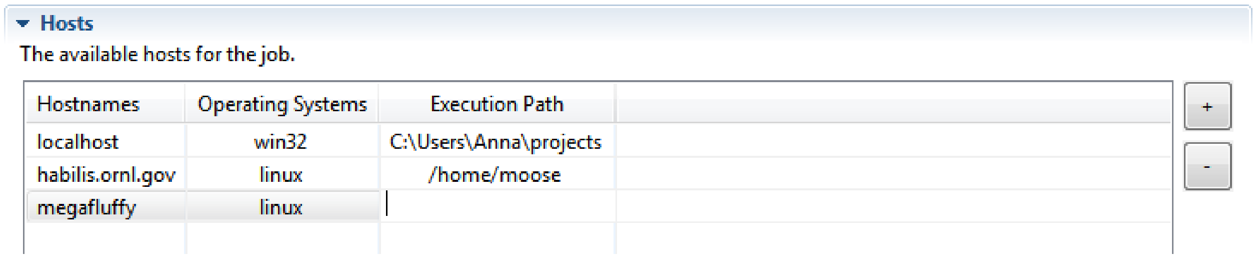
This section contains the name of the file(s) used by this Job.

Input File: 2D-rodlet_input.i

Mesh: coarse10_rz.e

Browse...
2D-rodlet_input.i
Browse...
*.i files already in the /ICEFiles/default directory
Use the **Browse** button to select *.i files located elsewhere on the system

Fig. 34. Select the input file you'd like to launch with the selected MOOSE application.



Hosts

The available hosts for the job.

Hostnames	Operating Systems	Execution Path
localhost	win32	C:\Users\Anna\projects
habilis.ornl.gov	linux	/home/moose
megafluffy	linux	

Fig. 35. The NiCE Host name table. Hosts can be added by simply clicking the '+' button.

```
/home/user/trunk/relap-7/relap-7-opt
```

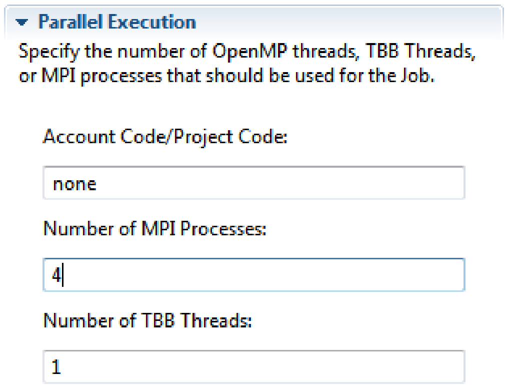
I would set the execution path in NiCE to be:

```
/home/user/trunk
```

If you are launching on a remote machine, also be sure that you have appropriate privileges for the execution path.

5.5.3 Setting Parallel Execution (Optional)

If you'd like to take advantage of parallel processing, you may specify the number of MPI process and/or Intel Thread Building Block (TBB) threads. To use multiple MPI processes, change the marked field


Parallel Execution
Specify the number of OpenMP threads, TBB Threads, or MPI processes that should be used for the Job.

Account Code/Project Code: none

Number of MPI Processes: 4

Number of TBB Threads: 1

Fig. 36. The NiCE Parallel Execution tab, used for setting any parallel options for the given MOOSE run.

to an integer value anywhere between 1 and 10000. Note that `mpirun` must be specified in the host machine's `PATH` variable. If you choose not to change this field, the default value of 1 MPI process is used.

To use multiple TBB threads, change the marked field to an integer value anywhere between 1 and 256. Note that the host machine must have Intel TBB support. If you choose not to change this field, the default value of 1 TBB thread is used.

5.5.4 Launching the Problem

Once the input file(s), host, and any parallel execution options are specified, save your settings. If you make any subsequent changes to the *MOOSE Launcher* form, you will have to re-apply them by saving the form in the same way.

Finally, using the *Process* menu in the upper right-hand corner, select the *Launch the Job* task from the drop-down menu and click the *Go!* button. Depending on your host machine's configuration, you may be prompted for login credentials. You should shortly begin seeing the standard console output in NiCE as your problem begins to solve.

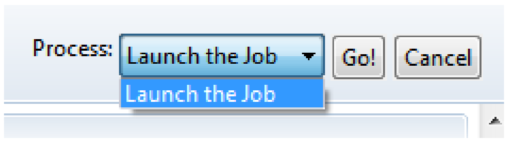


Fig. 37. Launch the MOOSE job with the Process drop-down menu.

5.6 Visualizing Output with VisIt

5.6.1 Prerequisites

To use the *VisIt Tools*, NiCE requires the installation of VisIt https://wci.llnl.gov/simulation/computer-codes/visit/ (minimum version 2.8.2) developed by Lawrence Livermore National Laboratory, either locally or on a remote machine. The *CSV Plotting Tools* require no additonal software to be installed.

5.6.2 Visualization Perspective

To use NiCE's visualization tools, you first must switch to the *Visualization Perspective*. The *Visualization Perspective* is an alternative workbench that contains UI components necessary for visualization that are not normally exposed in the default NiCE perspective. To access the *Visualization Perspective*, use the the NiCE toolbar at the top and navigate to: *Window > Open Perspective > Other...* Select *Visualization* in the window that pops up and click *OK*. Alternatively, you can also access the same pop-up menu by clicking the *Open Perspective* button in the upper right-hand corner of the NiCE workbench. Once the *Visualization Perspective* opens, you should notice the workbench contains some new UI components. Make note of the following panels, as we'll be referring to them in the following sections.

5.6.3 Connecting to VisIt

Once you've switched to the *Visualization Perspective*, the first step necessary is to connect to your VisIt installation through NiCE. To do this, click on the *Launch VisIt* button located in the NiCE toolbar near the top. A dialog will pop up offering you three options for connecting to VisIt:

1. **Launch VisIt locally**
   If you installed VisIt on your local machine, use the *Browse* button to direct NiCE to your local installation directory. Using this method of connecting will launch a new VisIt session. Optionally, you can also set a port number (default 9600) and password if you might want to share your VisIt session with another user.
2. **Launch VisIt remotely**
   If you installed VisIt on a remote machine, specify the hostname and full path to the VisIt installation directory. Using this method of connecting will launch a new VisIt session. Optionally, you can specify a port number (default 9600) and password if you might want to share your VisIt session with another user. If you need or want to use an external gateway, you may specify a URL and port number as well.
3. **Connect to VisIt**
   If you'd like to connect to session of VisIt already running somewhere else, specify the hostname, port number and password set on the VisIt session; you will need to obtain this information from the

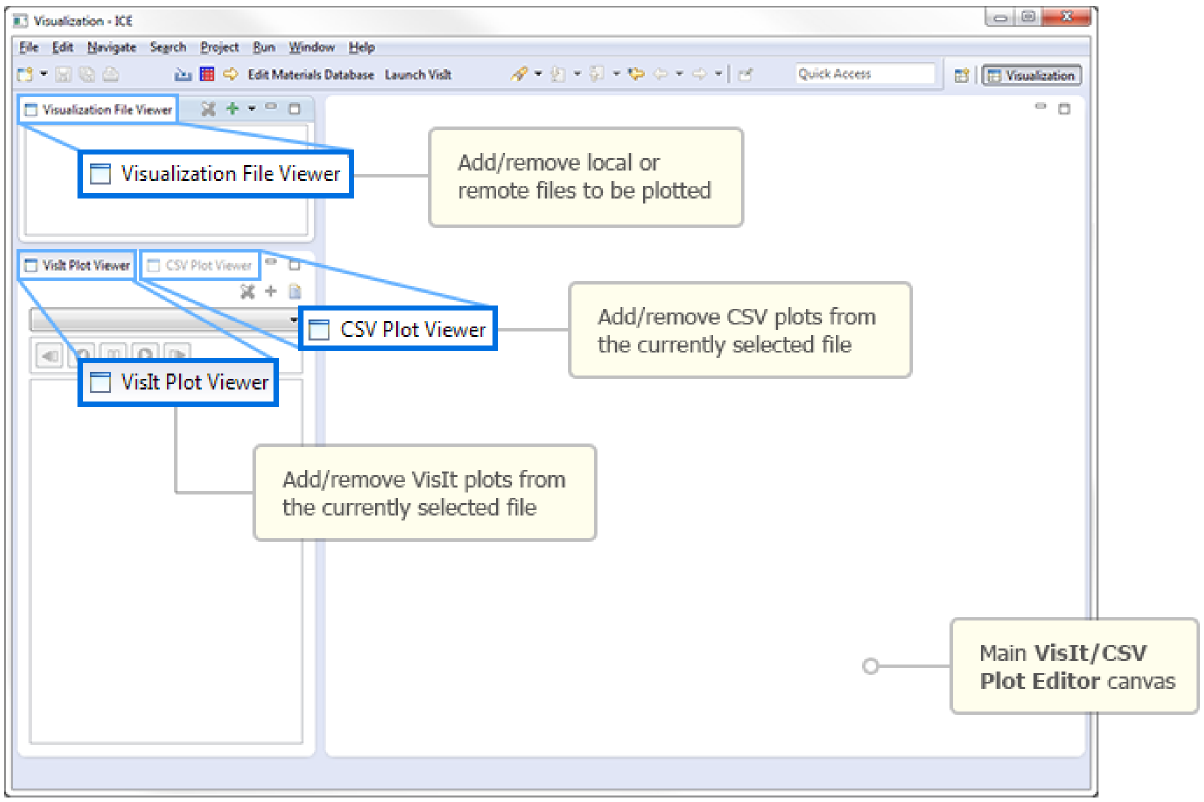

Visualization - ICE
File Edit Navigate Search Project Run Window Help
Edit Materials Database Launch VisIt
Quick Access
Visualization
Visualization File Viewer
Visualization File Viewer
Add/remove local or remote files to be plotted
VisIt Plot Viewer
CSV Plot Viewer
CSV Plot Viewer
Add/remove CSV plots from the currently selected file
VisIt Plot Viewer
Add/remove VisIt plots from the currently selected file
Main VisIt/CSV Plot Editor canvas

Fig. 38. The NiCE Visualization Perspective.

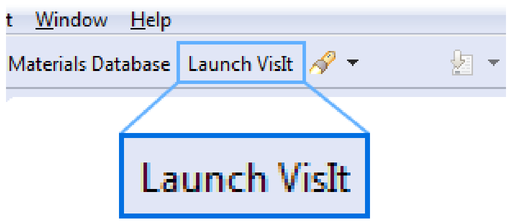
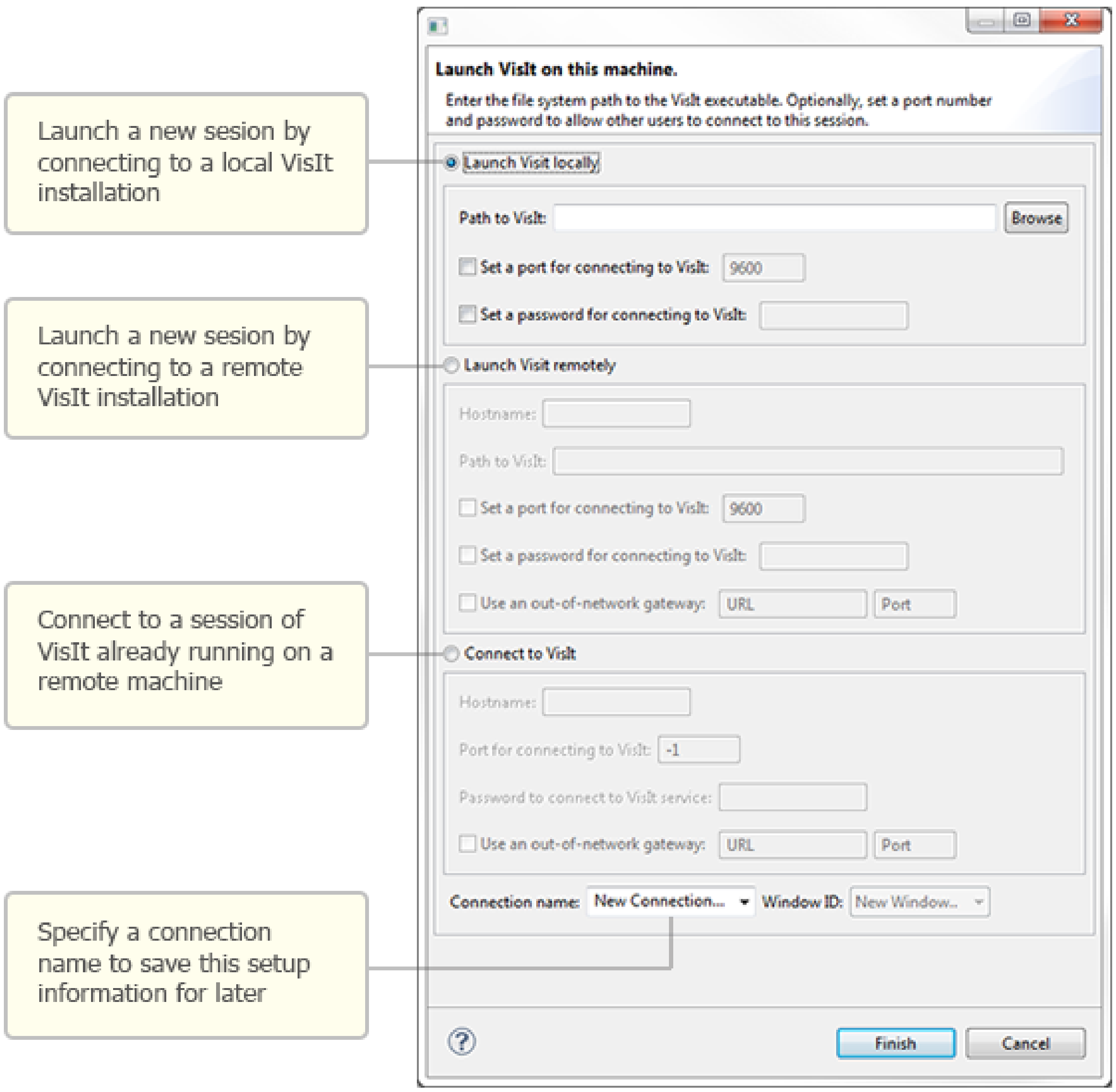


Fig. 39. The *Launch VisIt* action button.



Launch a new sesion by connecting to a local VisIt installation

Launch a new sesion by connecting to a remote VisIt installation

Connect to a session of VisIt already running on a remote machine

Specify a connection name to save this setup information for later

Launch VisIt on this machine.

Enter the file system path to the VisIt executable. Optionally, set a port number and password to allow other users to connect to this session.

(●) Launch VisIt locally

Path to VisIt: [ ] Browse

☐ Set a port for connecting to VisIt: 9600

☐ Set a password for connecting to VisIt:

( ) Launch VisIt remotely

Hostname:

Path to VisIt:

☐ Set a port for connecting to VisIt: 9600

☐ Set a password for connecting to VisIt:

☐ Use an out-of-network gateway: URL Port

( ) Connect to VisIt

Hostname:

Port for connecting to VisIt: -1

Password to connect to VisIt service:

☐ Use an out-of-network gateway: URL Port

Connection name: New Connection... Window ID: New Window...

Finish Cancel

Fig. 40. The NiCE Launch VisIt Wizard, showing the different methods for connecting to VisIt.

person who initially launched the VisIt session. If you need or want to use an external gateway, you may specify a URL and port number as well.

Regardless of which method you choose to connect to VisIt, enter a *Connection name* at the bottom of the pop-up dialog. This will allow you to re-use this connection information in the future. If you are connecting to an existing session, specify a *Window ID* between 1 and 16 as well. Which *Window ID* to use depends on how you would like to connect to VisIt. If multiple users connect using the same *Window ID*, they will all see, and be able to interact with, the same screen. However, if you would like multiple users to each have their own unique session, assign a unique *Window ID* to each user. The VisIt installation will be able to serve up to 16 unique connections at a time. Once you are done, click the *Finish* button at the bottom, and NiCE should begin connecting to VisIt.

5.6.4 Adding/Deleting Files

To open a file, find the green “+" icon in the *Visualization File Viewer*. Clicking directly on the green “+" icon will prompt a local file browser to pop up. However, if your file is located on a remote machine, or if you’d like to add a file set, click on the drop-down button next to the green “+" icon.
This will offer you four ways to open file(s):

- Open a local file
- Open a remote file
- Open a local file set
- Open a local SILO set

Once you’ve selected your file(s), they should appear in the *Visualization File Viewer*. Finally, if you’d like to remove a file from the *Visualization File Viewer* list, select it, and click the red “X” button.

5.6.5 Adding/Removing Plots

To begin adding plots, select your file in the *Visualization File Viewer* and click the green “+" icon in the *VisIt Plot Viewer*. If there is any plot-able data associated to your file, a dialog will pop up with a list of

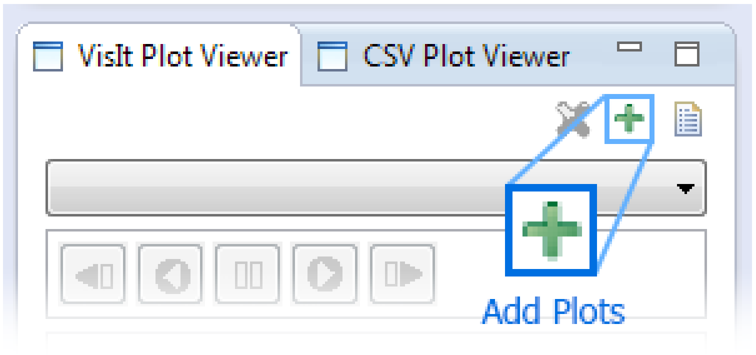


Fig. 41. To add a VisIt plot, click the Add Plot button in the NiCE VisIt Plot Viewer.

options to chose from. This can include mesh plots, scalar plots, vector plots, material block plots, and so forth. If there are multiple plots of each type available, you can select them all by checking off the entire

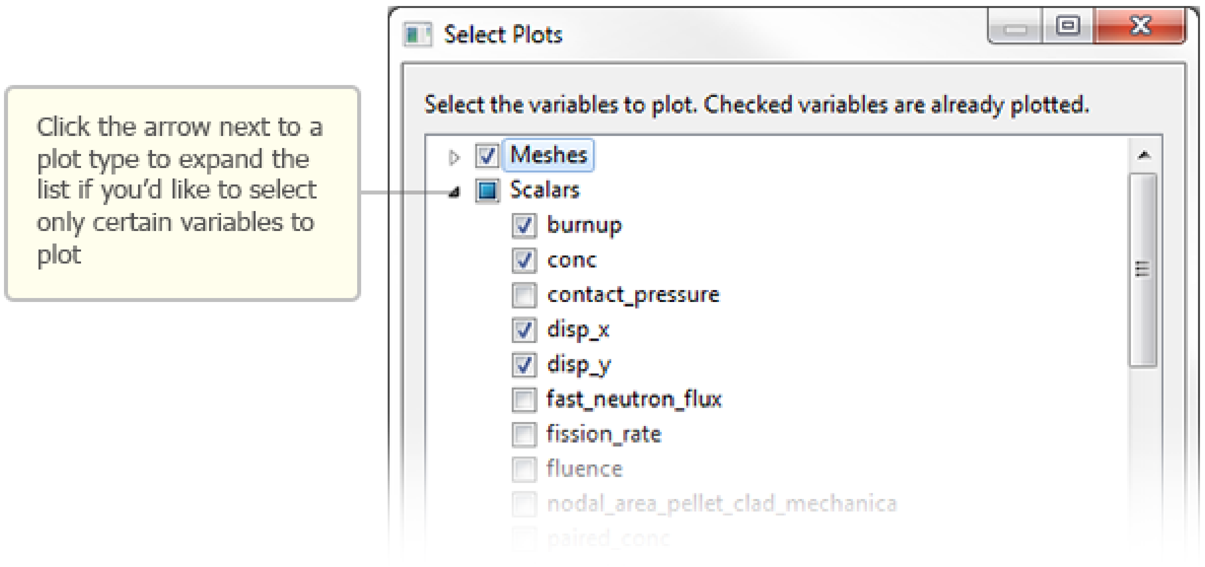

Click the arrow next to a plot type to expand the list if you'd like to select only certain variables to plot
Select Plots
Select the variables to plot. Checked variables are already plotted.
Meshes
Scalars
burnup
conc
contact_pressure
disp_x
disp_y
fast_neutron_flux
fission_rate
fluence
nodal_area_pellet_clad_mechanica
paired_conc

Fig. 42. The NiCE VisIt Add Plot Dialog.

category, or expand it to check off only selected plots. When you're done selecting your plot(s), click *OK*. The selected plots should appear as a list in the *VisIt Plot Viewer*. Lastly, if you'd like to remove a plot from the *VisIt Plot Viewer* list, select it and click the red "x" button.

5.6.6 Rendering Plots

To render a plot, simply click on it in the *VisIt Plot Viewer* and it will appear in the main *VisIt Editor* canvas. The *VisIt Plot Viewer* contains a drop-down menu with a list of plotting styles available for the currently selected plot. Depending on your selected plot, this can include mesh, pseudo-color, contour, volume, and so forth. Use this drop-down menu to select the plotting style you prefer, and the *VisIt Editor* will update in real time. The *VisIt Editor* is also interactive in that you can move your plot around by clicking and dragging the canvas. This may not necessarily be particularly useful for 2D plots, however, it allows you take a fully rotatable look at 3D plots like in the example below. And lastly, if there is any time series data associated to your plot, you can manually walk through the time steps, or play them continuously as a short video, using the playback buttons located in the *VisIt Plot Viewer*.

5.6.7 Executing Python Commands

While many of VisIt's features are already accessible in NiCE, work to enable a more robust list is continually on-going. In the meantime, features not yet integrated into NiCE can still be accessed via Python commands by clicking the Python script button located in the *VisIt Plot Viewer*.

Writing Python scripts for VisIt is beyond the scope of this tutorial, however, you are welcome to refer to the VisIt Python Interface Manual (`https://wci.llnl.gov/simulation/computer-codes/visit/manuals`) provided by the VisIt development team at Lawrence Livermore National Laboratory.

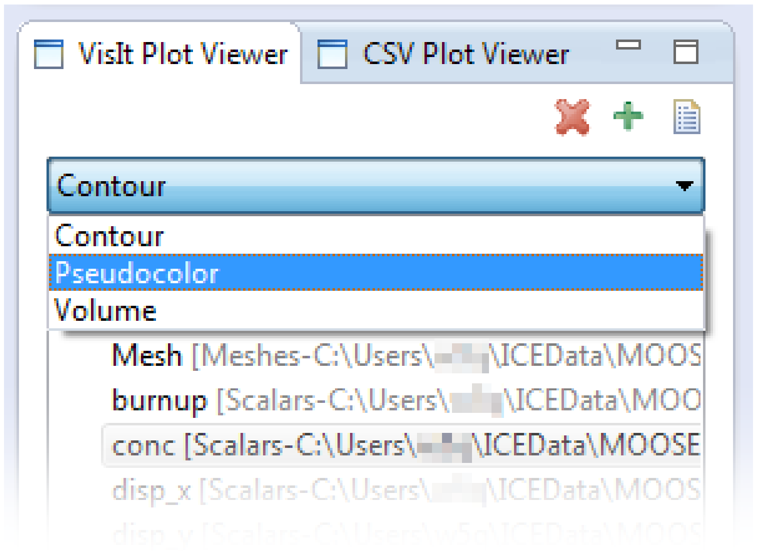

VisIt Plot Viewer
CSV Plot Viewer
Contour
Contour
Pseudocolor
Volume
Mesh [Meshes-C:\Users\ \ICEData\MOOS
burnup [Scalars-C:\Users\ \ICEData\MOO
conc [Scalars-C:\Users\ \ICEData\MOOSE
disp_x [Scalars-C:\Users\ \ICEData\MOOS

Fig. 43. To change the VisIt plot style, click the drop-down and select the desired style.

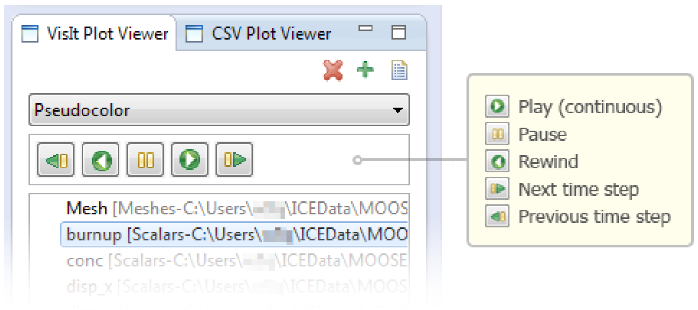

VisIt Plot Viewer
CSV Plot Viewer
Pseudocolor
Play (continuous)
Pause
Rewind
Next time step
Previous time step
Mesh [Meshes-C:\Users\ \ICEData\MOOS
burnup [Scalars-C:\Users\ \ICEData\MOO
conc [Scalars-C:\Users\ \ICEData\MOOSE
disp_x [Scalars-C:\Users\ \ICEData\MOOS

Fig. 44. The NiCE-VisIt time step control buttons.

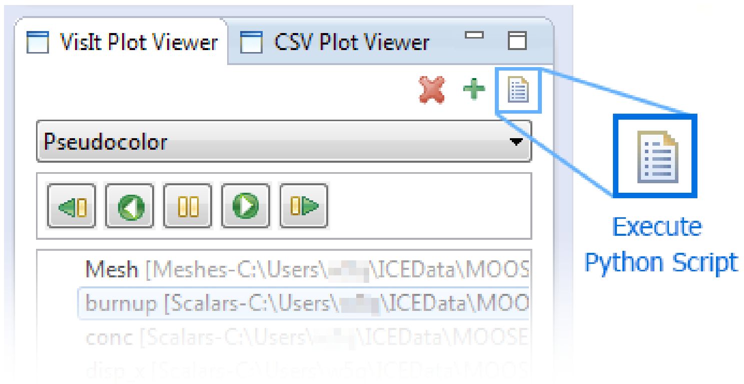

VisIt Plot Viewer
CSV Plot Viewer
Pseudocolor
Execute Python Script
Mesh [Meshes-C:\Users\ \ICEData\MOOS
burnup [Scalars-C:\Users\ \ICEData\MOO
conc [Scalars-C:\Users\ \ICEData\MOOSE

Fig. 45. To execute a python script to modify the VisIt plot, click the Execute Python Script button in the VisIt Plot Viewer.

6. Acknowledgements

The NiCE team would like to specifically acknowledge the RELAP-7 team, specifically Rich Martineau and Cristian Rabiti, 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) and the Advanced Modeling and Simulation Office within DOE-NE.