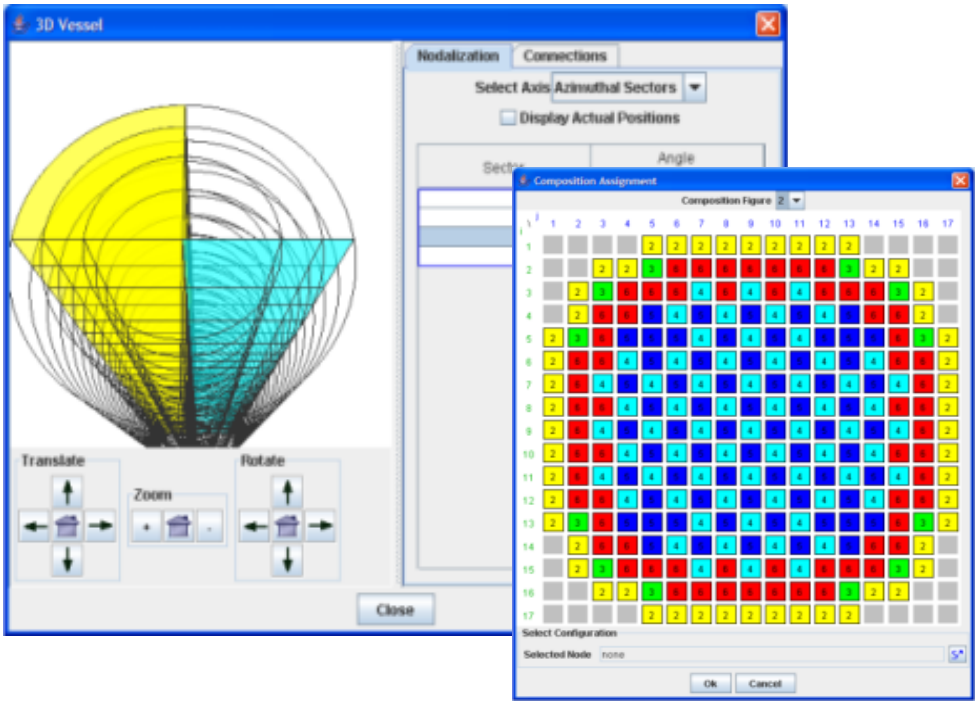
Symbolic Nuclear Analysis Package (SNAP)
SNAP/TRACE User Workshop
Intermediate SNAP Exercises

March 2018



Applied Programming Technology, Inc.
Bloomsburg, PA 17815

Table of Contents

Introduction

This set of exercises focuses on intermediate and advanced SNAP functionality. The following topics are covered:

- SNAP Variables – tools for defining values used across the model and functions that modify their values from user-defined code.
- Interactive Controls – tools for changing the state of an active calculation.
- Post-processing – animating stream task results and annotating a model.
- Model Notes – HTML annotations that can be attached to any component or attribute in the model.
- Working with AptPlot in a Job Stream.

The following assumptions are made by these exercises:

- The SNAP software suite has been installed.
- The user is familiar with SNAP basics: basic use of the Navigator and Property View, creating and connecting components, opening views, etc..
- A TRACE executable is available on the user's machine.
- AptPlot has been installed.
- Either LibreOffice, OpenOffice.org, or Microsoft Word with the ODF plug-in, have been installed.
- A PDF viewer has been installed.

Exercise 12. SNAP Variables and Parametrics

This exercise begins with an example of the SNAP variables feature. Variables provide a means of assigning calculated values to specific model inputs. This provides a great deal of flexibility: a single variable can be referenced in multiple places, edited in a 2D View, adjusted by functions, and varied for parametric submit.

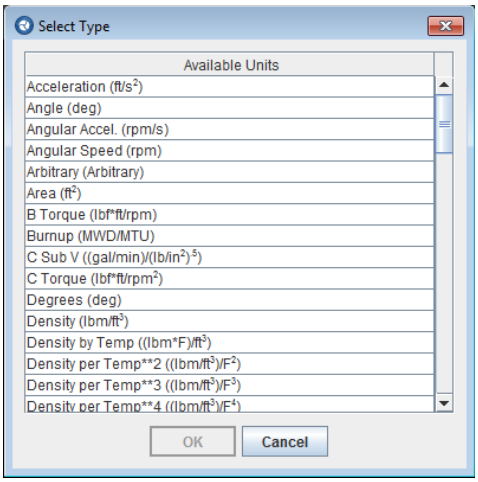
The first series of steps will create a variable and a reference to it.

1. Open **SNAP_Exercises/StandPipe5.med**, included with these exercises and make the following modifications:
    1. Select **StandPipeStream** in the Navigator and make sure its **Platform** is set to "**Local**" and its **Root Folder** to "**runs**" (or the appropriate root folder used in place of the **runs** directory).
    2. Select each TRACE step in **StandPipeStream** and make sure their **Application** is set to a valid TRACE application.

       *The property view will display the current **Application** in red if the property is invalid for the local configuration.*
2. Expand the **Numerics** category in the Navigator.

   *Several sub-categories will be displayed, including **Reals**.*
3. Create a new component in the **Reals** category.

   *The **Select Type** dialog will be displayed, as shown below. This dialog is used to select the unit type associated with the new Real variable. Real variables can only be referenced from values with a matching unit type.*

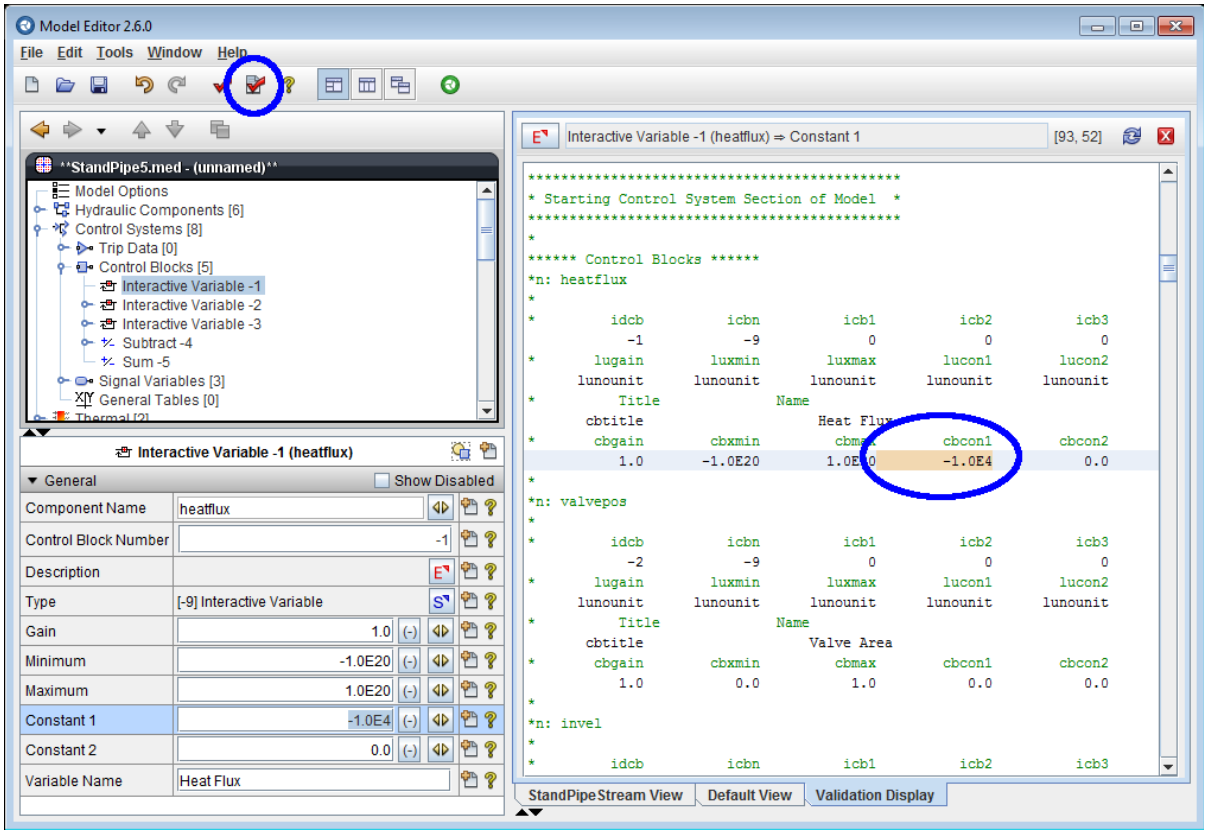


4. Select the "**No Unit (-)**" type, then press the **OK** button.

   *A new Real variable named **r1** is created and selected in the Navigator. Note the **Unknown** in the **Value** property: this variable has no value by default.*
5. Set the **Value** of the new variable to "**-1.0E4**".

6. Press the **Validation Display** ( ) button on the Model Editor's main toolbar.

   The SNAP Validation Display includes the contents of the current model in ASCII format and a toolbar that indicates the current cursor location and a brief description of the input at that location.



7. Scroll to Control Block -1, "heatflux" as shown above.

   Control Block -1 is the first component in the control blocks section.

8. Click on the value of the **Constant 1** ("cbcon1") property as shown above.

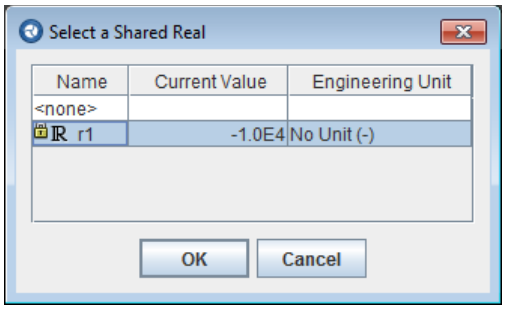
   *Clicking on a supported portion of the input will update the cursor location and enable the **Edit** toolbar button. Pressing the **Edit** button will open the graphical editor corresponding to the current cursor location. The same property selection and editing features are available in the ASCII Viewer for each component.*

9. Press the **Edit** button in the Validation Display toolbar.

   This will select the component (control block -1) in the Navigator and scroll the main Property View to the Constant 1 property, as shown above.

10. In the property editor for **Constant 1**, press the **Select Numeric Reference** button ( ).

    *The **Select a Shared Real** dialog will be displayed, as shown below.*



11. Select the row for **r1** and press the **OK** button.

    *The value for **Constant 1** will change to **r1(-1.0E4)**. This indicates that **Constant 1** is now referencing a real variable for its value. The value of the referenced variable will be substituted anywhere that the value of **Constant 1** is used within the model.*

The next several steps leverage the drawn variable functionality, which allows variables to be displayed and edited in a 2D View.

12. Open and display **Default View**, if it is not already open.
13. Drag **r1** from the Navigator into the View, directly over the **Heat Controller** area.

    *The value of **r1** will be displayed in the view. This representation is called a Drawn Variable.*

14. Select **r1** in the Navigator, and change its Value to "**-5.0E4**".

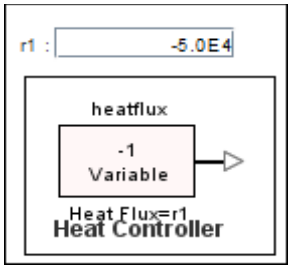
    *The value of **r1** displayed in the View will change to reflect the new variable value.*

15. Right-click the **r1** drawn variable in the View to display its pop-up menu.
16. In the menu, de-select the **Show Units** check-box

    *The **-** on the right side of the drawn variable will be hidden. Real variables with unit types other than **No Unit (-)** will display a more descriptive unit here.*

17. Right-click the **r1** drawn variable and select the **Editable** check-box

    The drawn variable will change from a label to a text field. The next several steps will demonstrate that the value of the variable can now be changed in the view.



18. Press the **Lock** button ( ) in the upper-left corner of the view to lock it.

    Look for a lock icon in the left-hand side of the view toolbar. If the icon is red and appears open ( ), then views are not locked. If it is green and appears closed ( ), then the views are locked.

    Locking a 2D View will prevent inadvertent repositioning of components and will activate any drawn variables in the view. Editable drawn variables can only be modified in a locked view.

19. Left-click within the field.

    *The drawn variable field will gain focus, and the **r1** variable will be selected in the Navigator.*

20. Type in the value "**-1.0E4**" and press the Enter key.

    *The **Value** attribute displayed in the Property View will also change to reflect the new variable value.*

The next several steps demonstrate python functions that can be used to modify variables with user-written code.

21. Create another **Real** variable, again with **No Unit (-)** for the type.

    *The new variable is automatically named **r2** and selected in the Navigator.*

22. Set the **Value** of the new variable to "**0.0**".

23. In the Property View, set the **Interactive Variable** value to "**False**".

    *The **Value** property becomes uneditable, and a new property, **Initial Value**, appears below. In addition, the lock icon next to **r2** in the Navigator disappears. These changes have several implications.*

    *The **Interactive Variable** property controls the context in which a variable's value can be modified. Interactive variables can be modified directly in the Navigator or in an editable drawn variable. Non-interactive variables can be modified by functions. The lock next an interactive variable is used to indicate that functions cannot modify its value.*

    *The **Initial Value** of a non-interactive variable is copied into the **Value** field before functions are executed. It serves as a stable starting value, allowing functions to reproduce their results with each successive evaluation.*

24. Create a new component in the **Functions** category.

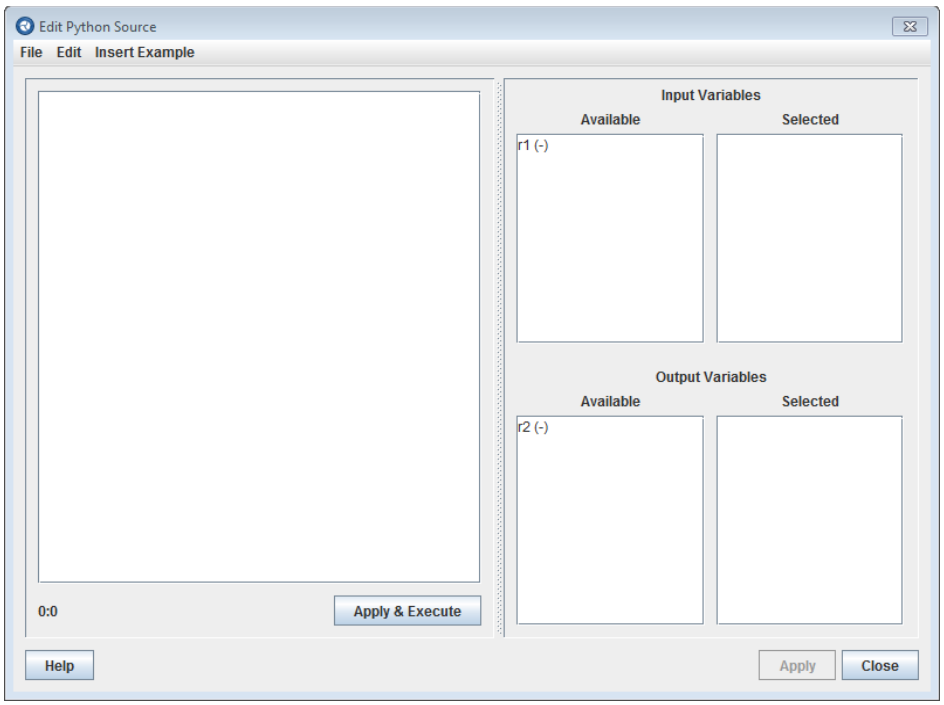
25. Select Python Function in the **Select Numeric Function Type** dialog and press the **OK** button.

    *A new function, **f1**, is created and selected in the Navigator. The function type is Python, meaning that the user-defined calculation is written in the Python programming language.*

    Note: SNAP supports the 2.5 Python standard via the Jython implementation. For more information, including tutorials for the Python programming language, see http://docs.python.org/release/2.5/ and http://www.jython.org/.

26. In the Property View, edit the **Python Source** property by pressing the **E** button in the editor.

    *The **External Function** window will open as shown below.*



    *The Python External Function window contains a python editing area (left) and a list input/output variables (right). The **Input Variables** indicate which variable values are available and which were retrieved by the function the last time it was executed(Selected). The **Output Variables** indicate which variables are available and which were modified by the function the last time it was executed (Selected).*

    The next several steps use this dialog to define the python function.

27. In the text area to the left, enter the following logic. Note that case is important: characters must be capitalized (or not) as shown below.

```
# Retrieves the value of variable r1
x = GetVariable( "r1" )
```

    *The first line is a comment. Anything on a line after the # character is ignored. The second line uses the custom GetVariable function to retrieve the value of the **r1** variable into a new variable named **x**. The value of **x** will be used in the next step.*

28. After the lines written above, add the following. While the case rule still applies, relative spacing must also be preserved.

```
# Init and set y
y = float('nan')
if x < -5.0e4:
    y = -1.0
elif x == -5.0e4:
    y = 0.0
else: # x > -5.0e4
    y = 1.0
```

*The first line after the comment declares and initializes a variable named **y** so that it can be used by the rest of the function. The following lines are very simple branching that set the value of **y** based on the value of **x**.*

The indents here are very important: Python defines blocks by spaces, instead of using explicit opening and closing characters (such as the curly-brace characters employed by C-based languages).

Note: the initial value of the **y** variable, **nan**, stands for *Not a Number*. This is used to indicate an invalid initial starting value that must be set explicitly, based on the state of **x**. Variables initialized to **nan** can often be useful, as all operations performed on **nan** also return **nan**, and all comparisons (except not-equal) return **false**. If the value was not assigned properly after the **nan** initialization, the value set to **r2** would show up as **Unknown**.

29. After the lines written above, add the following.

```
# Set variable r2 value to y
SetVariable( "r2", y )
```

*Similar to the first lines of code in the function, this custom function sets the value of the indicated variable to the specified value, in this case the variable **y**.*

30. Press the **Apply & Execute** button, then the **Close** button.

The logic defined in the last several steps will be stored and then executed. A dialog will briefly appear, indicating the evaluation of the function. On some machines, this may occur too quickly to notice.

Note: If an error console appears while executing the functions, go back to the Python code and make sure that it was entered exactly as listed above.

31. Select the **Functions** category.

The properties for this sub-category allow executing all functions, saving the calculated values, and setting whether functions are executed automatically (such as for each task in a parametric export) or explicitly (through the above button).

32. Press the **Execute Functions** button.

Again, a dialog will briefly appear, indicating the evaluation of the function.

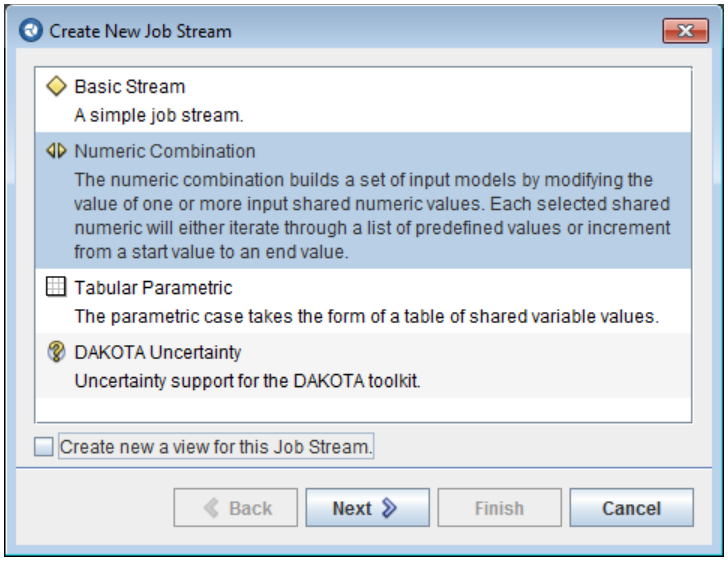
33. Select **r1** in the Navigator and note its **Value** of **-1.0E4**.
34. Select **r2** in the Navigator and note its **Value** of **1.0**.

*This is the expected value, given the logic entered into the function. As the value of **r1** was greater than -5.0e4 at the time of evaluation, the value for **r2** was set to 1.0. Later, when examining parametrics, the other values will be displayed.*

The following steps describe the basics of parametric streams. All streams have a type; only the Basic type has been demonstrated to this point. Other stream types are *parametric* types: they describe a type of stream where multiple variations of the base model are exported and run.

35. Right-click the **Job Streams** category and select **New** from the pop-up menu.

    *The **Select Stream Type** dialog will be displayed.*



36. De-select (un-check) the "**Create a new view for this Job Stream.**" check-box.

    An existing 2D View will be used to display the job stream.

37. Select the **Numeric Combination** item and press the **Next** button.

38. Select **"A Single Step TRACE Stream"** from the available options.

39. Press **Finish** to create the stream.

    *The new stream is created and selected in the Navigator. Note the different icon between **StandPipeStream** and the new stream. This indicates the type of individual streams.*

    *The **Parametric Properties** attribute is available in the new stream. This property is used to define all settings related to parametric runs, regardless of the stream type.*

40. Enter the following values for the new stream:

    Name: **ParametricStream**
    Platform: **Local**
    Root Folder: **runs**
    Relative Location: **TRACE/**

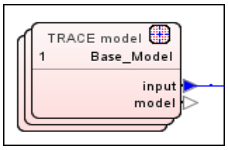
41. Expand **ParametricStream** and select the **Base_Job** step.

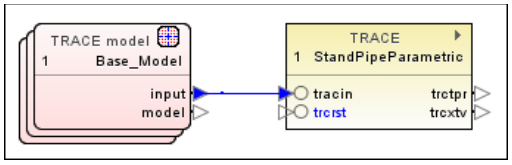
42. Name the step "**StandPipeParametric**".

43. Delete the **AptPlot 2** step.
44. Open and display **StandPipeStream View**.
45. From **ParametricStream** in the Navigator, drag both the model node and the TRACE step into the View.
46. Select ParametricStream model node, **TRACE model 1**, in the Navigator.

    *A new property, **Parametric**, will appear in the Property View.*
47. *Set the model node's **Parametric** property to "**True**".*

    Note that the model node for the parametric stream is rendered differently. This indicates a parametric model node that will export a set of inputs as part of a stream. The model node is currently red as the parametric properties have not been defined in the job stream.

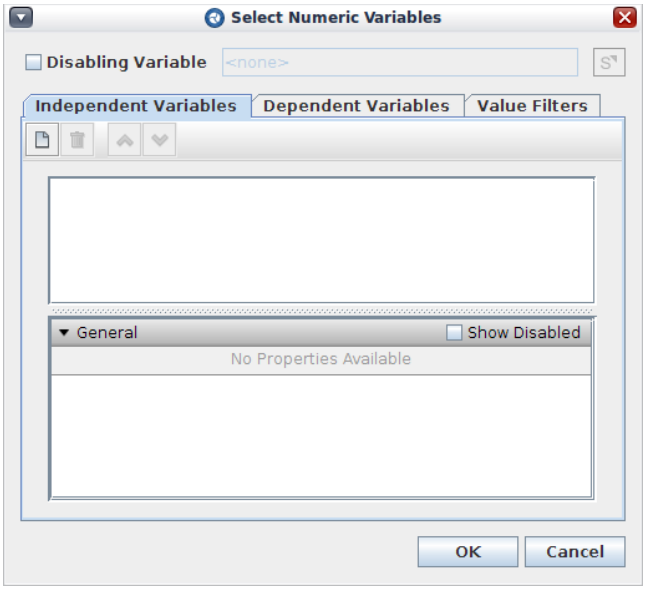




With the basics of the stream defined, the exercise will now turn to defining the parametric properties.

48. Select **ParametricStream** in the Navigator.
49. Edit the **Parametric Properties** by pressing the **E** button in the property editor.

    *The **Select Numeric Variables** dialog will appear, as shown below.*



50. In the **Independent Variables** tab, press the **New Variable** button ( ).

    *A **Select Variable** prompt is displayed.*

51. Select **r1** at the prompt and press the **OK** button.

    *A new row appears in the list for **r1** and its corresponding parametric properties are displayed below. The **r1** variable is now part of the* basis *for the numeric combination.*

52. In the parametric properties, set the following values:

    Use List: **False**
    Start Value: **-9.0e4**
    End Value: **-1.0e4**
    Increment: **2.0e4**

    *These properties define the range of values that will be substituted into the corresponding variable (**r1** in this case) during parametric submits. The values -1.0e4, -3.0e4, -5.0e4, -7.0e4, and -9.0e4 will be part of the sequence.*

    Note A Numeric Combination parametric stream creates a parametric iteration for each combination of the specified Independent Values. The current stream has five parametric iterations, one for each value in the sequence indicated above. Given another Independent Variable whose parametric properties indicated a range of three values, the number of parametric iterations would increase to 15. With only a single step in the stream (StandPipeParametric) this will result in 15 parametric tasks.

    When **Use List** is set to **True**, an explicit list of values can be defined.

53. Select the **Dependent Variables** tab.

    Note: The dependent variable tab does not have a property view. Unlike an independent variable, a dependent variable is not used to define the parameters of the combination. Instead, its value will be sampled and added to the list of parametric keywords associated with each task. Each set of keywords acts as a signature of sorts for an individual parametric task. The parametric keywords created by this stream will be shown in a later step.

54. Note the Value Filters tab. This tab allows filtering the generated task based on the values of the independent variables.

55. Add **r2** to the list of dependent variables. The steps are the same as those used to specify an independent variable.

56. Press the **OK** button.

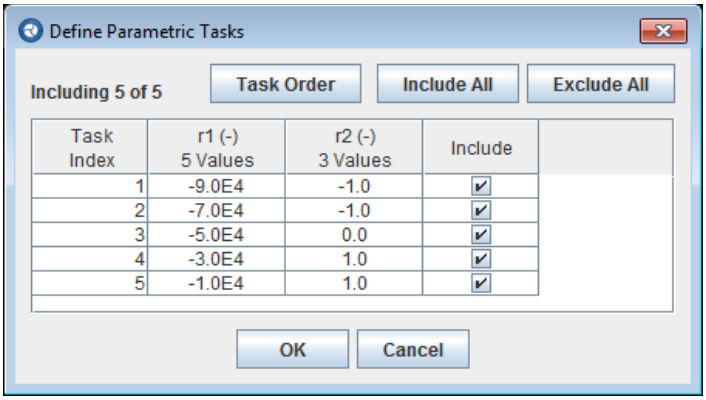
    The parametric properties are now set for the stream.

57. Select the TRACE step **StandPipeParametric** in the Navigator or in the View.

    *A new property, **Parametric Tasks**, will appear in the Property View. The value should indicate "**Including 5 of 5**".*

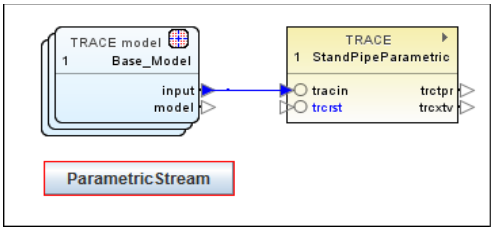
58. Edit the **Parametric Tasks** property by pressing the **E** button in the editor.

    *This will display the **Define Parametric Tasks** dialog, as shown below. This dialog illustrates the parametric tasks associated with a particular step, and can be used to disable specific tasks for both the step and any downstream steps.*



*This dialog illustrates the parametric keywords described earlier. The values of **r1** and **r2** in each row indicate the keyword signature for that task. Take specific note of the values for **r2**. Each reflects the state of **r2** after evaluating functions for the current parametric value of **r1** in that task.*

59. Press the **OK** button to close the dialog.
60. From the Navigator, drag ParametricStream into the **StandPipeStream View**, under the **StandPipeParametric** model node.

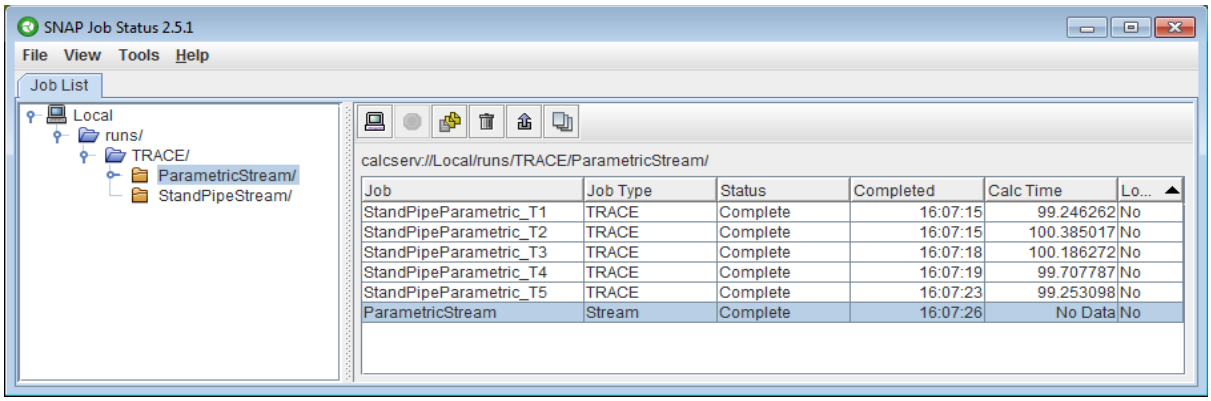


*A button titled **ParametricStream** is added to the View. This button can be used to submit the stream. Like other interactive controls, it can only be used when the View is locked.*

61. Lock the View and press the **ParametricStream** button.
62. Press **OK** to confirm the job stream submission.

    *After a moment, Job Status will appear and highlight the stream. The ParametricStream folder will contain five **TRACE** tasks: **StandPipeParametric_T1 - StandPipeParametric_T5**.*

63. Wait for the tasks to complete.
64. Select **StandPipeParametric_T1** in the table, right-click the row, then select **View Files → Text Files → tracin – StandPipeParametric_T1.inp**.



The Output Viewer will be displayed, showing the contents of the input file. Note the top of the input file includes a meta data block that shows the context for the parametric job stream.

```
*
*i: Numeric Combination of 1 variables.
*i: Independent Var        Type   Value  Range
*i:               r1  No Unit(-)  -9.0E4  [-9.0E4 / -1.0E4 / 2.0E4] [start / end / increment]
*i:
*i:   Dependent Var        Type   Value
*i:               r2  No Unit(-)   -1.0
*
```

65. Navigate to the **Control Blocks** section and in the first block, **heatflux**, take note of the **cbcon1** value of **-9.0E4**.
66. Select **StandPipeParametric_T5** in the table, right-click the row, then select **View Files → Text Files → tracin – StandPipeParametric_T5.inp**.
67. Once again, navigate to **heatflux** in the input and take note of the **cbcon1** value of **-1.0E4**.

    Note: The **cbcon1** value represents **Constant 1**, the value of the referenced variable **r1**. This replacement between **tracin** inputs demonstrates the changes made to the model for each parametric iteration.

68. Close Job Status.
69. Select **File → Close All** from the main menu.
70. Press the **Discard All** button.

Exercise 13. Animating a Model

This exercise introduces SNAP's post-processing capabilities. The following steps describe how to create an animation display using a standpipe model similar to those created in previous exercises.

1. Open **SNAP_Exercises/StandPipe6.med**, included with these exercises, and make the following modifications:
   (a) Select **StandPipeStream** in the Navigator and make sure its **Platform** is set to "**Local**" and its **Root Folder** to "**runs**" (or the appropriate root folder used in place of the **runs** directory).
   (b) Select each TRACE step in **StandPipeStream** and make sure their **Application** is set to a valid TRACE application.

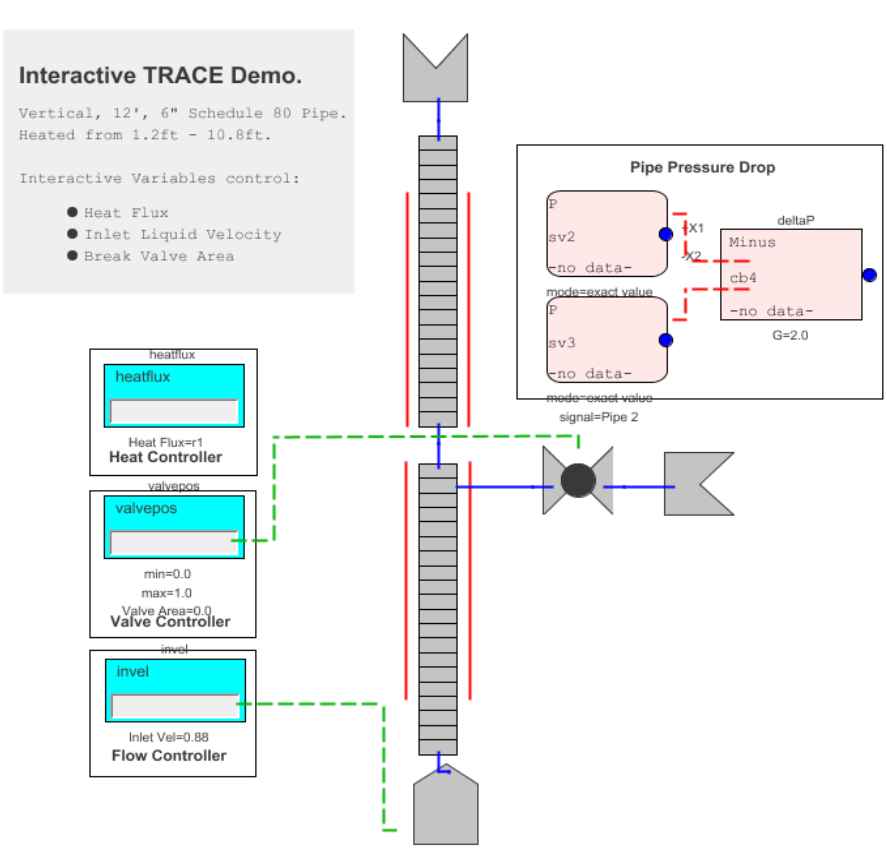
   *The property view will display the current **Application** in red if the property is invalid for the local configuration.*
   (c) Select the **StandPipeRestart** TRACE step and set the **Start Paused** option to **Off**.
2. If **Default View** is not open or displayed, open and select it now.
3. Unlock the View, if needed.
4. Select all of the components in the view: right-click on any empty space in the view and choose **Select → All** from the pop-up menu.

   If any components are selected before opening the pop-up menu, clear the selection by left-clicking on empty space in the view. Otherwise, the right-click pop-up will display entries for the selected item. Alternatively, the keyboard shortcut Ctrl-A when the 2D View has input focus will select all components (Command-A on a Mac).
5. With all components in the view selected, right-click the selection and select **Copy** from the pop-up menu.
6. Create a new **Animation** model by pressing the New Model button and selecting "Animation Model" in the **Select Model Type** dialog.

   *Once the new model is created and displayed in the Navigator, a blank **Default View** will be displayed.*
7. Right-click anywhere in the **Default View** and select **Paste** from the pop-up menu.

   This will paste the TRACE model drawings into the animation to create the initial animation view. The view should appear similar to the following image.

   The paste created a number of display elements in the view that are automatically connected to the appropriate data channels. Any number of additional items can be added to the view, such as data value readouts, interactive controls, strip charts, etc.. Additional animation elements will be added later in the exercise.



8. In the Navigator, switch back to **StandPipe6.med**.
9. Submit the **StandPipeStream** job stream and wait for the **StandPipeRestart** to complete.
10. Select the new unsaved Animation model in the Navigator.

In order to animate the view, connections must be established to the calculation submitted in the previous exercise. This will provide a working set of data channels and interactive variables for the calculation and simplify development of the interactive view.

11. In the Navigator, locate and expand the **Data Sources** node of the animation model.
12. Select the data source labeled "**Master (NewSource)**".
13. Press the **E** to the right of the **Source Run URL** property for this data source.

    *A **Select Data Source** window will appear, used to select the calculation that the animation model should connect to. This dialog is extremely similar to Job Status, providing a consistent interface for selecting jobs and viewing their properties.*
14. Expand **Local**, then **runs**, then **TRACE**, then select the **StandPipeStream** folder.

    A list of calculations is displayed on the right.

15. Select the **StandPipeRestart** job in the table and press the **OK** button.

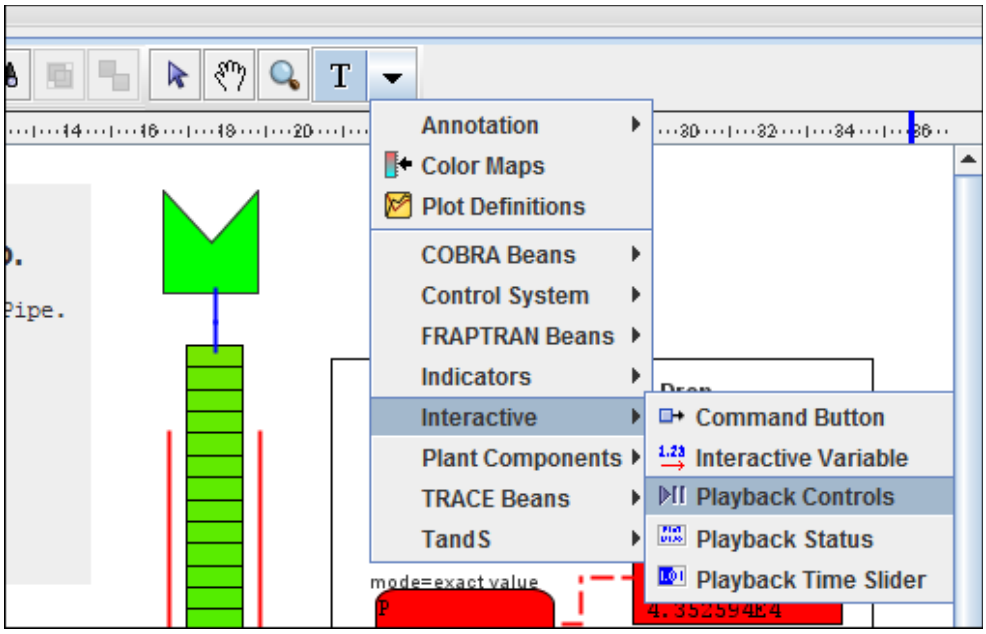
    The master data source is now associated with the completed job.

16. Press the **Connect to Data Sources** button ( ) located on the main toolbar to activate the connection.

    The animation model will now connect to the data source, providing it access to the calculation results. Notice that a number of buttons next to the Connect button are now available. These playback controls will be explained shortly.

With the animation model connected, a number of new features are available. The animation can "play back" the results of the run, modifying the display-based data channel values at a given point in time in the calculation. Before animating the results, the next several steps add additional display elements to the view.

17. Select **Interactive → Playback Controls** with the Insertion Tool.



18. Click on the view to place the interactive controls.

Note: The controls can be easily repositioned using the select tool after the initial placement.

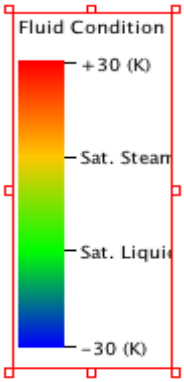
19. Expand the **Color Maps** category in the Navigator.

    SNAP uses Color Maps to translate component data and thermal-hydraulic conditions to color.

20. Select the **Fluid Condition Color Map** and examine its properties.

    This component provides a mapping of thermal hydraulic condition over the sub-cooled, saturated, and super-heated regions. The colors at the ends of each region are supplied

along with a pair of offsets to map temperatures below the saturation temperature (sub-cooled) and above the saturation temperature (super-heated).



21. In the Navigator, right-click the **Fluid Condition Color Map** and select **Add to View → Default View** from the pop-up menu.

    This will add the range to the view, which will appear similar to the image on the right. The range can be re-sized and repositioned as needed.

22. Using the Insertion Tool, add a **Data Value** (located under the **Indicators** menu) to the display and set its properties as follows:

    Channel Name: **time**
    Numerical Format: **%.2f**

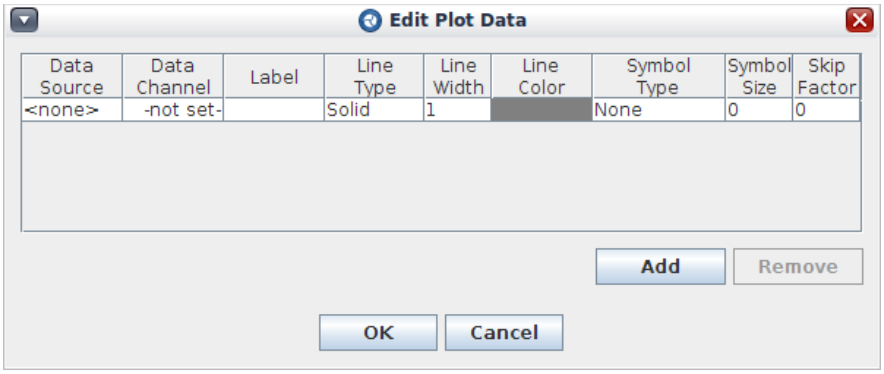
Note: Data values use the format string standard from the C programming language for all numerical formats.

23. Using the insertion tool, add a **Strip Plot** (located under the **Indicators** menu) to the display and set its properties as follows:

    Main Plot Title: **Temperature**
    Y Axis Number Format: **%.3e**

24. Edit the **Plot Data** property by pressing the **E** button in its property editor.

    *The **Edit Plot Data** dialog will be displayed, as shown below. This dialog is used to define the lines that appear on the strip plot. Each row in the table represents a line on the plot.*



25. Using the table cell editors, set the following values in the table row:

    Data Source: **Master (StandPipeRestart)**
    Data Channel: **tln-1A01**
    Label: **Pipe 1**
    Line Color: **255,0,0** in the **RGB** tab.

26. Press the **Add** button to add another line to the plot. Set its values as follows:

Data Source: **Master (StandPipeRestart)**
Data Channel: **tln-2A20**
Label: **Pipe 2**
Line Color: **90,90,255** in the **RGB** tab.

27. Press the **OK** button.
28. Press the **Play** button in the playback controls on the Main toolbar.

    When the animation begins you will notice elements along the pipe changing color to indicate the current fluid condition, a read-out of current time in the data value, and the temperature of the indicated cells is plotted in the strip plot.

29. Lock the animation using the **Lock** button.

    Locking an animation will activate any interactive controls contained in the view.

30. Press the **Rewind** button in the Playback Controls inside the view.
31. Press the **Clock** button in the Playback Controls inside the view.
32. Select the **Replay Proportional to Real Time** check-box and set the value to 1.0.
33. Press the **OK** button.
34. Press the **Play** button in the Playback Controls inside the view.

    Notice that the replay is now slowed down to display 1 second advancing for each second of real time. This can be very useful when replaying fast-running calculations.

35. Press the **Clock** button in the Playback Controls inside the view.
36. Un-check the **Replay Proportional to Real Time** check-box.

    Note: This property is saved as a user preference and must be deactivated for future exercises.

37. Press the **OK** button.

Optional Exercise: Add an Axial Plot bean to the view, which displays the heat structure's inside and outside temperature as two different data sets along the length of the heat structure.

Exercise 14. Interactive Controls

SNAP animation models can interface directly with interactive runs to manipulate the calculation during execution. Display beans can send commands to the job by setting interactive variables to preset or user-specified values.

The following steps introduce interactive controls in an animation model.

1. Open the sample file **StandPipe_Anim_interactive.med**, included with this exercise.

   The version of StandPipe_Anim.med used here is modified to include additional interactive controls and an axial void profile.

2. Ensure that the view is locked.

   The interactive controls demonstrated in this exercise only work in a locked view.

3. Open **SNAP_Exercises/StandPipe6.med**, included with these exercises, and make the following modifications:

   (a) Select **StandPipeStream** in the Navigator and make sure its **Platform** is set to "**Local**" and its **Root Folder** to "**runs**" (or the appropriate root folder used in place of the **runs** directory).

   (b) Select each TRACE step in **StandPipeStream** and make sure their **Application** is set to a valid TRACE application.

   *The property view will display the current **Application** in red if the property is invalid for the local configuration.*

4. Set the **Animation Model** to **StandPipe_Anim_interactive.med,** included with this exercise.
5. Under the **Cases** category, select **Restart Case**.
6. Edit the **Restart Model** property by pressing the **E** button in the editor.

   The virtual model will open so the restart model can be edited.

7. Select the **Model Options** node in the Navigator.
8. Edit the **Timestep Data** property by pressing the **E** button in the editor.
9. Change the **End Time** value to "**1.0E6**", then press the **OK** button.

   This sets an extremely long calculation time, so that the interactive commands can be demonstrated without the model running to completion.

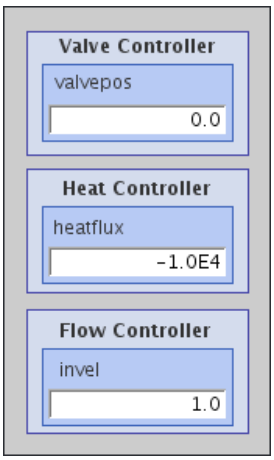
10. In the Restart Case panel, press the **Save Case** button ().
11. In the **StandPipeStream View**, or in the Navigator, select the **StandPipeRestart** step.

    This exercise will be animating the results of this step.

12. Submit the stream: right-click **StandPipeStream** in the Navigator and select **Submit Stream to Local** from the pop-up menu.

13. Press **OK** to confirm the submission.
14. Wait for the StandPipeRestart case to initialize and to Pause.
15. In the Model Editor, select the **File → Open** menu item.
16. Select the **StandPipe_Anim_interactive.med** model in the Navigator.
17. Select **Data Sources → Master** in the Navigator.
18. Set the **Source Run URL** to select the StandPipeRestart case that is currently paused and press the **OK** button.
19. Press the **Connect** button, then the **Play** button in the Main Toolbar.
20. Resume the calculation by selecting **Yes** at the prompt.

    *Take note of the text fields in the **Valve Controller**, **Heat Controller**, and **Flow Controller** panels. These fields are Interactive Variable beans. Typing a number into the field and pressing Enter sets the value of the interactive variable referenced by the bean.*



21. Enter "**-1.0E5**" in the **Heat Controller** interactive variable and observe the results.

    The void profile will shift in response to the change in power.
22. Repeat the last step with a value of "**-1.0E4**".
23. Enter "**1.0**" into the **Valve Controller** interactive variable.

    The valve on the left of the diagram will change from red to green to indicate that it opened.
24. Repeat the last step with a value of "**0.0**".

    The valve changes back to red, illustrating a close.
25. Enter "**5.0**" into the **Flow Controller** interactive variable and observe the results.

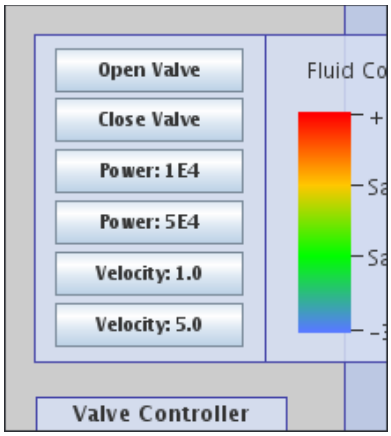
    The control block to the right of the field will slowly increase to the new flow value.
26. Repeat the last step with a value of "**1.0**".
27. Halt the job with the **Stop** button ( ).

    *Take note of the buttons to the left of the **Fluid Condition** color map. These are **Command Button** beans. When pressed, a command button issues one or more preset interactive commands.*
28. Switch back to the **StandPipe6.med** model.

29. Resubmit **StandPipeStream** and wait for the **Resume Calculation** prompt.
30. Resume the calculation and wait for the **Void Profile** to stabilize.
31. Press the **Power: 5E4** button and observe the results.

    *Pressing this button has the same effect as entering "**5.0E4**" in the **Power Controller** field.*
32. In sequence, press the **Power: 1E4**, **Open Valve**, **Close Valve**, **Velocity: 1.0**, and **Velocity: 5.0** buttons, observing the results of each.

    Each of these buttons matches the commands entered above and should create equivalent results.
33. Pause the job with the **Pause** button ( ).



The next steps will modify one of the existing interactive control buttons to change the command being sent to TRACE.

34. Unlock the view and select the **Power: 1E4** interactive control button in the view.
35. Open the **Commands** property editing dialog.

    Note: This dialog controls the interactive variable values that will be sent to TRACE when the button is pressed. Notice that the current value is set to -10,000 and the Command field is set to cb 1:[Heat Flux]. This indicates that the interactive control block 1 will be set to -1.0E4.
36. Change the **Value** field to -20000.
37. Press the **New** button.
38. Select the **Command** field for the new row.
39. Expand the provided drop-down list, and select **cb 2: [Valve Area]**.
40. Set the Value to 1.0 and press the **OK** button
41. Set the **Label** to "2E4 & Open".
42. Lock the view.
43. Press the **Close Valve** button, then the **Power: 5E4** button.
44. Resume the calculation.
45. Press the **2E4 & Open** button.

    Notice that the Valve indicator turns to green, the valvepos value is set to 1.0 and the heatflux value is set to -2.0E4.
46. Halt the calculation, close both models, and exit the Model Editor.

Exercise 15. AptPlot in Job Streams

In addition to the SNAP integration explored earlier, AptPlot can be used in job streams as an application to create complex plots.

The following steps create the AptPlot step that forms the basis of this exercise.

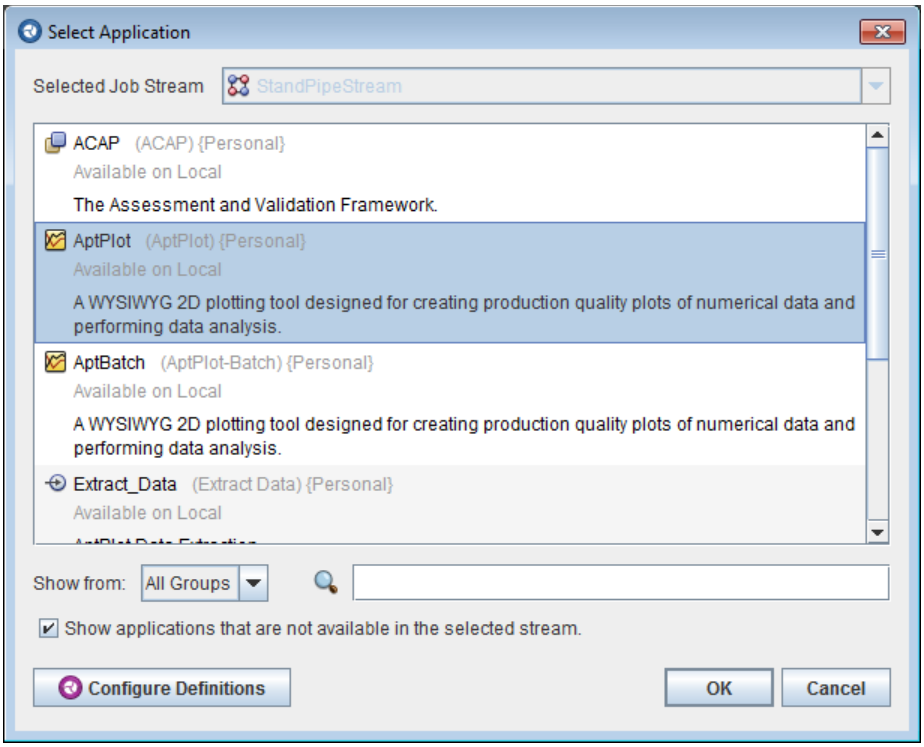
1. Open **SNAP_Exercises/StandPipe7.med**, included with these exercises, and make the following modifications:
    1. Select **StandPipeStream** in the Navigator and make sure its **Platform** is set to "**Local**" and its **Root Folder** to "**runs**" (or the appropriate root folder used in place of the **runs** directory).
    2. Select each TRACE step in **StandPipeStream** and make sure their **Application** is set to a valid TRACE application.

        *The property view will display the current **Application** in red if the property is invalid for the local configuration.*
2. Open the **StandPipeStream View** if it is not already open.
3. Expand the **Cases** category in the Navigator and select the available **Restart Case**.
4. Edit the **Restart Model** property by pressing the **E** button in the editor.

    The virtual model will open so the restart model can be edited.
5. Select the **Model Options** node in the Navigator.
6. Edit the **Timestep Data** property by pressing the **E** button in the editor.
7. Change the **End Time** value to "**1000.0**", then press the **OK** button.

    This shortens the calculation to a more manageable run-time.
8. In the Restart Case panel, press the **Save Case** button ().
9. Expand **StandPipeStream**.
10. Create a new **Stream Step**.

    *The **Select Applications** dialog will be displayed as shown below.*



11. Select the **AptPlot** application and press the **OK** button.

    A new AptPlot step will be created and selected in the Navigator.

12. Set the **Name** of the AptPlot step to **SamplePlot**.

13. Add the AptPlot step to the view by dragging it from the Navigator and releasing it to the right of the **StandPipeRestart** step.

    Note: The step has only one input: an optional input labeled **param**. Plot file inputs are not enabled by default. As AptPlot can open any number of files from a wide range of file types, an AptPlot step must define its inputs explicitly.

The next several steps add a parameter file to the stream and attach it to the AptPlot step. This parameter file will be used by default for all plots that do not explicitly define their own parameter file.

14. In the **Parameter File** editor, make sure the check-box is selected, then press the **S** button.

    A pop-up menu will appear, asking for the location of the parameter file. If compatible External Files in the stream were present, these would also be listed in the menu.

15. Press the **Select Local File** item in the pop-up menu.

    A file selector will appear.

16. Navigate to and select the "**SNAP_Exercises/twographs.par**" file included with this exercise.

17. Press the **Open** button in the file browser.

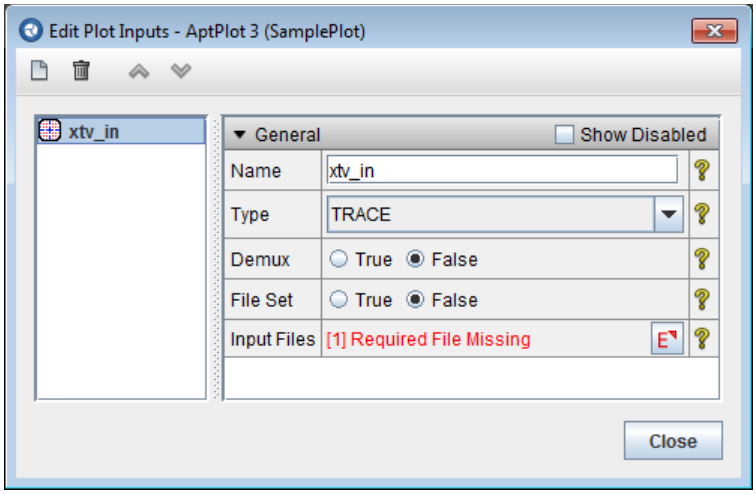
*A new External File has been created in the stream, representing the parameter file. In addition, this new file definition has been connected to the optional **param** input on the **SamplePlot** step.*

18. Expand the **Files** category under the **StandPipeStream**.
19. Drag **External File 1** from the Navigator into the view, just above the **StandPipeRestart** step.

The next several steps define the inputs to the **SamplePlot** step.

20. Select the **SamplePlot** node in the Navigator.
21. Press the **E** button in the **Plot Inputs** editor.

    *The **Edit Plot Inputs** dialog is displayed, as shown below. This dialog is used to define the files that the AptPlot step will use as sources of data.*



22. Press the **New Input** button ( ).

    A new input is added to the list and selected, displaying its properties to the right.

23. Set the properties of the new input as follows:

    Name: **xtv_in**
    Type: **TRACE**

    *Notice that a new input has appeared on the AptPlot step in the view, labeled **xtv_in**. Each input defined in the **Plot Inputs** editor defines another input for the step itself.*

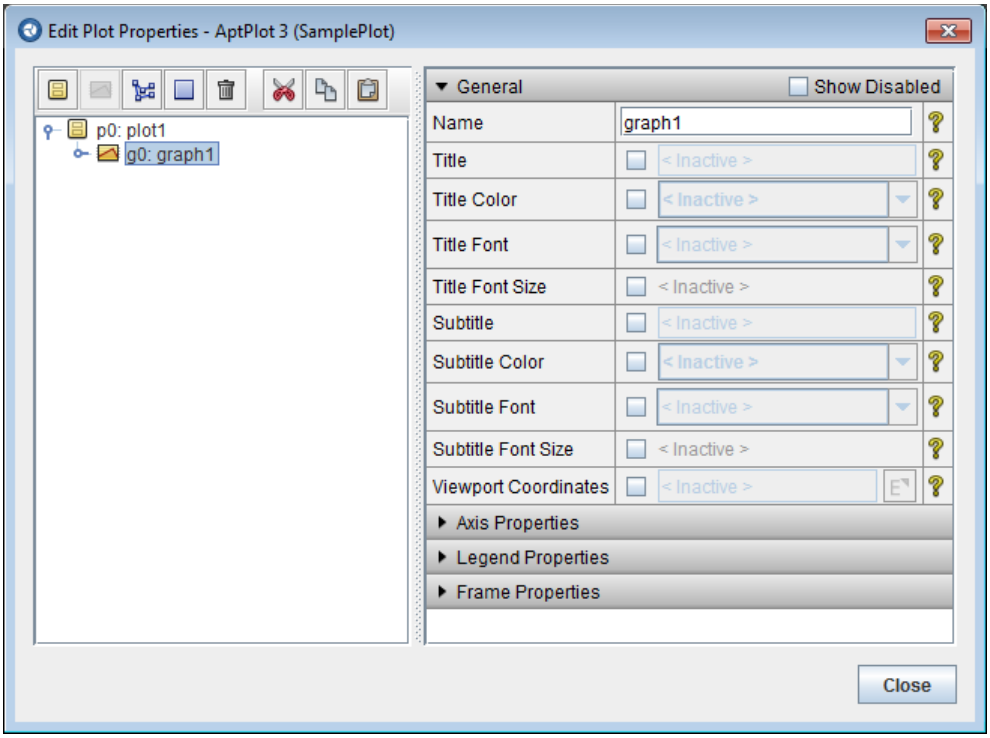
24. Press the **Close** button.
25. In the view, connect the **trcxtv** output of the **StandPipeRestart** step to the **xtv_in** input of the AptPlot step.

The following steps will define the contents of the plot generated by the step.

26. In the **SamplePlot** properties, press the **E** button in the **Plots** editor.

    *The **Edit Plot Properties** dialog is displayed, as shown below. This dialog is the heart of an AptPlot step definition: all plots produced by an AptPlot step are defined here. The*

tree on the left defines plots, graphs, data sets, and annotations that will be generated by the step. The properties on the right allow editing the selected definition.



27. Expand the **p0: plot1** node in the tree.

    *A default graph is included in the plot: **g0: graph1**. Where a plot is an entire AptPlot canvas, each graph is a specific segment of that canvas dedicated to displaying its underlying data sets.*

28. Select **g0: graph1**.

    The properties for the graph definition are shown to the right.

    Note: Almost all properties for the graph are optional properties: the property editor displays a check-box to the left used to activate the property. Unless activated, definition properties do not change the plot formatting. This is provided so that AptPlot steps have the flexibility to employ parameter files for general formatting, while step definitions override specific properties as needed. Alternatively, the entire plot can be defined in the AptPlot step without the use of a parameter file.

29. Set the following properties in **g0: graph1**:

    Title: **Sample Plot**
    Subtitle: **Created for SNAP Workshop**

30. With **g0: graph1** still selected, press the **New Data Set** button ( ).

    *A new set definition, **s0**, is created in graph **g0** and selected. Each data set is a collection of independent and dependent data retrieved from one of the AptPlot step's inputs.*

31. Set the following properties of **s0**:

    Input: **xtv_in**
    Dependent Data: **pn-2A01**
    Legend Entry: **Pressure (2A01)**

    *These settings indicate that the data set will retrieve its data from the TRACE XTV file connected to **xtv_in**, the **pn-2A01** time dependent data will compose the contents of the data set, and the specified text will appear for the data set in the legend.*

32. Right-click on **g0** and select **New Data Set** from the pop-up menu.

    *Another set definition, **s1**, is created in the graph.*

33. Set the following properties of **s1**:

    Input: **xtv_in**
    Dependent Data: **pn-2A02**
    Legend Entry: **Pressure (2A02)**

34. Select **p0** and press the **New Graph** button ().

    *A new graph, **g1**, is created in plot **p0**.*

35. Create a new data set in **g1** and set its properties as follows:

    Input: **xtv_in**
    Dependent Data: **rovn-2A01**
    Legend Entry: **Density (2A01)**

36. Press the **Close** button in the dialog.

The remaining steps define the output files that will be generated by the step, then submits the stream to the Local Calculation Server.

37. In the **SamplePlot** properties, press the **E** button in the **Plot Outputs** property editor.

38. Press the **New Output** button ().

    *A **Select Plot** prompt will be displayed, asking which plot in the step will be the basis of this output. All outputs on an AptPlot step are specific to a single plot. For images, such as the output that will be created by this exercise, this indicates which plot will be displayed in the generated image.*

39. Select **p0: plot1** in the Select Plot prompt, then press the **OK** button.

    The new output will be created and selected in the dialog.

40. Set the properties of the new output as follows:

    Name: **p0_image**
    Type: **PNG**

    *Notice that a new output has appeared on the AptPlot step in the view, labeled **p0_image**. Each output defined in the **Plot Outputs** editor defines another output for the step itself.*

41. Create another output with the following properties:

    Name: **p0_plot**
    Type: **AptPlot File**
    Plot: **p0: plot1**

42. Press the **Close** button.

    *The plot step is now setup to generate two output files: one PNG image of plot **p0**, and one APF (AptPlot's native file format) of **p0**.*

43. Select the **StandPipeRestart** step and make the following changes:
    1. Disable the **Animation Model** property.
    2. Change **Start Paused** to "**Off**".
44. Submit **StandPipeStream**: right click it in the Navigator and select **Submit Stream to Local** from the pop-up menu.
45. Press **OK** to confirm the submission.

    The stream will be submitted and Job Status will be displayed after a few seconds.

46. Wait for the **StandPipe**, **StandPipeRestart**, and **SamplePlot** tasks to complete.
47. Select **SamplePlot** in the table.
48. Press the **View Files** button () in the Job List toolbar.

    A pop-up menu appears. This menu is used to open files associated with a task.

49. Select **Images → p0_image_png** from the pop-up menu.

    An image of the plot will appear in the system's native image viewer.

50. Launch AptPlot: from the Windows Start menu, select **All Programs → Plotting Tools → AptPlot**.
51. In AptPlot's main menu, select **File → Open**.

    A file browser will appear.

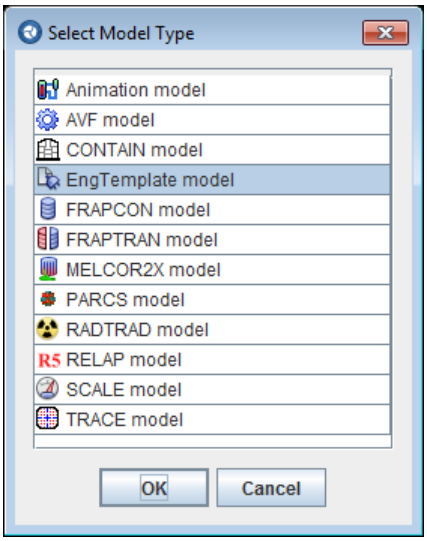
52. Navigate to the directory in which the stream was submitted.
53. Open the **StandPipeStream** folder, then the **SamplePlot** folder, and finally select the **p0_plot.apf** file.
54. Press the **Open** button.

    The plot will be recreated in AptPlot.

55. Close AptPlot, Job Status, and the Model Editor.

Exercise 16. Tabular Parametric and Axial Plotting

This exercise will illustrate some of the features provided by Engineering Template models, the Tabular Parametric stream type, and the AptPlot job step.



1. Open the Model Editor, if it is not already open.
2. Press the "**Create a new model**" button.
3. Select "**EngTemplate model**" and press **OK**.

   A new Engineering Template model will be created and added to the Navigator.

   An Engineering Template model allows the user to interact with multiple models and potentially multiple analysis codes, simultaneously. This interaction can take the form of a high-level controlled single model, multiple models that interact with one another, or separate cases that run independently but are ultimately compared to one another.
4. Save the new model as "**Standpipe_Template.med**" in a temporary location by using the **File → Save** menu item.
5. Select the **Model Options** node in the Navigator.
6. Set the **Title** to "SP Template".
7. Select the "**Reference Models**" category node in the Navigator.
8. Right-click on the "**Reference Models**" node and select the **New** option from the right-click pop-up menu.

   Note: Engineering templates reference models through Reference Model components. These components reference models by selecting the Model Editor Document (MED) files in which they are contained. The models selected as model references are referred to as "underlying models".
9. Select "**TRACE model**" from the list of available model types and Press **OK**.

   A new Reference Model component with the name "Model Reference 1" will be added to the Navigator and selected.
10. Press the **E** button in the **Input Model** property editor.
11. Open the "Underlying Standpipe" model provided with this exercise.

    *This file is located at: "**SNAP_Exercises/Underlying_StandPipe.med**"*

    A dialog will appear indicating that units selected for the Engineering Template model (SI) and the underlying model (British) do not match.

12. Press **Yes** to switch the Engineering Template model to British units.

    Note: The Engineering Template model can use either British or SI display units. This setting does not affect the units selected by underlying models. All values displayed and/or edited in an engineering template are automatically converted to the appropriate units when they are passed to an underlying model.

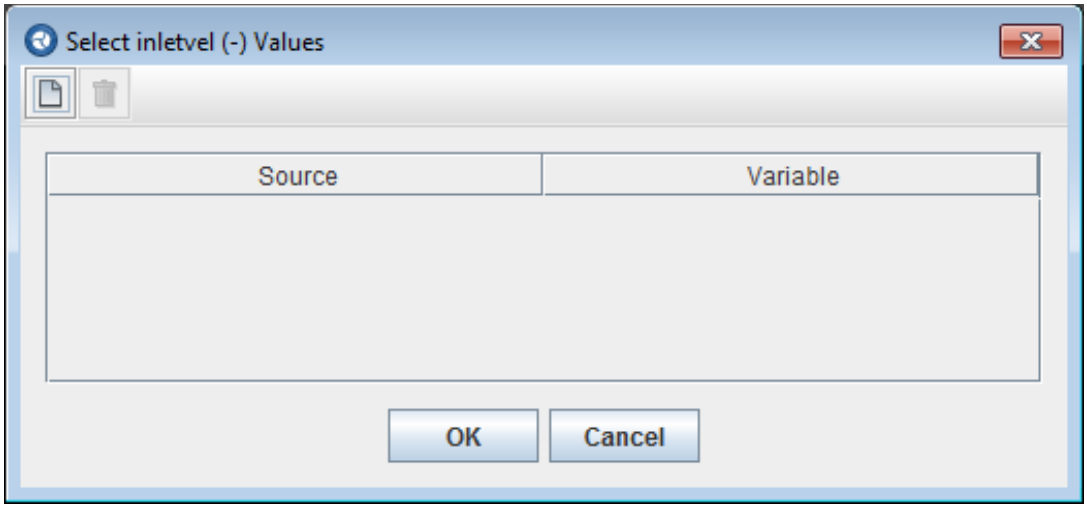
The next set of steps creates a pair of global variables that will be used to modify the underlying model.

13. Expand the "**Global Variables → Global Reals**" category node in the Navigator.
14. Create a new Global Real by right-clicking the "**Global Reals**" node and selecting **New** from the pop-up menu.
15. Select the "**No Unit (-)**" unit type from the list and press **OK**.

    This will create a new global real using the "No Unit" unit type and select it in the Navigator.

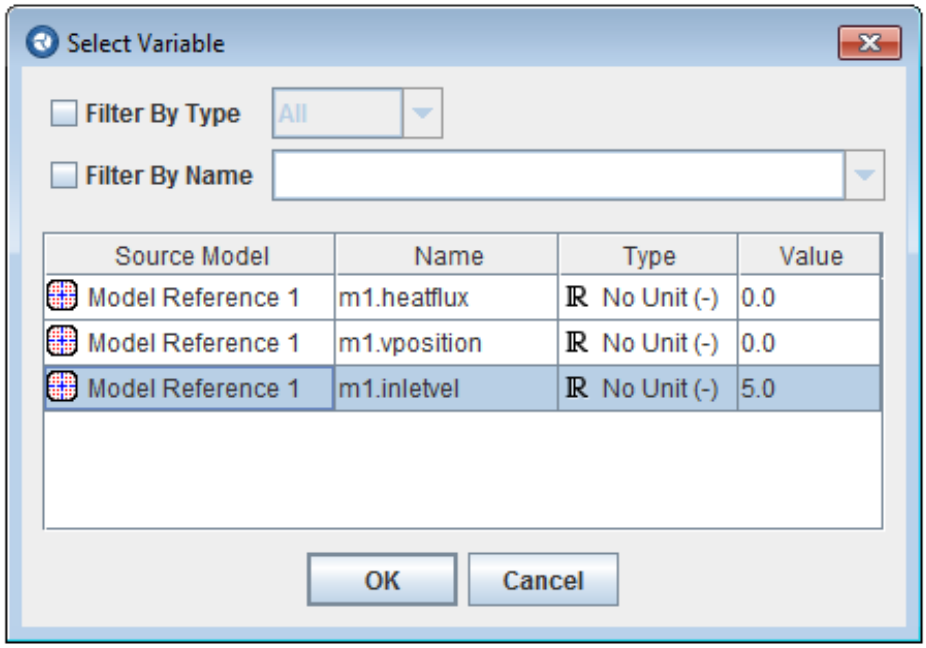
16. Set the **Name** property to "inletvel".
17. Press the **E** button in the **Reference Variables** property editor.

    This will open the "Select inletvel Values" dialog to allow references to be made to variables in the underlying model.



18. Press the **New** ( ) button on the toolbar.

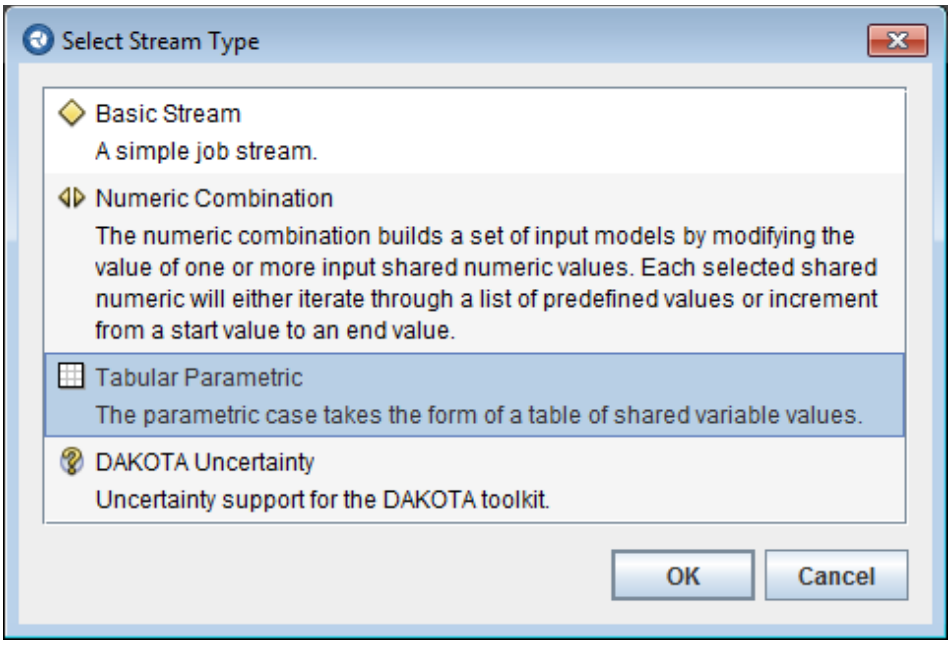
19. Select “m1.inletvel” from the list.



20. Press **OK** to select the variable.
21. Press **OK** to close the “Select Values” dialog.
22. Create another new Global Real by right-clicking the “**Global Reals**” node and selecting **New** from the pop-up menu.
23. Select the “**No Unit (-)**” unit type from the list and press **OK**.
24. Set the **Name** property to “valvepos”.
25. Press the **E** button in the **Reference Variables** property editor.
26. Press the **New** button on the toolbar.
27. Select “m1.vposition” from the list.
28. Press **OK** to select the variable.
29. Press **OK** to close the “Select Values” dialog.

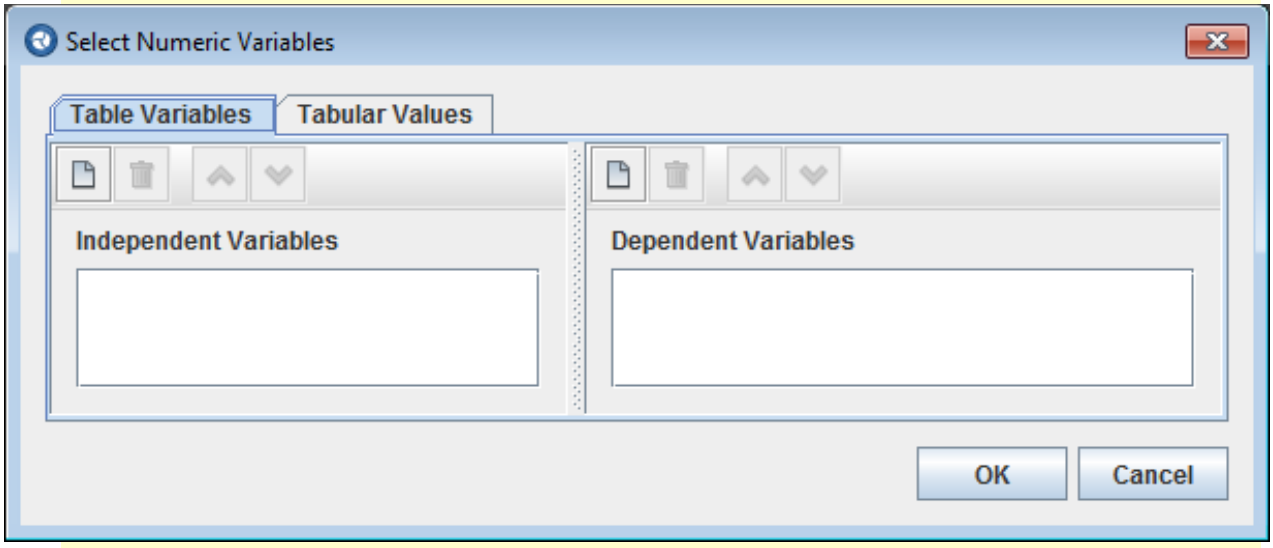
This set of steps will create a new tabular parametric job stream. Tabular Parametric streams use a specialized stream type that defines parametric iterations using a table of variable values.

30. Create a new Job Stream by right-clicking the “Job Streams” node and selecting **New** from the pop-up menu.
31. Select the **Tabular Parametric** stream type and press **OK**.



32. Press **Finish** to close the Create New Job Stream dialog.
33. Set the **Name** of the new job stream to "Template_Stream".
34. Press the **E** button in the **Parametric Properties** property editor.

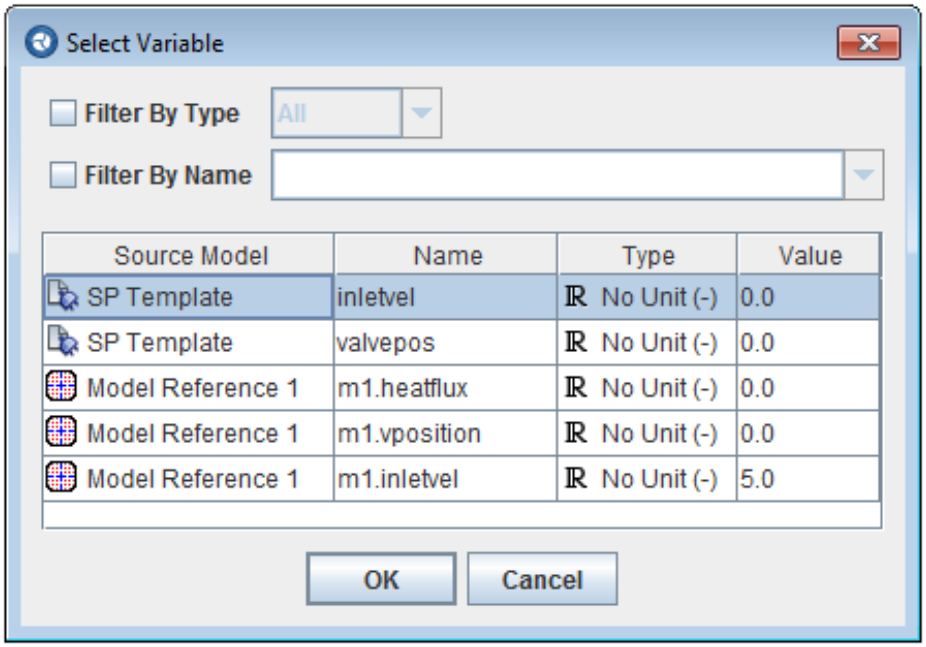
    This will open the Select Numeric Variables dialog used to configure the tabular parametric stream type.



Note: Independent variables are the variables that will be modified by each row of the tabular parametric to create the resulting parametric iterations. Independent variables can be added, removed, and re-ordered, in the Table Variables tab using the tool-bar provided on the left side of the dialog.

Dependent variables are additional variables that will be included in the parametric keywords but will not be modified directly. These variables can be added, removed, and re-ordered, using the tool-bar provided on the right side of the dialog.

35. Press the Independent Variables **New** button (on the left) to add a new independent variable.



Notice that there are two variables with the name "inletvel" included in the list. The first is a global variable in the engineering template (StandPipe_Template.med). The second is an underlying variable (Underlying_Standpipe.med). The Source Model column is included in the list for each variable to ensure that the correct variable can be identified and selected.

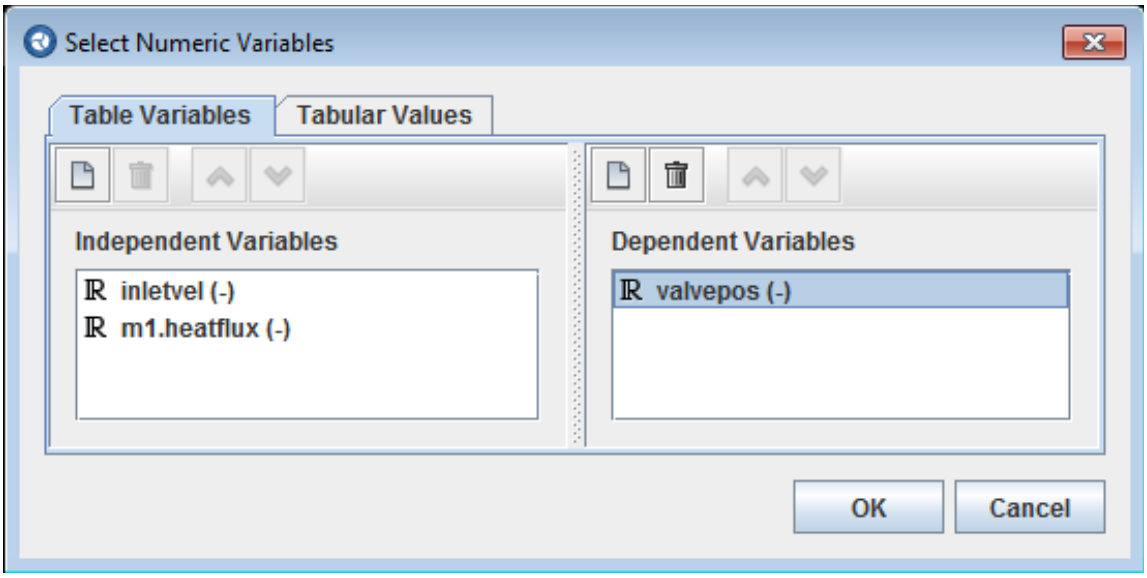
36. Select the global variable "**inletvel**" from the list of variables.
37. Press **OK** to select the variable.

    This will add the global real variable "inletvel" to the list of independent variables.

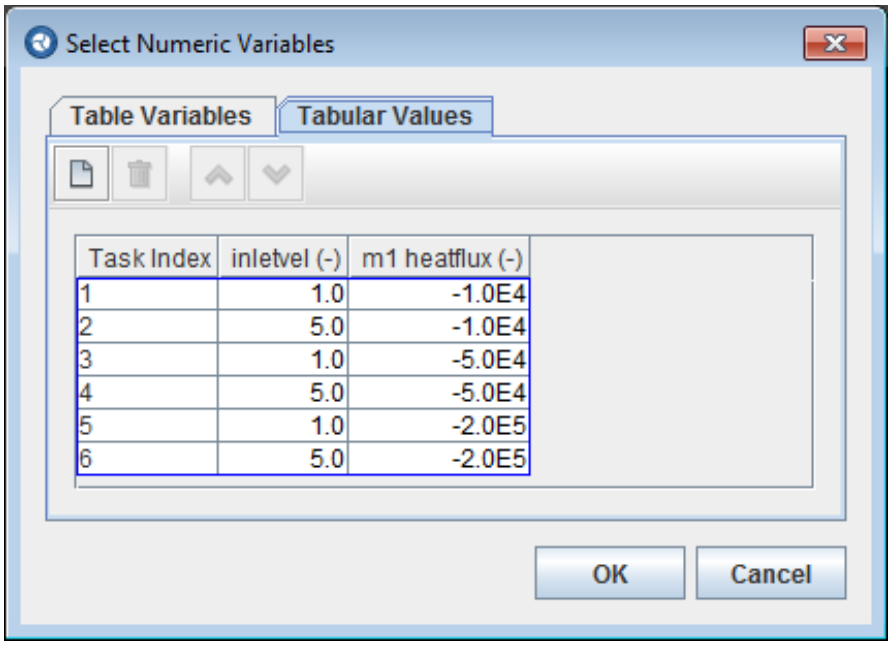
38. Press the **New** button again to add a second independent variable.
39. Select "m1.heatflux" from the list of variables.
40. Press **OK** to select the variable.

    Notice that the selected variable is displayed in the list of independent variables as "m1.heatflux". The "m1" indicates that the variable is a variable in the first underlying model.

41. Press the Dependent Variables **New** button to add a dependent variable.



42. Select “valvepos” from the list of variables.
43. Press **OK** to select the variable.



44. Select the **Tabular Values** tab.

Note: The table data that makes up the tabular parametric is located in the Tabular Values tab. The table contains a column for task index and a column for each of the independent variables selected in the Table variables table. The tool-bar at the top of the tab includes buttons for creating new rows, removing rows, and reordering rows.

45. Press the **New** button six times to add six new rows to the table.

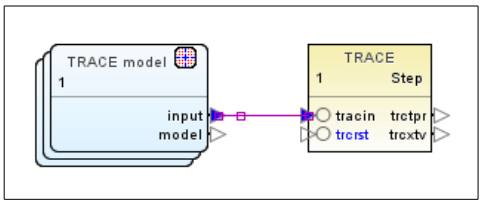
46. Enter the following inletvel and m1.heatflux values.

Task Index	inletvel (-)	m1.heatflux (-)
1	1.0	-1.0E4
2	5.0	-1.0E4
3	1.0	-5.0E4
4	5.0	-5.0E4
5	1.0	-2.0E5
6	5.0	-2.0E5

47. Press **OK** to complete the tabular parametric properties.
48. Expand the "**Job Streams → Template_Stream → Model Nodes**" node in the Navigator.
49. Select the "**TRACE model 1**" node in the Navigator.
50. Set the **Parametric** property to "True".
51. Drag the "**TRACE model 1**" node from the Navigator directly into the "**Default View**".

    This will create a representation of the model node in the view that can be connected to other job stream elements.
52. Create a new job step by right-clicking the "**Stream Steps**" node and selecting **New** from the pop-up menu.
53. Select **TRACE** from the list of applications and press the **OK** button.

    This will create a new TRACE job step and select it in the Navigator.
54. Set the **Name** property to "StandPipe".
55. Set the **Interactive** property to "On".
56. Drag the "**StandPipe**" job step node from the Navigator into the "**Default View**".



57. In the "**Default View**", use the Connect Tool to connect the "**input**" connection point of the TRACE model node to the "**tracin**" connection point of the "**StandPipe**" step.

The next set of steps will describe how to use the AptPlot step to create an axial plot of the void in the standpipe for each of the six TRACE executions in the job stream.

58. Create a new job step by right-clicking the "Stream Steps" node and selecting **New** from the pop-up menu.

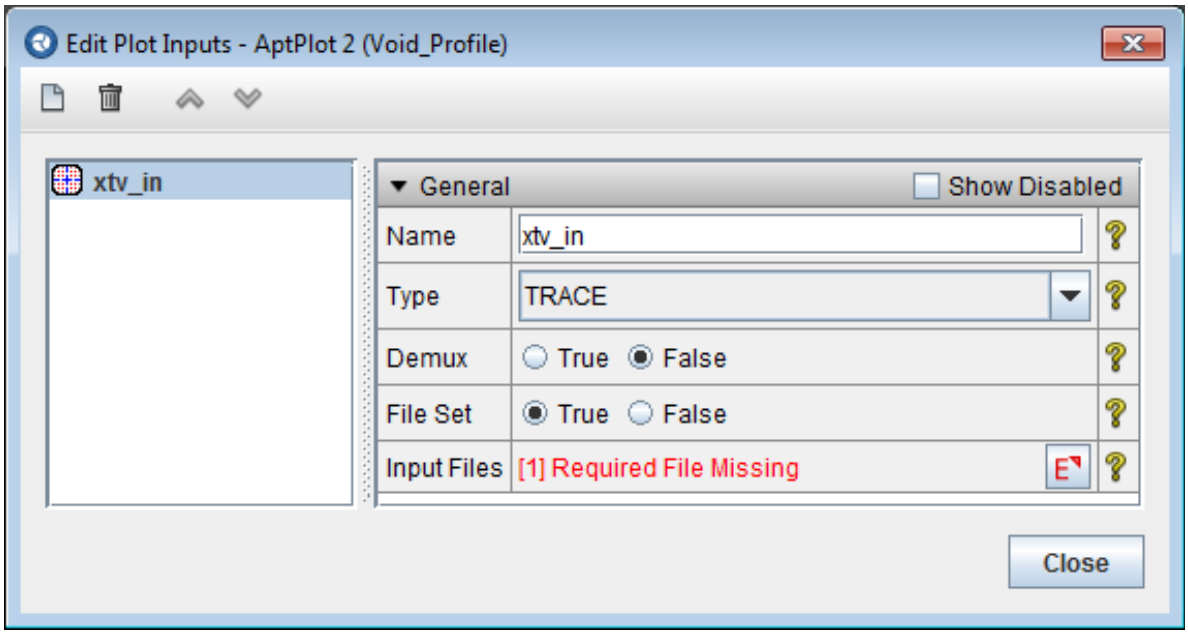
59. Select **AptPlot** from the list of applications and press the **OK** button.

    Note: The AptPlot step is a specialized job step included with the Model Editor that can specify any number of plots, all of which are generated each time its parent stream is run.

60. Set the **Name** property to "Void_Profile".

61. Press the **E** button in the **Plot Inputs** property editor.

    This will open the plot inputs dialog shown below.



62. Press the **New** button to create a new plot file input.

63. Set the **Name** of the new input to "xtv_in".

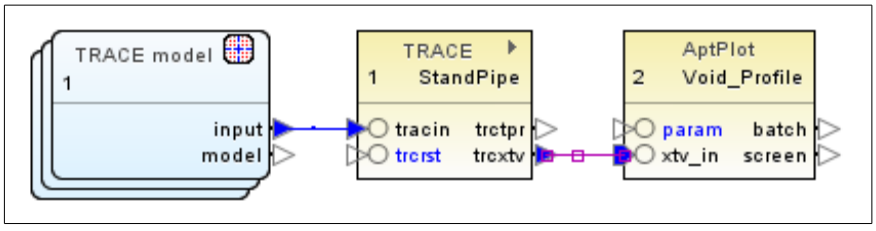
64. Set the **Type** to "TRACE".

65. Set the **File Set** property to "True".

66. Press the **Close** button to close the dialog.

67. Drag the "**Void_Profile**" job step node from the Navigator into the "**Default View**".

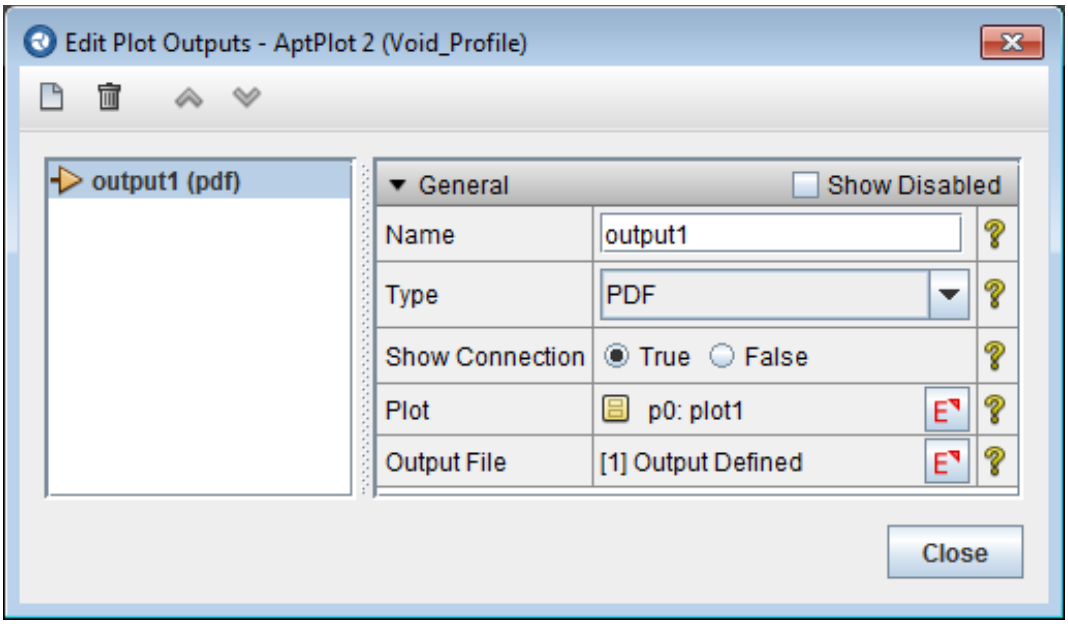
68. In the "**Default View**", use the Connect Tool to connect the "**trcxtv**" connection point of the "**StandPipe**" step to the "**xtv_in**" connection point of the "**Void_Profile**" step.



69. Select the "**Void_Profile**" step in the Navigator or the view.

70. Press the **E** button in the **Plot Outputs** property editor.

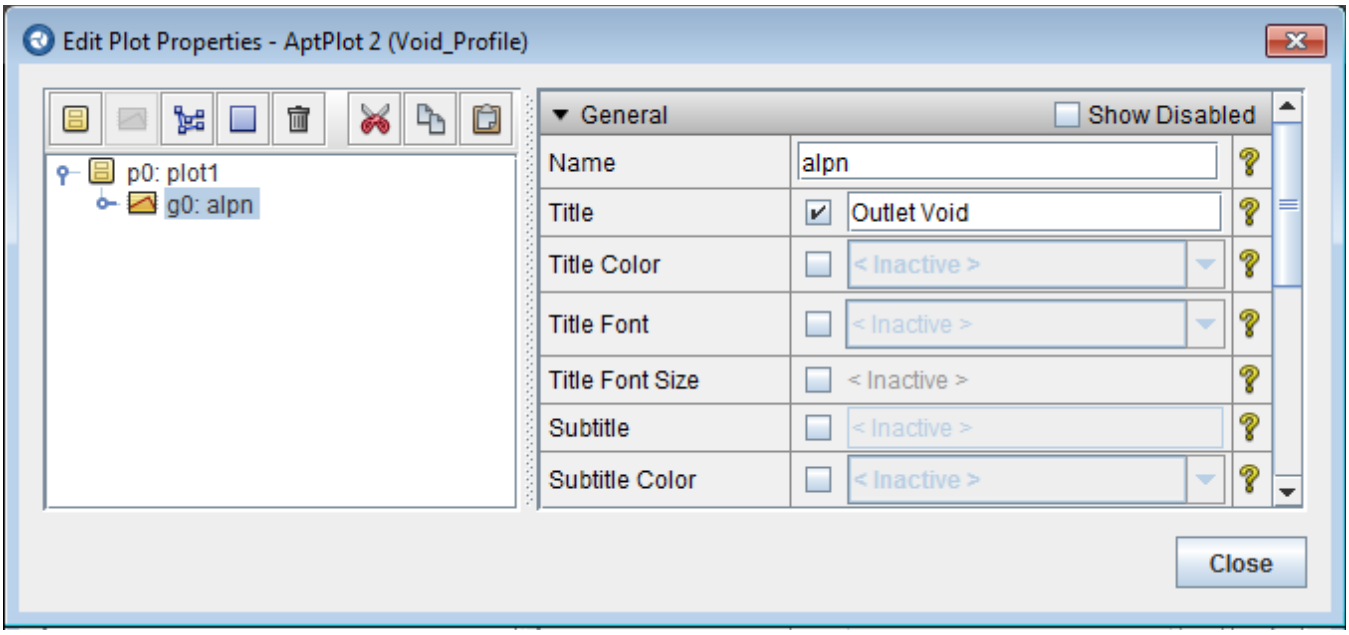
71. Press the **New** button to create a new plot output.



72. Select the "**p0: plot1**" plot definition and press **OK**.
73. Set the output **Name** to "void_profile".
74. Set the output **Type** to "PDF".
75. Press **Close** to close the plot outputs window.
76. Press the **E** button in the **Plots** property editor.

    This will open the Plot Properties dialog. This editing dialog is the heart of the AptPlot step. Within it, the plots, graphs, data sets, and annotations created by the step are added, edited, and removed. Note that this dialog is non-modal; other dialogs and Model Editor functions can be used while this dialog is open. Changes made in this dialog can be undone and redone with the standard undo/redo buttons and menu items.

    The tool-bar over the tree structure can be used to add or remove plots, graphs, data sets, and annotations. Graphs can only be added when the parent plot is selected; data sets and annotations can only be added when the parent graph is selected. Additionally, cut, copy, and paste actions are available on the right side of the tool-bar. All of these actions are also available from the right-click pop-up menu of the various entries in the tree.

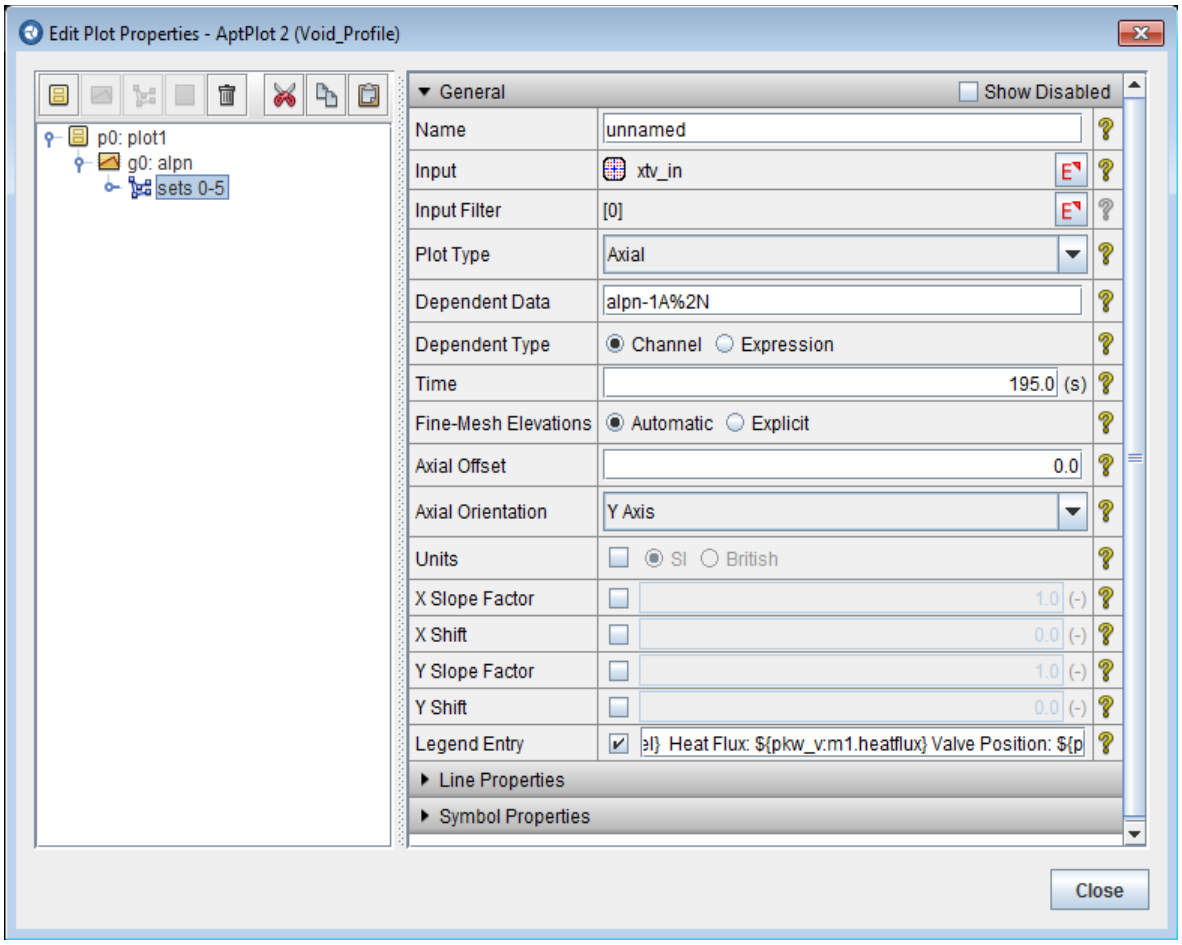


77. Select the "**g0: graph1**" graph node.
78. Set the following **g0** properties:

Name:	alpn
Title:	Outlet Void
X Axis Label:	Void
X Axis Scaling:	Explicit
X Axis Bounds:	[-0.1, 1.1]
Y Axis Label:	Elevation (m)
Legend Coordinates:	[0.65, 0.35]
Legend Text Font Size:	50

79. Press the **New Set** ( ) button to create a new data set.

    This will create a new set "s0" and select it, displaying its properties to the right.



80. Press the **E** button in the **Input** property editor.

    This will open the Select Input dialog to allow a plot input to be selected for the data set.

81. Select "xtv_in" and press OK. Set the following **sets 0-5** properties:

    - **Plot Type**: Axial
    - **Dependent Data**: alpn-1A%2N
    - **Time**: 195.0 (s)
    - **Fine-Mesh Elevations:** Automatic
    - **Axial Orientation:** **Y** Axis

82. Set the **Legend Entry** property to the following single line:

    Inlet Velocity: ${pkw_v:inletvel} Heat Flux: ${pkw_v:m1.heatflux}
    Valve Position: ${pkw_v:valvepos}

    This will replace the text in the legend entry representing each line on the plot. The portions of the legend that are enclosed in dollar braces "${" and brace "}" are called "tokens." These tokens will be replaced with the token value when the job stream is

*submitted. Note that the heat flux variable is set to "m1.heatflux". "m1" is the **Short Name** property of the reference model that is the source of the "heatflux" variable.*

This legend entry uses three "Parametric Keyword" tokens (${pkw_v: <keyword>}) to make it clear which velocity, heat flux, and valve position values are represented by which void profile line.

83. Expand the **sets 0-5** node.

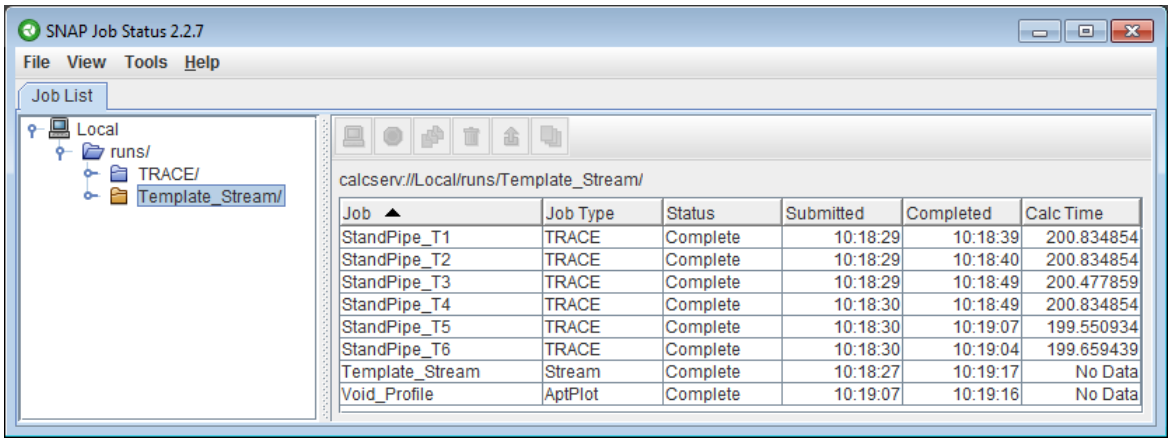
    *Note that the **sets 0-5** node has 5 children (s0 through s5).*

84. Select the first child node: **s0**

    *The first property of s0, **Keywords**, is the list of parametric keywords that were used to automatically create this set as a result of the tabular parametric values. Pressing the View (🔍) button will show the list of keywords and values in an easy to read pop-up.*

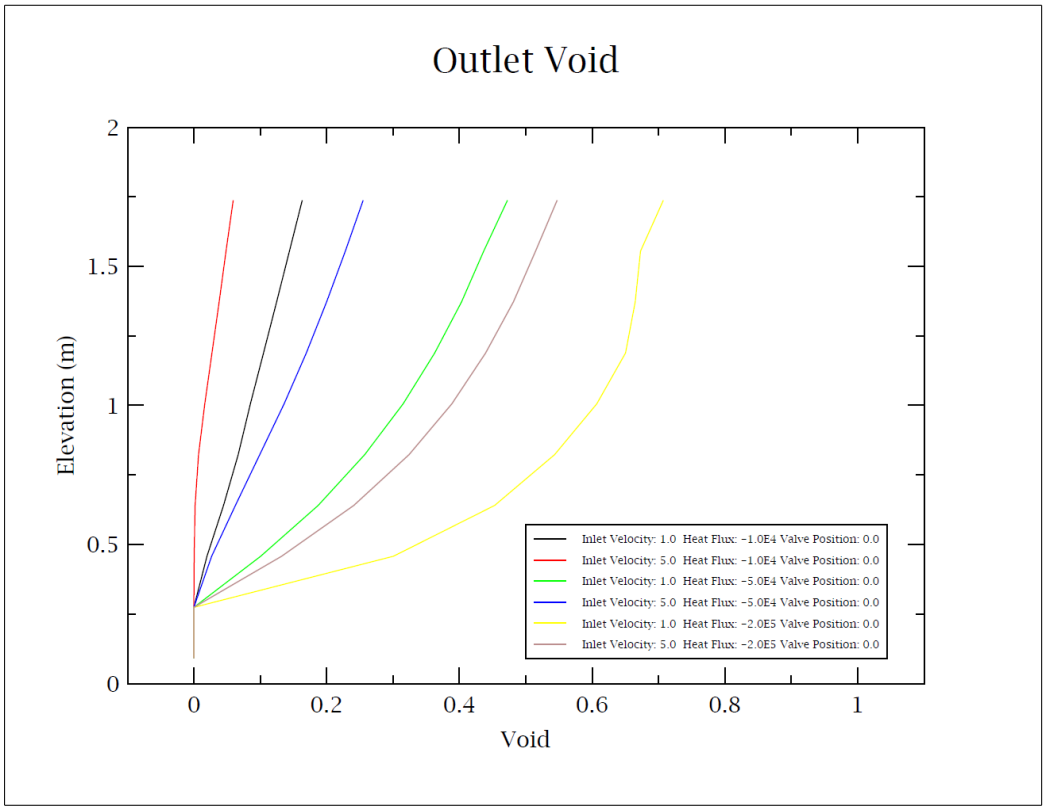
85. Press the **Close** button to close the plot propertied dialog.
86. Select the "**Template_Stream**" node in the Navigator.
87. Drag the "**Template_Stream**" node into the Default View.
88. Press the **Lock** (🔓) button to lock the view.
89. Press the **Template_Stream** button in the Default View to submit the stream.
90. Press **OK** to confirm the submission.

    This will submit the stream and open Job Status automatically.



91. Wait for the stream to complete.
92. Select the **Void_Profile** row in the table.
93. Press the **View Files** (🗐) button to open the files pop-up menu.
94. Select the "**PDF Documents → void_profile.pdf**" item in the pop-up menu.

    This will open the void profile PDF file in the system's default PDF viewer. Note the gradual void increase along the length of the pipe.



95. Close the PDF viewer.

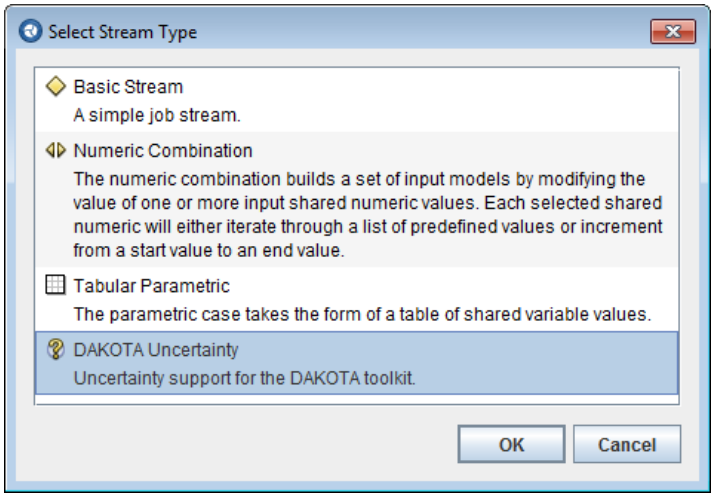
Exercise 17. Uncertainty Quantification with TRACE and DAKOTA

This exercise introduces the DAKOTA Uncertainty Quantification (UQ) job stream type, and DAKOTA stream step. After this exercise is complete the analyst will be familiar with defining the parameters of a UQ calculation in a TRACE model using sensitivity coefficients, extracting figures of merit from resulting plot files, and the DAKOTA report generation.

This exercise will take a null-transient W4Loop model and perform an uncertainty quantification on the fuel gap gas conductivity factor.

1. Open **SNAP_Exercises/UQ_W4Loop.med**, included with these exercises.
2. Open the **UQ Stream** view if it is not already visible.
3. Select "**Job Streams → UQ_Stream"** in the Navigator.

   This job stream is currently set to a Basic Stream. In the next steps, the stream type will be set to DAKTOA Uncertainty and the uncertainty parameters will be initialized.



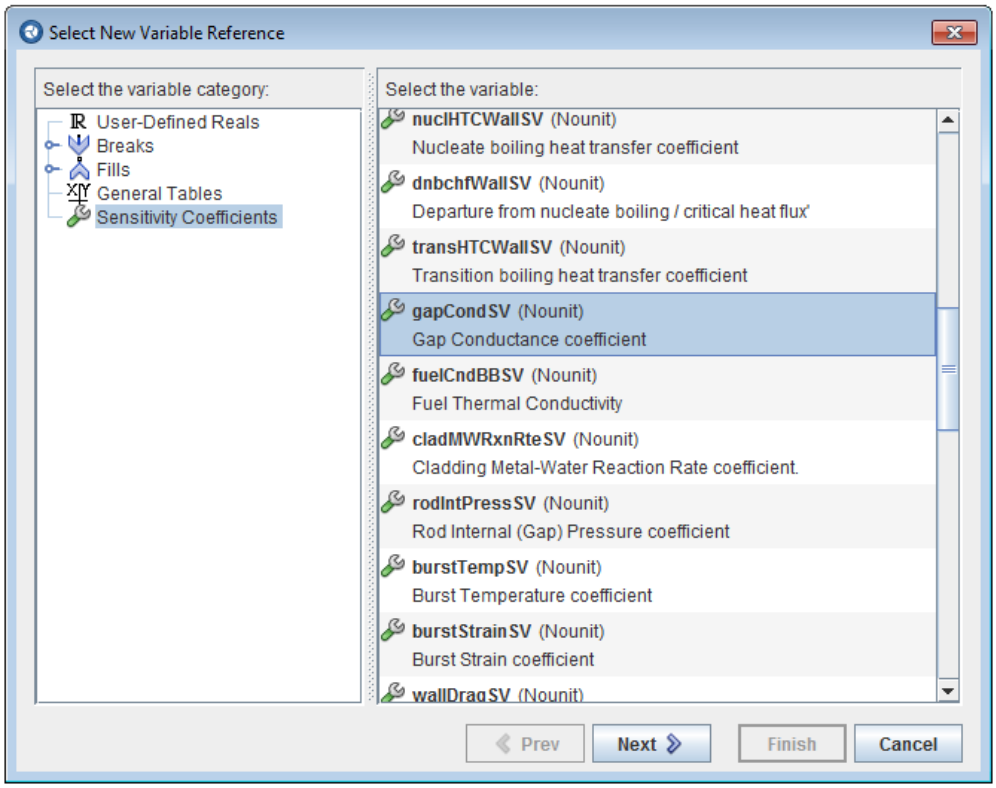
4. Open the **Stream Type** property editing dialog.
5. Select the **DAKOTA Uncertainty** stream type and press the **OK** button.
6. Open the **Parametric Properties** editing dialog.

   This dialog is the primary control and configuration of an Uncertainty Quantification. This is used to define the labels for the Figures of Merit, select the input variables, and define the distribution used to generate the input variable values. Finally, the Report tab allows the contents of the generated DAKTOA report to be configured.

7. Add a new Figure of Merit (FoM) by pressing the new ( ) button next to the **Figures of Merit** label.
8. Set the **Name** of the FoM to "FuelTemp".
9. Check the **Upper Limit** check box for the "FuelTemp" FoM.
10. Select the **Variables** tab.

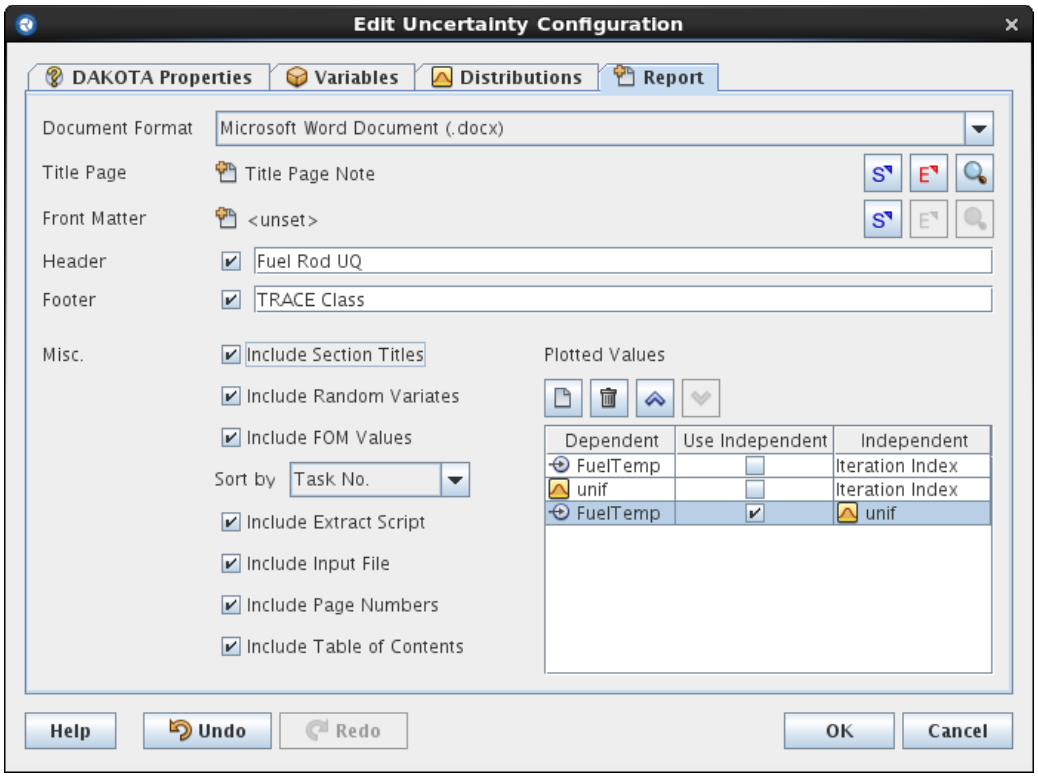
This tab contains the model input values that are modified as part of the Uncertainty Quantification. The TRACE plug-in allows UQ stream types to modify sensitivity coefficients, boundary component table data, and variable values. For this exercise, we will be modifying the sensitivity coefficient that controls the gap conductance of the fuel rods.

11. Add a new variable by pressing the new ( ) button at the top of the dialog.
12. Ensure that the **Sensitivity Coefficients** entry is selected in the variable category list.
13. Select "**gapCondSV**" and press the **Next** button.



14. Select the **Factor** radio button and press the **Finish** button.
15. Select the **Distributions** tab.
16. Set the **Name** of distribution **d1** to "unif".
17. Set the **Distribution** type to Normal.
18. Set the **μ (Mean)** value to 1.0.
19. Set the **σ (STDV)** value to 0.1.

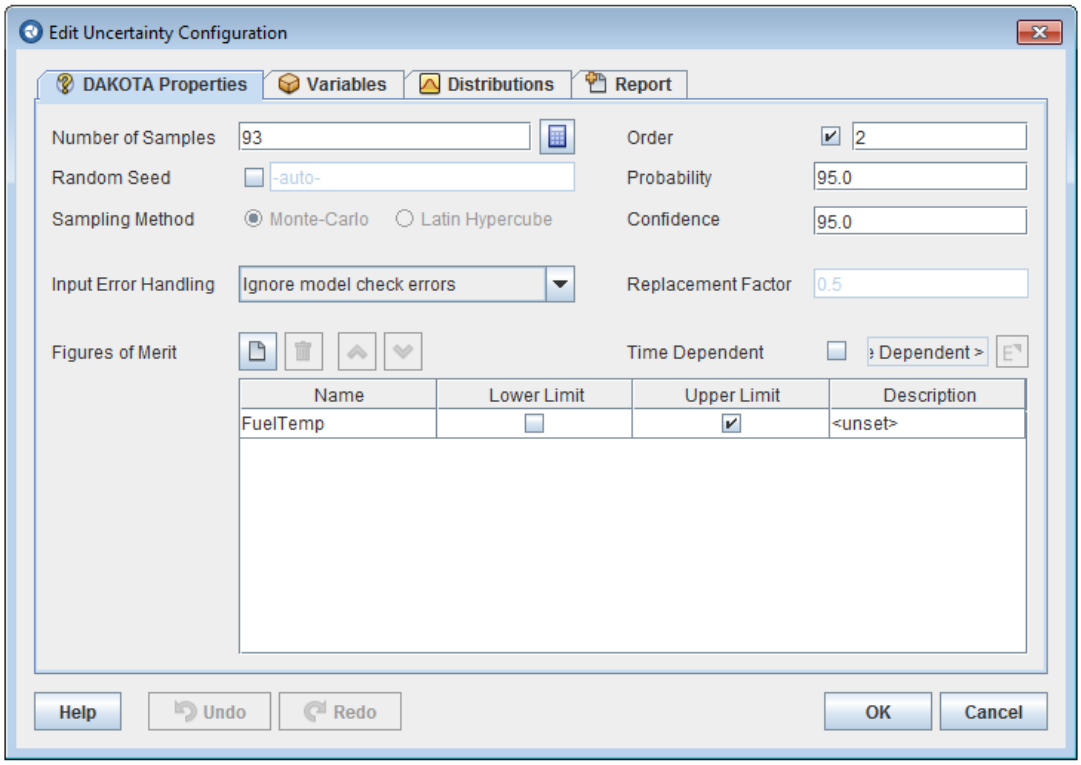
This defines Normal distribution about 1.0 with a standard deviation of 0.1. This will result in values that range between 0.7 and 1.3, as shown in the Probability Distribution graph. The distribution could be further modified with the Minimum and Maximum fields.



20. Select the **Report** tab.
21. Set the **File Format** to "Microsoft Word Document (.docx)".
22. Set the **Title Page** to "Title Page Note" by pressing the S button to the right of **Title Page**.
23. Activate the **Header** and set the text to "Fuel Rod UQ".
24. Activate the **Footer** and set the text to "TRACE Class".
25. Add a new plot by pressing the ( ) button below the **Plotted Values** label.
26. Select "**FuelTemp**" in the list and press the **OK** button.
27. Check the **Use Independent** check-box for the new plot.
28. Click in the **Independent** column in the **FuelTemp** row and press the S button that appears.
29. Select the "**unif**" entry and press the **OK** button.
30. Add a second new plot and select "**FuelTemp**" in the list.
31. Add a third new plot and select "**unif**" in the list.

    The report will include these three plots. The first is a plot that shows the figure of merit value vs the sensitivity coefficient, the second shows the figure of merit value vs the iteration index, and the third shows the sensitivity coefficient values vs the iteration index.

The final step in setting up the initial uncertainty stream type properties is to define the number of executions that will be performed. This can be entered manually or calculated to ensure the desired confidence level.



32. Select the **DAKOTA Properties** tab.
33. Activate the **Order** property by checking the check box.
34. Set the value of the **Order** property to "2".
35. Check the **Upper Limit** checkbox for the "FuelTemp" **Figure of Merit**.
36. Press the **Calculate** ( ) button next to the **Number of Samples** field.

    *This uses the selected **Sampling Method**, **Probability** and **Confidence** levels to determine the number of samples required for the targeted number of **Figures of Merit**. The number of samples indicates the number of input models that will be generated.*
37. Press **Yes** to proceed.
38. Press the **OK** button to close the dialog.

    The Stream properties are now defined. The rest of the modifications for submitting the UQ case are done in the job stream. The next steps will set the restart model node to Parametric. This indicates to the job stream that the variables modified by the UQ properties should be applied to the restart case.
39. Select the **UQ Stream** view.
40. Select the **Null-Transient** model node in the view.

41. Set the **Parametric** property to “True”.

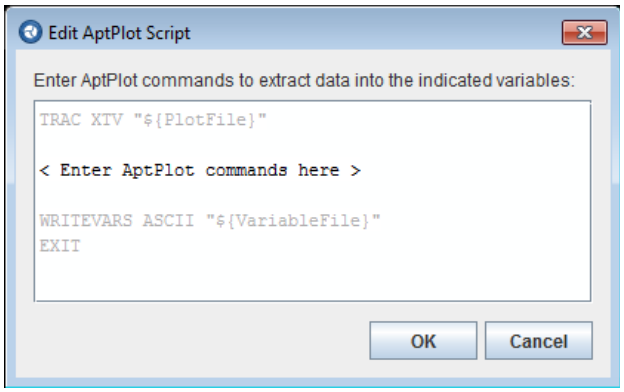
    Note: The model node now appears as a stack of models. This is the indication of a parametric model node.

The next steps require creating an Extract Data step to populate the figure of merit values. The Extract Data step uses AptPlot to parse the plot file generated by each TRACE execution. A single value is retrieved from the plot file and stored as the figure of merit for that run.

42. Use the **Insert Tool** to add a new Stream Step to the right of the TRACE Restart_Job step.
43. Set the **Insert into Job Stream** drop-down list to “UQ_Stream”.
44. Select **Extract Data** from the list of applications.
45. Press the **OK** button to finish adding the step.
46. Set the **Name** of the step to “Extract”.
47. Set the **Plot File Type** to “TRACE”.

The following steps will add an AptPlot script to the Extract data step that determines the figure of merit value from the TRACE plot file. For this exercise the maximum center-line temperature of heat structure 140 at the end of the calculation will be stored as the figure of merit “FuelTemp”.

48. Open the **AptPlot Script** property editing dialog.

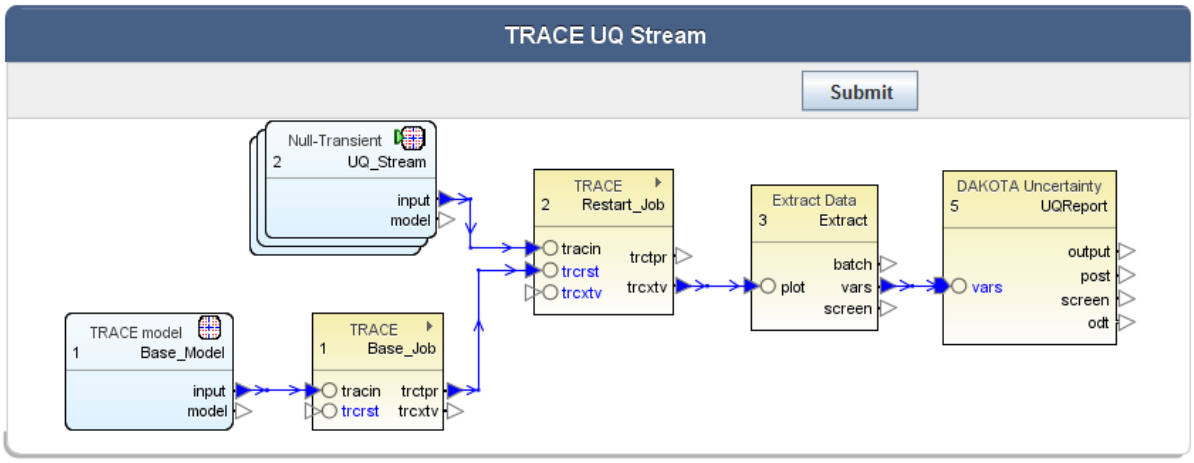


49. Replace the “< Enter AptPlot commands here >” line with the following text:

```
CALC "xval = maxXval(T0_sv1)"
CALC "<tin> = getPtsAtX(xval, 'T0_tsurfi-140A%2N')"
CALC "FuelTemp = maxYval(<tin>)"
```

50. Press the **OK** button.
51. Use the connect tool to connect the **trcxtv** output of the **Restart_Job** step to the **plot** input of the **Extract Data** step.
52. Use the **Insert Tool** to add a **DAKOTA** job step to the right of the **Extract Data** step.
53. Set the step **Name** to “UQReport”.

54. Use the **Connect Tool** to connect the **vars** output of the **Extract Data** step to the **vars** input of the **DAKOTA** job step.



The job is now ready to submit. The next steps will submit the job to the Calculation Serve, and examine the generated report.

55. Right-click the **"UQ_Stream"** node in the **Navigator** and select the **Submit Stream to Local** menu item.
56. Press **OK** in the confirmation dialog for executing 188 jobs and wait for the streams to complete.
57. Once the stream is complete, expand the "**UQ_Stream"** node in the Job Status Navigator.
58. Right-click the "**UQReport"** node and select the **Open Folder** menu item.
59. Open the ***report.docx*** file in the UQReport directory.

Note: This file can also be opened through the "View Files → Documents → docx – report.docx" menu item from the completed UQReport task in Job Status.

This completes the Uncertainty Quantification exercise.

Exercise 18. Creating a TRACE Model Notebook

The Model Editor provides a means of generating model-wide reports as a single annotated document, called a "model notebook". Information such as calculations, export data, model status, attribute descriptions, etc. are included in the notebook along with "model notes." Model notes are HTML notes that can be associated with components and their attributes. This exercise will briefly explore the process of creating a new model note and including it in a model notebook.

The following steps will guide you through the exercise.

1. Open **SNAP_Exercises/TraceTypPWR_Notebook.med**, included with these exercises.

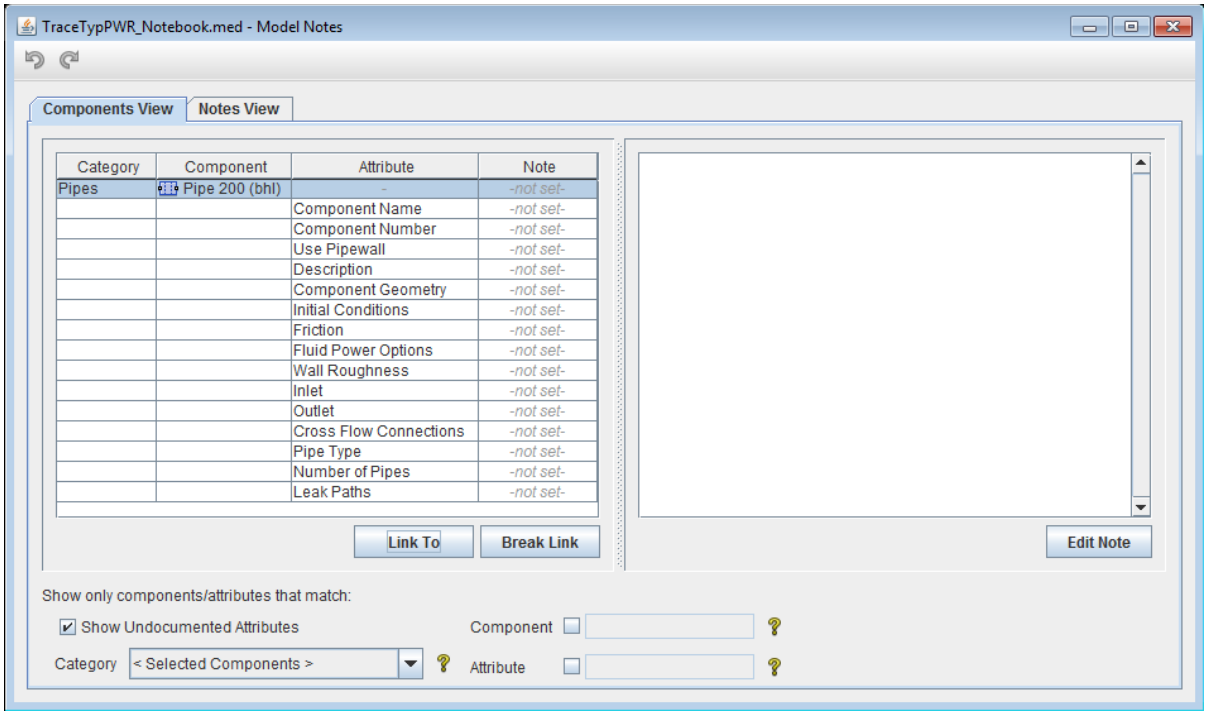
   This model was created by converting the RELAP5 typpwr model to TRACE using the SNAP R52TR plug-in. The original model and the conversion process were documented with model notes.

2. Select **Pipe 200** in the Navigator.

   ***Pipe 200** is located under **Hydraulic Components → Pipes**.*

3. Press the **New Model Note** button ( ) At the top of the Property View, to the right of the the **Pipe 200** label.

   *The **Model Notes** dialog will be opened, as shown below. This dialog is used to manage all notes in the model.*

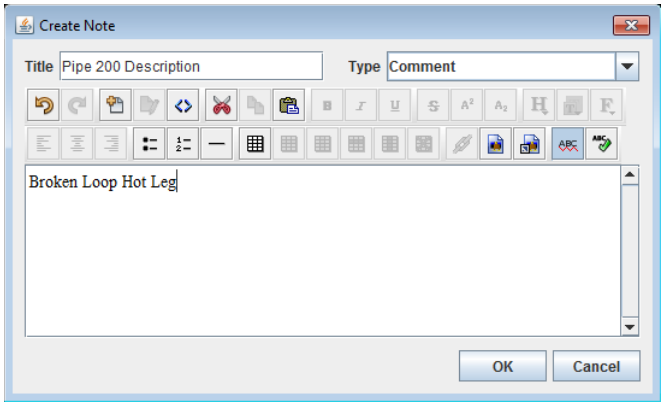


*The central table lists components in the model and their attributes. By opening the window from the context of **Pipe 200**, the dialog is opened to a view restricted to the properties of that component.*

4. Select the first row, which represents **Pipe 200** itself.
5. Press the **Link To** button to create a new note or link to an existing note.

   A prompt will appear asking which action to take. This can be used to add an entirely new model note or to add a reference to an existing note.
6. Make sure "**Link to a new note**" is selected at the prompt, then press the **OK** button.

   The Create Note window is displayed, as shown below. This dialog will be used over the next several steps to define the contents of the note.



7. At the top of the dialog, set the **Title** of the new note to "**Pipe 200 Description**".
8. Left-click in the central text area and type "**Broken Loop Hot Leg**".
9. Press the **OK** button in the **Create Note** dialog.

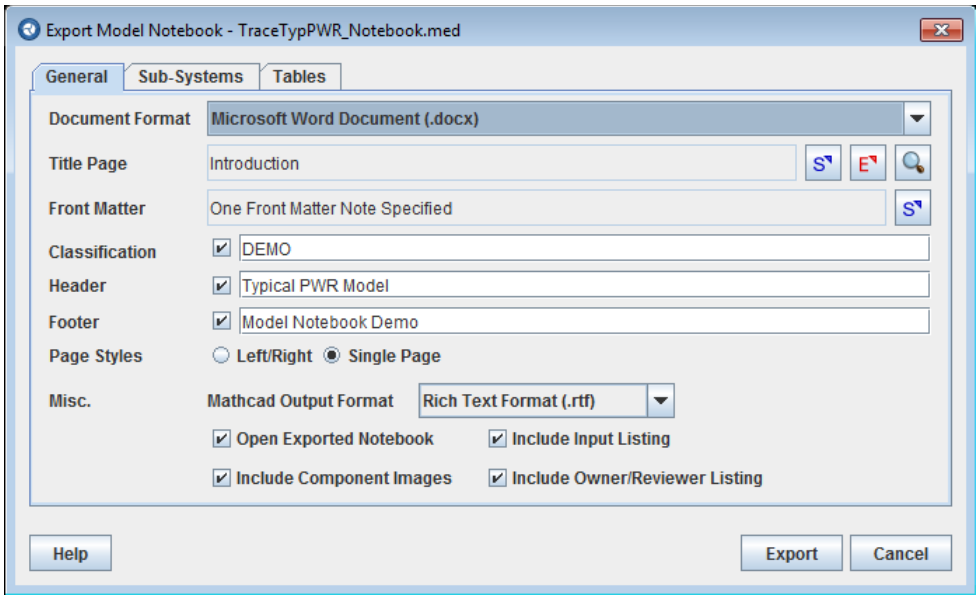
   *Back in the **Model Notes** dialog, the **Note** column for **Pipe 200** will indicate that the new note has been associated with this component.*

   A more detailed description of the Model Notes and Create Note dialogs can be found in the SNAP User's manual.
10. Close the **Model Notes** dialog.

Note: The **Model Note** button for the component has a different icon ( ). This indicates that a note has been associated with the indicated component or attribute. Otherwise, the button functions identically as before.

11. Right-click on the **TraceTypPWR_Notebook.med** model node and select **Export → Model Notebook** from the pop-up menu.

    *The **Export Model Notebook** dialog will appear, as shown below. This dialog allows exporting a full description of the model as a word-processor document containing a full report of the model contents.*



12. Set the **Document Format** to "Microsoft Word Document (.docx)".

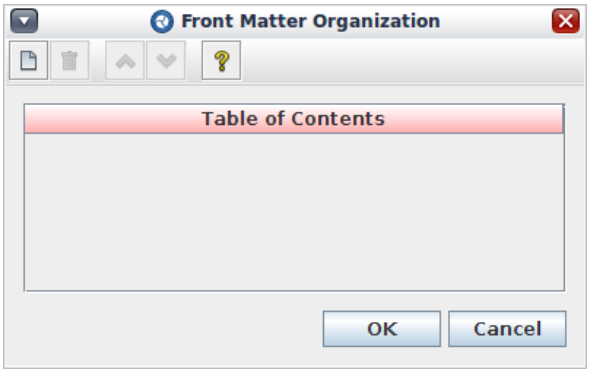
    Model notebooks can be exported in either Open Document Format (.odt) or Microsoft Word XML (.docx) format.

13. Press the **Title Page Select** button (**S**).

    *The **Title Page** field can be used to select, edit, and preview a model note used as the first page in the report.*

14. Select the "**Title Page**" note from the list and press **OK**.

15. Press the **Front Matter Select** button (**S**).



*The **Front Matter Organization** dialog will appear, as shown above. This dialog is used to specify the contents shown at the beginning of the generated report.*

16. Press the **New** button ( ).

*A **Model Note** selection dialog will appear.*

17. Select **Introduction** in the dialog and press the **OK** button.

***Introduction** will appear in the list above **Table of Contents**. This indicates that the first thing in the dialog (after the title page, if one was specified) will be the Introduction note, followed by the table of contents.*

18. Select **Introduction** in the list and press the **Down** button ( ).

The items are reordered. Now, in the generated document, the table of contents will appear before the note.

19. Press the **OK** button in the **Front Matter Organization** dialog.

Note: The **Front Matter** in the **Export Model Notebook** dialog will change to indicate that the front matter of the report will contain a single note.

20. Enable the **Classification** field and enter the text "**DEMO**".

The classification label will appear on the header and footer of the document, centered, in a large font. Typically this is used for labels such as UNCLASSIFIED, PROPRIETARY, etc..

21. Enable the **Header** field and enter the text "**Typical PWR Model**".

This text will appear in the header of every page.

22. Enable the **Footer** field and enter the text "**Model Notebook Demo**".

This text will appear in the footer of every page.

23. Change **Page Styles** to **Single Page**.

This controls whether the generated document is intended to be printed in double-sided book format or as a document with neutral positioning.

24. Make sure that each check-box in the **Misc** section is selected.

    Note: Each of these miscellaneous settings is described in the SNAP User's manual available by pressing the **Help** button.

25. Press the **Export** button.

    *The **Export Model Notebook** dialog will be replaced by a file browser, used to select the location at which the notebook is created.*

26. Name the report "**StandPipeReport.docx**" and press the **Export Model Notebook** button.

    A series of progress dialogs will appear while the report is generated. Once complete, the report will be opened in the application associated with DOCX files.

27. In the word-processor, update all of the fields in the document.

    *For most versions of Microsoft Office, select the entire contents of the document (Ctrl-A) then right-click and select "**Update Field**".*

28. Examine the document.

    Note that the Introduction appears after the table of contents. Also examine how the various components are represented throughout the report.

29. Close the word processor.

Exercise 19. Job Stream Sequences

Introduction

Job Stream Sequences were added to the Model Editor to support analyses that require looping, convergence, or optimization. A Job Stream Sequence is a set of Job Streams that will be executed in order. Each entry (Job Stream) in the Sequence will be executed in turn until all of the entries have been completed. If looping behavior is required for an entry in a Sequence then it can be executed multiple times as separate "iterations" controlled by a user-defined Python Script. The script can be used to change variable values and determine (based on keywords) whether the loop is finished.

The Python scripts can also be used to set the file locations in External Files and File Sets by using Dynamic File Replacement. This allows Job Streams to be chained together using the output files of one stream as inputs for subsequent streams. Dynamic File Replacement allows the available files to be searched using a range of criteria including file type, keyword values, and iteration number.

Sequence entries (i.e. Job Streams) are executed in the order in which they are defined in the Sequence. If an entry will be executed only once (i.e. no looping) then it will be placed in the Sequence folder directly. If an entry will be executed multiple times then the set of executions will be placed in a sub-folder. The Python scripts for setup (Entry_setup.py), iteration (Entry_iterate.py), and their output (Entry.py_screen) will also be placed in this sub-folder.

The purpose of this exercise is to introduce the basic functionality of Job Stream Sequences. After completing this tutorial the user will be able to create a new Sequence, define an iteration script, and select the output of one stream in a Sequence as input for another. For more detailed information on the specific features covered in this tutorial, refer to the SNAP User's Manual.

1. Open the included **SNAP_Exercises/StandPipe_Sequences.med** model.
2. Select the **Job Streams** node in the Navigator.

   Selecting the Job Streams node in the Navigator will show the Sequences property editor in the main Property View. This editor shows the number of sequences currently defined in the model.
3. Open the Sequences dialog by pressing the **Edit** button on the **Sequences** property.

   Note: The sequences dialog is the primary user interface for creating, editing and submitting, Job Stream Sequences. This dialog has a tree of Sequences on the left with each Sequence's Entries as child nodes. The right side of the dialog displays the properties of the selected node (Sequence or Sequence Entry) on the left. The main toolbar is at the top of the dialog and contains various buttons for creating and editing Sequences.

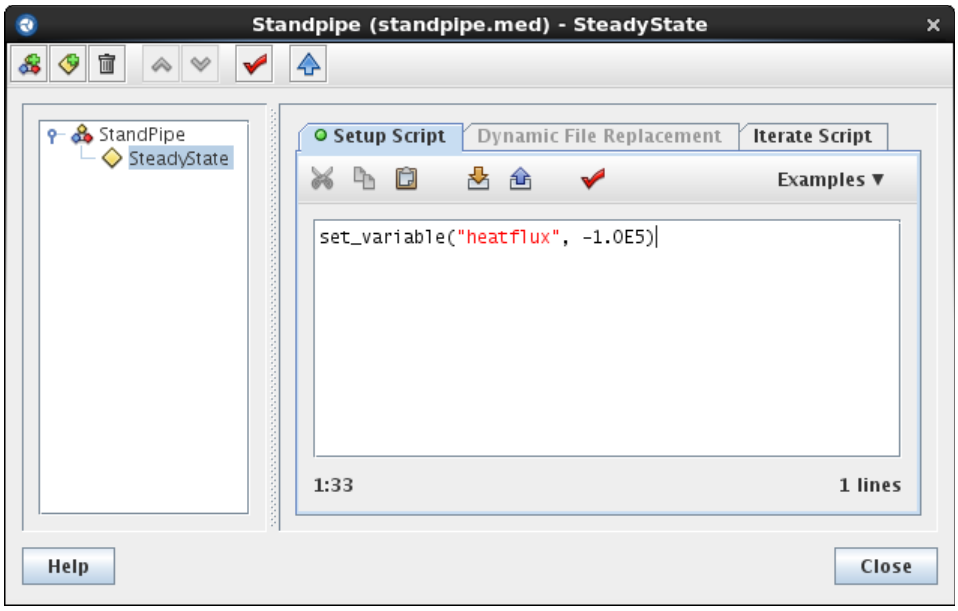
The next step is to create a new Sequence and define its submission properties. The submission properties will apply to all of the Job Streams in the Sequence.

4. Press the **New Sequence** () button.

   This button adds a new Job Stream Sequence to the current model.
5. Set the **Name** field to "StandPipe".
6. Set the **Relative Location** to "DEMO/".

   Notice that the initial platform is set to Local. Sequences are currently only supported on the Local Calculation Server. In the next few steps we will add a Job Stream to the Sequence and define the initial value for a variable.
7. Press the **New Sequence Entry** () button.

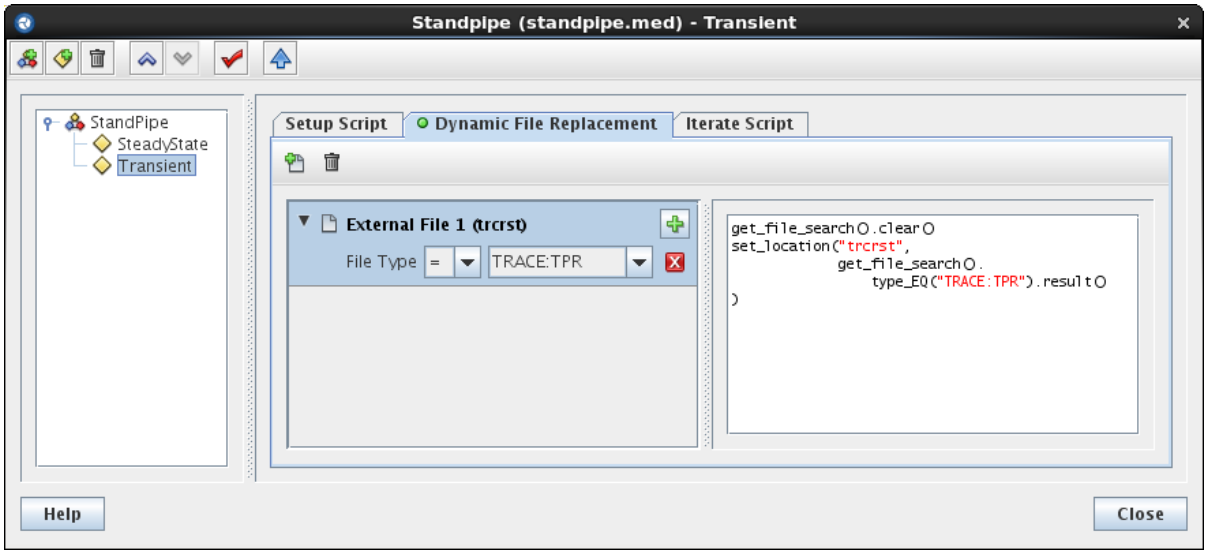
   Note: This button adds a new Entry (Job Stream) to the currently selected Sequence. This button will only be enabled when a Sequence is selected in the tree and the model contains a Job Stream that is not already included in the selected sequence.
8. Select the "**SteadyState**" Job Stream and press the **OK** button.
9. Ensure that the **Setup Script** panel is shown.
10. Click on the **Examples** button and select **Set A Variable Value** from the menu.
11. Change the referenced variable from "V1" to "heatflux".
12. Change the initial value from 1.0 to -1.0E5.



This will set the value of the heatflux variable to -1.0E5 before the stream is executed. Now we will add an additional Sequence Entry that will use the output generated from the SteadyState stream as input.

13. Press the **New Sequence Entry** () button.
14. Select the "**Transient**" Job Stream and press the **OK** button.

    Notice that the Dynamic File Replacement tab is now enabled. This means that the selected Sequence Entry contains either an External File or File Set and that the Entry is not the first Entry in the Sequence.
15. Select the **Dynamic File Replacement** tab.
16. Press the **New File Replacement** () button.



17. Select "**External File 1 (trcrst)**" in the selection dialog and press **OK**.

    Note: The Python script appears in the right panel of the dialog. This script will be executed after the setup python script and before executing the Job Stream. The first line clears any previous file searches. The next line indicates that the file location for File Set 'trcrst' is being defined. The following lines define the criteria for the file search. The last line shows that the file type for the file must be 'TRACE:TPR'. Currently, in the SteadyState Job Stream, there is only one output file with the file type 'TRACE:TPR, so this provides enough information to return the TPR output file from the steady state case.

18. Press the **Check Sequence** () button.

    This will open an Error Report dialog for the currently selected Sequence. This report contains the results of validating the Sequence as well as each of the Job Streams referenced by Sequence Entries. In this case, there is a warning stating that the "Transient" job stream contains an error. This is due to the fact that External File 1 does not have a location defined. The Dynamic File Replacement added above will correct this problem before the stream is executed.

19. Press the **Close** button to close the Error Report.
20. Press the **Submit Sequence** () button.
21. Press **OK** to submit the sequence without correcting the warning.
22. Wait for the submitted Sequence to complete.

    Job Status will open to show the Sequence Manager's log output while the Sequence is running. The locations of the Python output files are listed directly in the log to make them easier to find.

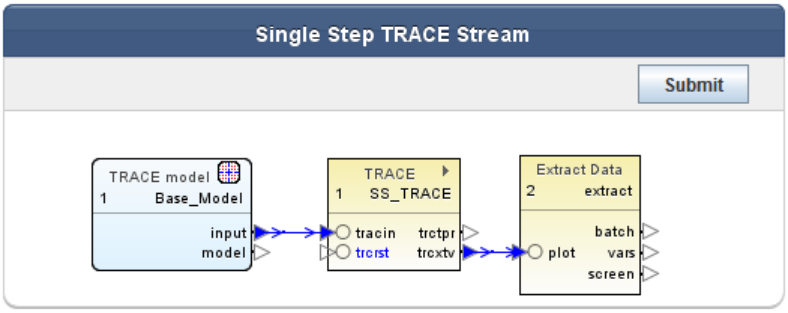
```
[<Date>] - Writing Python stderr/stdout for Transient iteration 1 to: <Location>
```

    Notice that the streams are executed in the DEMO/StandPipe/ folder.

23. Close Job Status.

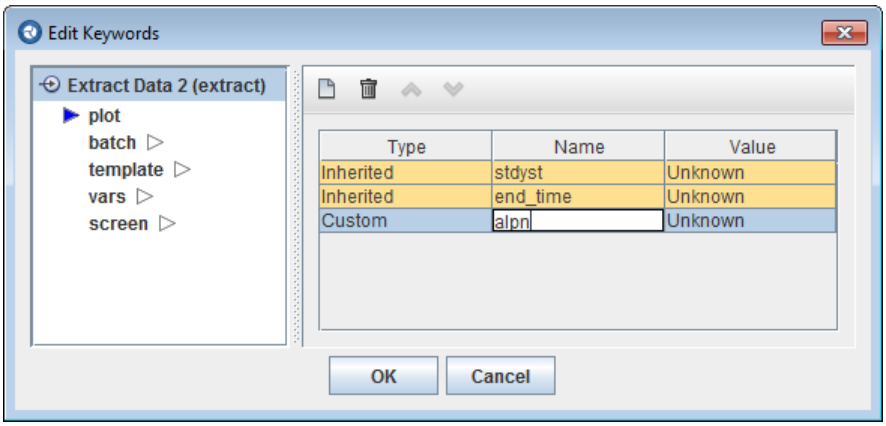
Next, we will modify the Sequence to iterate on the steady state case, changing the value of the heat flux variable until the void fraction at the end of the pipe is greater than 0.5. This will be accomplished using an Extract Data step to retrieve the value from the plot file and a post processor Python method to write the keyword value. The first step is to add an Extract Data step to the stream that will write the void fraction of the last hydraulic cell to a new keyword.

24. Press the **Close** button to close the Sequences dialog.
25. Select the "**Steady State Stream**" View in the view tabs.
26. Use the **Insert Tool** to add a new Extract Data Job Step to the right of the **SS_TRACE** Step.



27. Set the new Job Step's **Stream** to "SteadyState".
28. Set the new Job Step's **Name** to "extract".
29. Set the **Plot File Type** to "TRACE".
30. Press the **Edit** button for the **Input Files** property.
31. Click in the table cell at row 1 ("plot") column 4 ("source").
32. Press the **Edit** button to open the Select Input Source dialog.
33. Select "**TRACE 1 (SS_TRACE)**" and press **OK**.
34. Press the **OK** button to close the **Define Input Files for Extract Data** dialog.

Now we must define the keyword that will hold the plot steering criteria. To set the value of a keyword on a step or file the keyword must already exist. The next steps create a new keyword for the Extract Data step that will be used to steer the iteration.



35. Press the **Edit** button for the Keywords property.
36. Select "**Extract Data 2 (extract)**" in the list on the left hand side of the dialog.
37. Press the **New Keyword** ( ) button.
38. Set the **Name** of the new Keyword to "alpn".
39. Press the **OK** to close the **Edit Keywords** dialog.

Now the AptPlot script will be updated to extract the desired value to the var output file. Any number of variables may be written to this file, each included with the "SAVEVAR" method. For this example a single SCALAR variable is saved, which will then be read-in by a post-processor python method.

40. Press the **Edit** button for the **AptPlot Script** property.
41. Enter the following script and press **OK**.

```
CALC "ALPN = yAtMaxX(T0_alpn-21A20)"
SAVEVAR "ALPN"
```

42. Press the **Edit** button for the **Custom Processing** property.
43. Select the **Post-Execution Python** tab at the top of the dialog.
44. Enter the following script and press **OK**.

```
f = open("variables.dat", 'r')
for line in f:
    if line.startswith("SCALAR"):
        values = line.split("=")
        set_keyword('alpn', values[1].strip())
```

The extract step now defines a keyword with the void fraction from node 20 of pipe 21. The Sequence iteration can use this keyword value to determine the result of each SteadyState iteration.

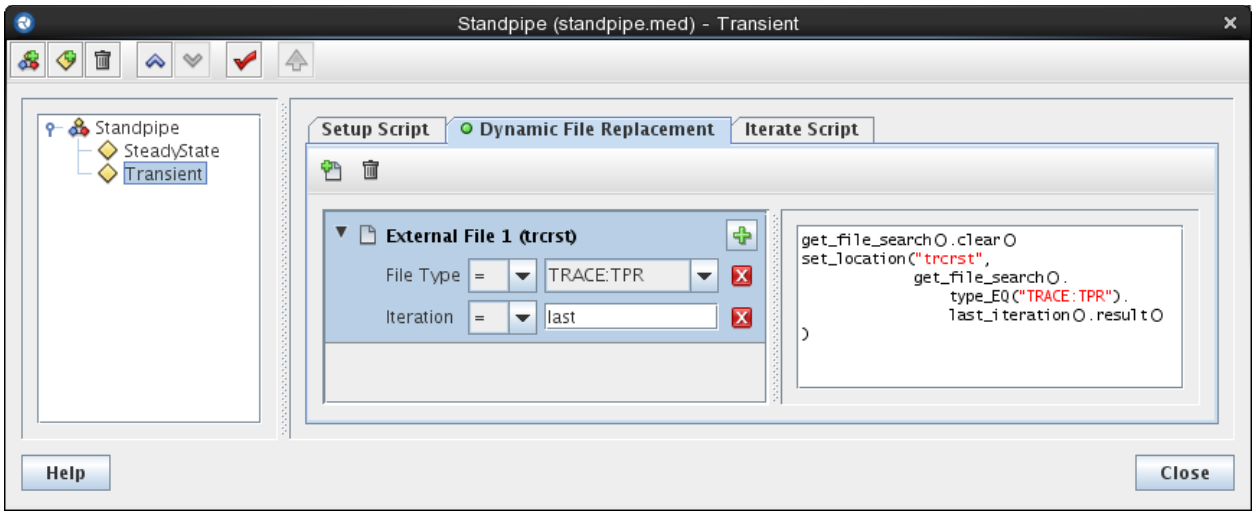
The next steps will add an iteration script to the SteadyState Sequence Entry. The Iteration Script will increase the constant heat flux of the pipewall heat structure until the void fraction of the pipe hits 0.5.

45. Select the "**Job Streams**" node in the Navigator.
46. Press the **Edit** button for the **Sequences** property.
47. Select the "**SteadyState**" Sequence Entry.
48. Select the **Iterate Script** tab.
49. Enter the following script into the panel.

```
print "Calculating Input For Iteration " + str(get_iteration_number())
# Get the keyword set on the extract step
alpn = get_keyword("extract/alpn")
print "ALPN Keyword = " + str(alpn)
# The end value is 0.5 void fraction
if alpn > 0.5:
    print "End Reached"
    set_finished()
else:
    # Keep increasing the heatflux by -1.0E5 until the desired
  # value is reached.
    heatflux = get_variable("heatflux") - 1e5
    print "Setting Heat Flux to: " + str(heatflux)
    set_variable("heatflux", heatflux)
```

The SteadyState Sequence Entry will now execute multiple times, creating a new TPR file for each iteration. The Dynamic File Replacements in the Transient Entry will now need to be updated to select just one of the TPR files.

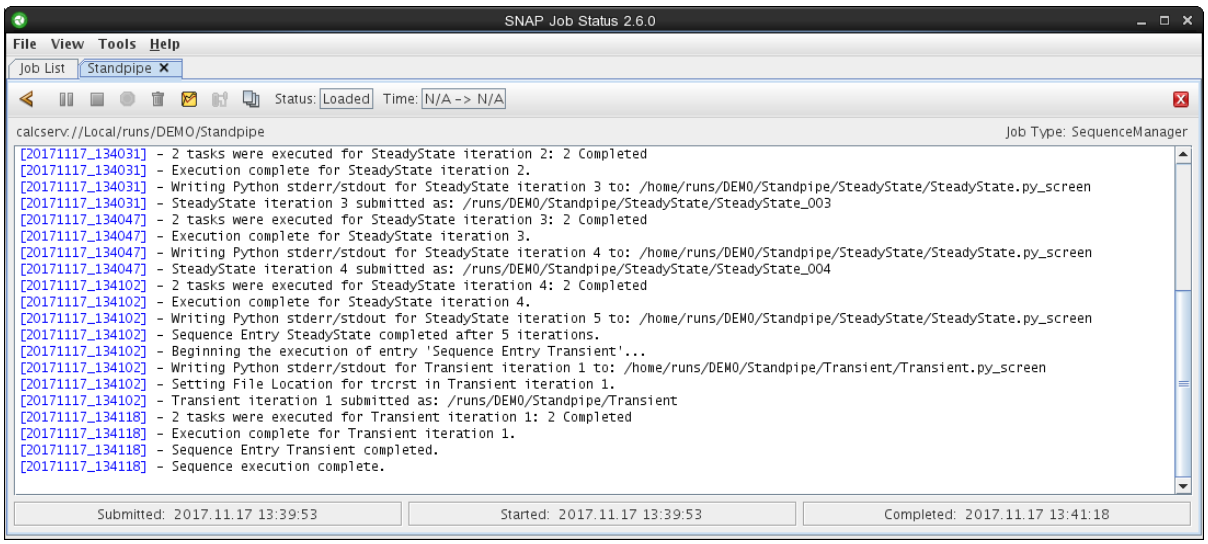
50. Select the "**Transient**" Sequence Entry on the left side of the dialog.
51. Select the **Dynamic File Replacement** tab.



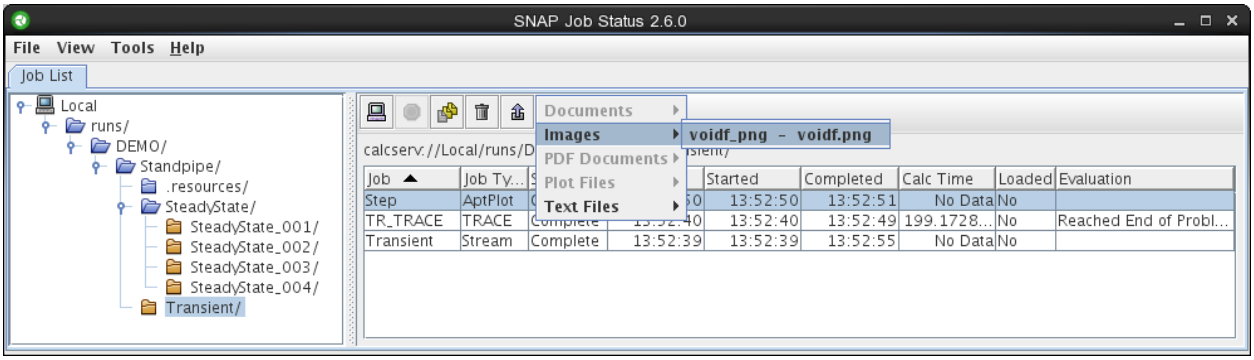
52. Press the **New Search Criteria** (+) button to for **External File 1 (trcrst)**.
53. Select "**Iteration**" and press **OK**.

Note: The value of the iteration criteria is set to "last". By default, the iteration criteria is used to select the most recent iteration. This ensures that the TPR file for the transient case comes from the last executed steady state iteration.

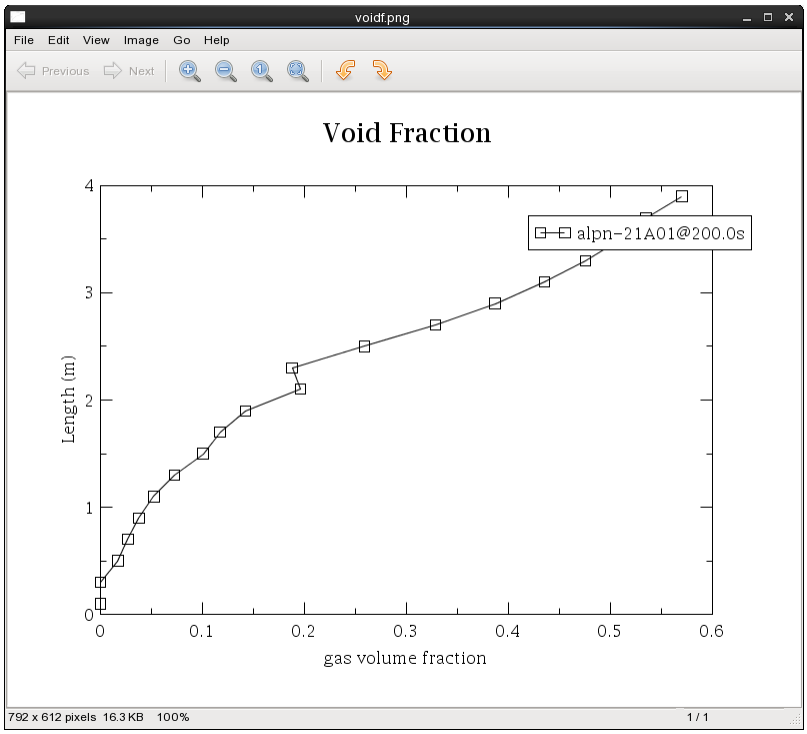
54. Select the "**StandPipe**" node in the **Sequence Tree** on the left.
55. Press the **Submit Sequence** ( ) button and wait for the sequence to complete.



56. In Job Status, close the **Standpipe** tab.
57. Navigate to the AptPlot step in the Transient Job stream.



58. Open the voidf.png output file.



This completes the Sequence Basics exercise.

Exercise 20. AptPlot Commands

One of AptPlot's most powerful features is its scripting language: any aspect of a plot can be specified by a command. This exercise completely recreates the plot from the last exercise using only commands.

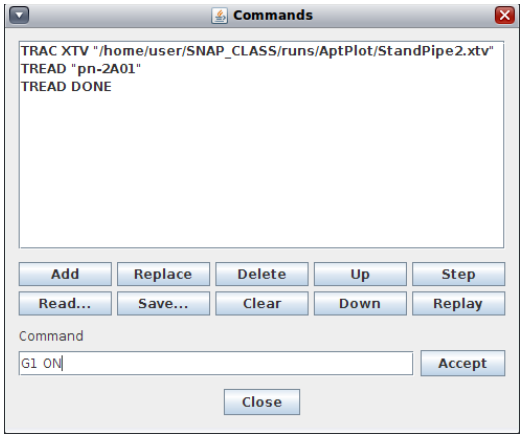
Before proceeding, a note about entering commands: each command in this exercise must be entered exactly as listed. If a command is entered incorrectly, one of two things will happen. The most likely occurrence is AptPlot will warn the user about syntax error or incorrect argument. In this case, close the error console and re-enter the command. The second type of error, entering a valid but incorrect argument, will alter the plot such that it will not match the expected result. This type of error may not be apparent until comparing the final plot to the figure. Be sure to enter each command exactly as listed, in the indicated order.

1. Open AptPlot. If it is already open, select **File → New** from the main menu and select **Yes** at the prompt.

   AptPlot is displayed with a new and empty plot.

2. Select **Window → Commands** from the main menu.

   This will display the Command Window, which can be used to enter, edit, save, and read commands.



3. Enter the following commands:

```
TRAC XTV "<install>/SNAP_Exercises/StandPipe.xtv"
TREAD "pn-2A01"
TREAD DONE
```

   Be sure to replace "**<install>**" with the complete path to the materials and resources provided with this exercise.

   This set of commands opens a TRACE plot file and reads channel data.

   The TRAC command opens the file. The XTV argument specifies the file type. The complete path of the plot file tells AptPlot where to locate the data.

The first time the TREAD command is used, it queues a channel for the next read. The second TREAD command performs the read, plotting channel data into the current graph.

4. Enter the following command:

```
VIEW XMIN 0.25
```

In the first AptPlot exercise, the bounds of the plot were adjusted to display the Y axis label. This command moves the left edge of the graph to the right by increasing the minimum X value from .15 to .25. The entered value is a viewport coordinate.

5. Press the **Redraw graph** button  in the main toolbar.

*Before the changes made by the last command are displayed, the graph must be redrawn**.** Few commands automatically redraw the graph. All commands entered in subsequent steps will share this requirement, so **make sure to press the redraw button after entering each set of commands**.*

6. Enter the following command:

```
G1 ON
```

The second graph is created. Note the "G1" portion of the command. Graphs can be specified in commands by the letter G followed immediately by an index. Graph indices start at 0, so the second graph is listed as "G1".

7. Enter the following command:

```
ARRANGE(2, 1, .2, .2, .4)
```

AptPlot has a custom ARRANGE command that performs the same task as the Arrange Graphs dialog seen in a previous exercise. The arguments are, in order: number of rows, number of columns, margin around the arranged graphs, gap between columns, gap between rows.

8. Enter the following commands

```
FOCUS G1
TREAD "rovn-2A01"
TREAD DONE
```

The FOCUS command sets the current graph selection to the second graph. The latter two commands should be familiar from an earlier step.

9. Enter the following command:

```
LEGEND .85, .30
FOCUS G0
LEGEND .85, .65
```

At this point in the previous exercise, the legends were moved to more reasonable locations. This command moves the legends' upper-left corners to the given coordinates. Once again, the locations are specified in view coordinates.

10. Enter the following commands:

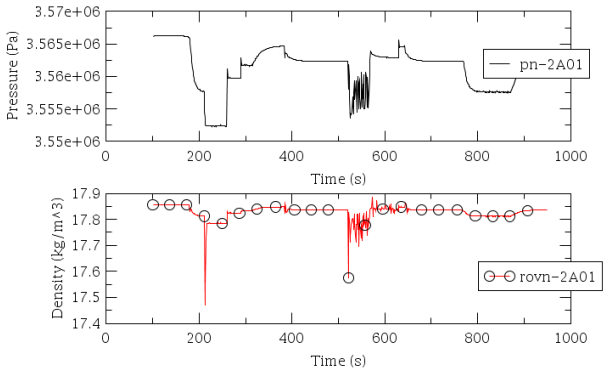
```
FOCUS G1
S0 LINE COLOR 2
S0 SYMBOL 1
G1.S0 SYMBOL SKIP 35
```

The last step of the original exercise set several properties of the set. These commands recreate that edit.

Notice the "S0" for the first two commands. Set identifiers work exactly like graph identifiers. The third command illustrates how graph and set indicators can be compounded into a discrete entity used to target a set regardless of the graph selection.

Note that the color and symbol are specified by an index instead of a label. This is true of all places where color, symbol, line style, font, etc. are specified in AptPlot commands. To determine the index of a specific color, count its location from the top of its editor, starting at 0.

Once all commands have been entered and a redraw has occurred, the final plot should match the following image:



The next several steps will demonstrate additional capabilities of the Command Window.

11. Press the **Save...** button in the Command Window.

    A file dialog is displayed. The Command Window allows saving a command session for future usage.

12. Save the commands to a local file named "**CommandExercise.bat**".

13. Press the **Clear** button in the Command Window.

    The command history is cleared.

14. In the AptPlot main menu, select **File → New** to clear the plot. Press **Yes** at the prompt to continue.

15. Back in the Command Window, press the **Read...** button.

A file dialog is displayed.

16. Select the **CommandExercises.bat** file stored earlier.

    The command history from earlier has been restored.

17. Select the first command in the command list.
18. Press the **Step** button in the Command Window.

    The TRAC command is executed and the selection in the command history moves to the next command. The Step button is used to quickly execute a block of commands in the Command Window.

19. Repeatedly press the **Step** button until all commands in the list have been executed.
20. Redraw the graph.

    The plot should appear exactly as it did before selecting "File → New".

21. Save the plot to **ex16_commands.apf**.
22. Using a text editor, open the **ex16_commands.apf** file saved in the previous step.

    AptPlot Files (APF) are text files. More specifically, these files contain a list of AptPlot commands that recreate a plot in its entirety.

    In the APF, notice that most commands are prefixed by the "@" character. This prefix distinguishes commands from data. Scrolling down to the bottom of the file, notice where AptPlot lists the contents of each data set. The data values of a set are not prefixed by any special character.

    When entering AptPlot commands and writing batch files, any command seen in an APF can be used by removing the "@" prefix. Once the user is familiar with AptPlot terminology, APF files are one of the best resources for quickly finding a specific command. Of course, an APF only uses commands that set a property to a complete value. AptPlot provides many other commands that perform formatting and layout, analysis and manipulation, or even store off-set data for other commands. These commands are all detailed in AptPlot's extensive documentation, available from the Help menu.

Exercise 21. AptPlot Scripting

Some advanced features of AptPlot batch scripting are explored in this exercise. The exercise uses the elevations specified in one series of data channels to create an axial profile plot of void fractions defined by another set of channels.

The following steps will guide you through the exercise.

1. Open AptPlot. If it is already open, select **File → New** from the main menu and select **Yes** at the prompt.

   AptPlot is displayed with a new and empty plot.

2. From the AptPlot main menu, select **Edit → Preferences**.
3. In the Preferences dialog, find the **Run in safe mode** option, which should appear around the middle of the dialog.
4. Ensure that the **Run in safe mode** option is unchecked and press the **Apply** button.
5. Press the **OK** button to close the dialog.
6. From the AptPlot main menu, select **Window → Commands**.
7. If the **Commands** dialog has commands listed from the previous exercise, press the **Clear** button.

   The existing commands in the command list will be removed.

8. In the **Commands** dialog, press the **Read** button.

   A file browser will appear, used to select a batch script file.

9. Navigate to and select the "**SNAP_Exercises/script.b**" file included with this exercise, then press the **Open** button.

   The command list at the top of the dialog will be populated with the contents of the file. These commands are listed as follows, and will be referred to throughout the exercise.

```
TRAC XTV "<PATH>\AptPlot\Transient.xtv"
GETP "<PATH>\AptPlot\axial.par"
CALC "<elevations> = getPtsAtX( 120.0, 't0_rdzNperm-31A%2N')"
CALC "<void1> = getAxial( 120.0, <elevations>, 't0_alpn-21A%2N')"
CALC "<void2> = getAxial( 150.0, <elevations>, 't0_alpn-21A%2N')"
CALC "<void3> = getAxial( 200.0, <elevations>, 't0_alpn-21A%2N')"
CALC "<void4> = getAxial( 250.0, <elevations>, 't0_alpn-21A%2N')"
CALC "<void5> = getAxial( 300.0, <elevations>, 't0_alpn-21A%2N')"
CALC "<void1> = flipXY( <void1> )"
CALC "<void2> = flipXY( <void2> )"
CALC "<void3> = flipXY( <void3> )"
CALC "<void4> = flipXY( <void4> )"
CALC "<void5> = flipXY( <void5> )"
PLOTVAR "<void1>"
PLOTVAR "<void2>"
PLOTVAR "<void3>"
PLOTVAR "<void4>"
PLOTVAR "<void5>"
TITLE "Simple Standpipe Problem"
```

```
SUBTITLE "Void Fraction"
XAXIS LABEL "Void Fraction"
YAXIS LABEL "Elevation (m)"
S0 LEGEND "120.0 s"
S1 LEGEND "150.0 s"
S2 LEGEND "200.0 s"
S3 LEGEND "250.0 s"
S4 LEGEND "300.0 s"
REDRAW
HARDCOPY DEVICE "PDF"
PRINT TO "<HOME>\VoidProfile.pdf"
PRINT
```

10. Select the **TRAC XTV** line at the top of the script.

    *The line will appear in the **Command** field at the bottom the dialog. This can be used to modify the line or execute it again.*

    This command is used to load the TRACE XTV file for use by AptPlot.

11. In the **Command** field, replace the "**<PATH>**" portion of the command with the full path to the "**SNAP_Exercises**" folder included with this exercise.

    ***Note:** Do not press the Enter key. This would execute the command. These commands will be executed as a complete script toward the end of the exercise.*

12. Press the **Replace** button in the dialog.

    The command in the command list will be replaced by the edited command.

13. Select the second command:

    ```
    GETP "<PATH>\SNAP_Exercises\axial.par"
    ```

    *Similar to a previous step, the **<PATH>** value needs to be replaced with the path to the AptPlot folder.*

14. In the **Command** field, replace the "**<PATH>**" portion of the command with the full path to the "**AptPlot**" folder included with this exercise.

15. Press the **Replace** button in the dialog.

    *The **GETP** command is used to retrieve an AptPlot parameter file, which contains all the formatting needed to create an attractive plot, and applies it to the current plot. Several commands later in the script will be used to override the formatting specified in the parameter file, allowing it to serve as a template which individual scripts can use as a basis.*

16. Examine the third command:

    ```
    CALC "<elevations> = getPtsAtX( 120.0, 't0_rdzNperm-31A%2N')"
    ```

    *This command is used to load elevation data from the XTV file. The first argument indicates that the values are retrieved at time **120.0** seconds. The second argument is a pattern that indicates the data channels from which the elevation data is retrieved. The **%2N** indicates that a channel index shall be substituted into that portion of the pattern, with a minimum of two digits used to represent the index. With this particular XTV file,*

*channels **rdzNperm-31A01** through **rdzNperm-31A20** will be used to retrieve elevation data.*

*The result is an Equation Interpreter vector assigned the name **<elevations>** that contains channel indexes as its independent data and elevations for the dependent data. The contents of the vector are as follows:*

Independent	Dependent	Retrieved From
1	0.1	rdzNperm-31A01 at time 120.0
2	0.3	rdzNperm-31A02 at time 120.0
3	0.5	rdzNperm-31A03 at time 120.0
...		
20	3.9	rdzNperm-31A20 at time 120.0

This will be used by the getAxial commands to create the axial plot vectors.

Note: All Equation Interpreter commands use the CALC prefix. The Equation Interpreter was added to AptPlot to provide calculation capabilities not found in the basic command syntax, such as the ability to use data channel names as a vector, automatic interpolation when performing calculations on vectors of varying lengths, and convenience functions for defining and operating on these vectors.

17. Examine the next section of commands:

```
CALC "<void1> = getAxial( 120.0, <elevations>, 't0_alpn-21A%2N')"
CALC "<void2> = getAxial( 150.0, <elevations>, 't0_alpn-21A%2N')"
CALC "<void3> = getAxial( 200.0, <elevations>, 't0_alpn-21A%2N')"
CALC "<void4> = getAxial( 250.0, <elevations>, 't0_alpn-21A%2N')"
CALC "<void5> = getAxial( 300.0, <elevations>, 't0_alpn-21A%2N')"
```

*These commands take the elevation data specified previously and use it to retrieve axial plot data, once again at the times indicated by the first argument. The **<elevations>** channel provided as the second argument has two purposes.*

1. *The independent data (which contains channel indexes) will be substituted into the channel pattern.*
2. *The dependent data (channel elevations) will be used as the independent data of the vector created by the command.*

*The third argument is another data channel pattern. This time, the generated vector assigned the name **<void1>** will have the following values:*

Independent	Dependent	Retrieved From
0.1	0	alpn-21A01 at time 120.0
0.3	0	alpn-21A02 at time 120.0
0.5	5.99E-003	alpn-21A03 at time 120.0
...		
3.9	0.29	alpn-21A20 at time 120.0

18. Examine the next set of commands:

```
CALC "<void1> = flipXY( <void1> )"
CALC "<void2> = flipXY( <void2> )"
CALC "<void3> = flipXY( <void3> )"
CALC "<void4> = flipXY( <void4> )"
CALC "<void5> = flipXY( <void5> )"
```

*This command is used to flip the independent and dependent columns of the **void** vectors so that the elevations are plotted along the Y-axis and the void fraction values are plotted along the X-axis. The command does not directly modify a vector, hence why the resulting vectors are assigned back to the same names.*

19. Examine the next set of commands:

```
PLOTVAR "<void1>"
PLOTVAR "<void2>"
PLOTVAR "<void3>"
PLOTVAR "<void4>"
PLOTVAR "<void5>"
```

This command plots the indicated vector on the current graph in the plot.

20. Examine the next set of commands:

```
TITLE "Simple Standpipe Problem"
SUBTITLE "Void Fraction"
XAXIS LABEL "Void Fraction"
YAXIS LABEL "Elevation (m)"
```

These commands are fairly self-explanatory: they define the title and subtitle of the graph, as well as the labels displayed along the X-axis and Y-axis.

21. Examine the next set of commands:

```
S0 LEGEND "120.0 s"
S1 LEGEND "150.0 s"
S2 LEGEND "200.0 s"
S3 LEGEND "250.0 s"
S4 LEGEND "300.0 s"
```

*These commands define the legend entries displayed for each data set created by the **PLOTVAR** commands above. The **S#** label is a qualifier used to indicate which set is being edited by the following command. For example, **S0** indicates the first set of the current graph.*

22. Examine the remaining commands:

```
REDRAW
HARDCOPY DEVICE "PDF"
PRINT TO "<HOME>\VoidProfile.pdf"
PRINT
```

*These commands are used to redraw the graph with all modifications made by the previous commands, then export a PDF file depicting the graph to a location on disk. Take note of the **PRINT TO** command. Once again, this command will have to be edited.*

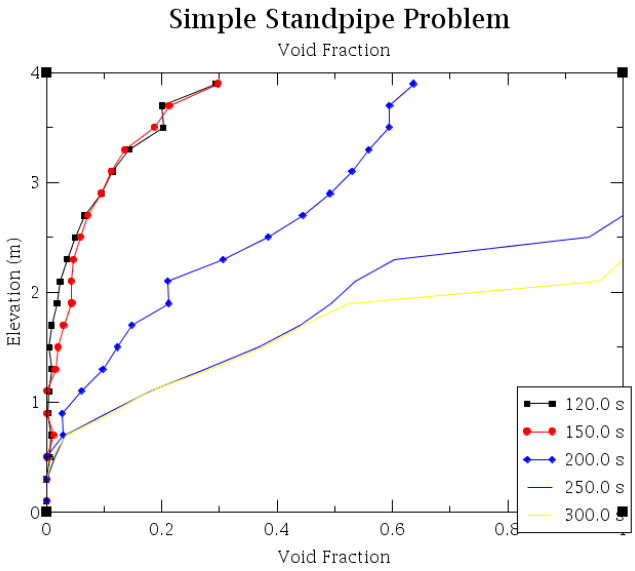
23. Select the **PRINT TO** command.

24. Replace the **<HOME>** portion of the command with the path of a convenient location to generate the PDF.

    Having adjusted the script, it can now be executed.

25. Press the **Replay** button in the **Commands** dialog.

    This button executes every command in the command list. It will take a moment to complete. Once finished, the plot should appear similar to the following:



26. In Windows Explorer, navigate to the directory specified previously for the generated PDF file and open the "**VoidProfile.pdf**" file.

    The generated PDF should match the graph displayed in AptPlot.

27. Close the PDF viewer and AptPlot.