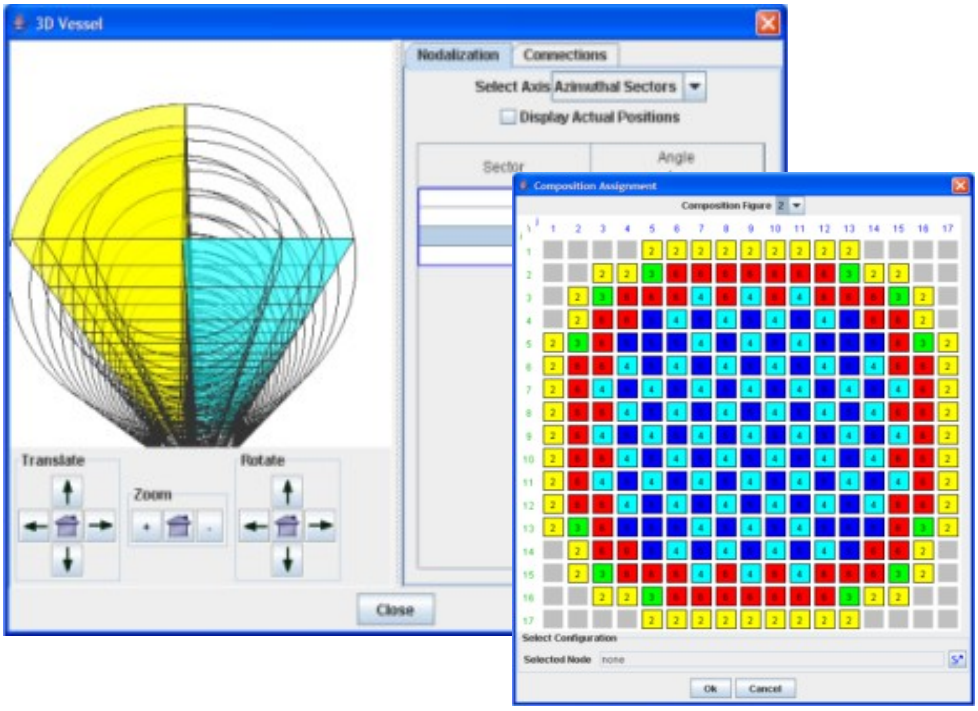
Symbolic Nuclear Analysis Package (SNAP)
SNAP/TRACE User Workshop
SNAP Afternoon Exercises

June 2014



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:

- Job Streams – an extensive set of tools for designing and executing application work flows.
- User-Defined Numerics – 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.
- Importing an externally completed job.
- 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.

Exercise 8. Introduction to Job Streams

This exercise introduces the basics of building job streams in the SNAP Model Editor. A stream consisting of a single TRACE step will be submitted to the local calculation server. Afterward, the job will be monitored from the SNAP Job Status application.

The following steps set up a **runs** folder on the Local Calculation Server. This is the location in which streams will be run through the exercises. If the SNAP configuration for the Local Calculation Server already contains a suitable submit location, skip the next six steps, and use this folder whenever the exercise refers to the **runs** directory.

1. Open Job Status. On Windows, select from the Start menu: **All Programs → SNAP → Job Status**.
2. In the tree to the left, expand the **Local** node. If a prompt appears asking whether to start the Calculation Server, press **Yes**.

   Expanding this node connects to the Local Calculation Server.
3. At the **No Root Folders** prompt, press **Yes**.

   A file browser will appear. This is used to select a location on the server where streams can execute.
4. In the file browser, navigate to a suitable run directory or create one. Press the **Mount** button once the folder is selected.

   A prompt will appear, asking for the name of the new mount point.
5. Enter the name "**runs**" for the folder and press the **OK** button.

   *The **runs** folder will appear under **Local**.*
6. Close Job Status.

The following steps build the simple stream and then submit it.

7. Open the Model Editor. On Windows, select from the Start menu: **All Programs → SNAP → Model Editor**.

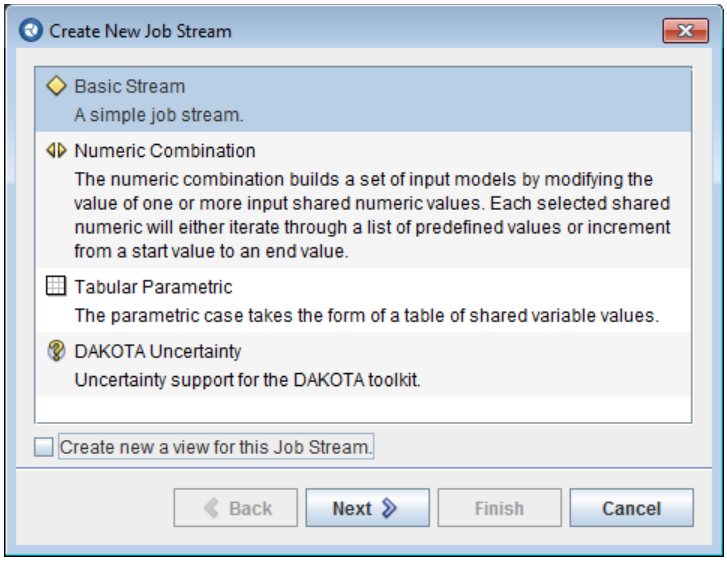
   The Model Editor will start up and display a welcome screen. Note that the welcome screen can be disabled and may not appear.
8. On the welcome screen, select the **Open a Model Document → Continue** >> option.

   *A file selection window will appear, prompting for the model to open. If the welcome screen was not displayed, select "**File → Open**" from the main menu.*
9. Open the standpipe sample model provided with this exercise. This file is located at "**SNAP_Exercises/StandPipe3.med**"

   After a moment, the model will be completely loaded and its primary view will be displayed. This is the model that will be submitted to the calculation server, which is covered in the next several steps.

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

    *The **Select Stream Type** dialog is displayed, as shown below.*



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

    A 2D View will be created to display the job stream in later steps.

12. Select **Basic Stream** in the stream type dialog and press the **Next** button.

    The basic stream type is the simplest of the stream types and does not allow parametrics of any sort. The other stream types are used to create parametric streams, a type of stream where the model is executed multiple times with some type of iterative modifications between each task. Parametric stream types are explored in a later exercise.

    The list of predefined job streams is displayed. Predefined job streams are used to speed the process of creating simple job streams for newly created or newly imported models. This exercise will start with an empty stream in order to examine the job stream creation process mode closely.

13. Select **An Empty Stream** in the stream type dialog then press the **Finish** button.

    A new, basic job stream is created and selected in the Navigator.

14. Set the following properties in the newly created stream:

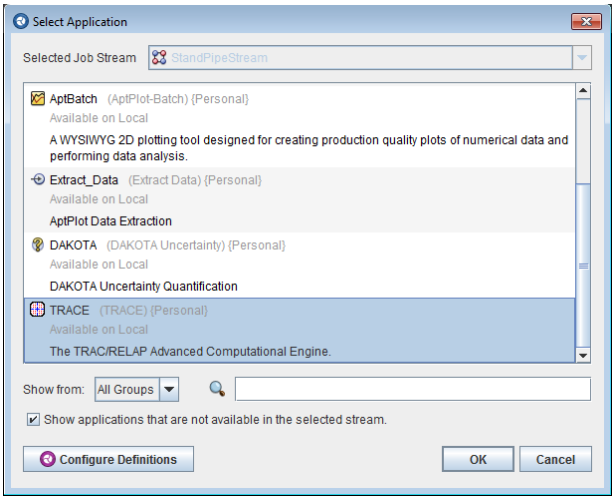
    Name: **StandPipeStream**
    Platform: **Local**
    Root Folder: **runs**
    Relative Location: **TRACE/**

15. In the Navigator, expand **StandPipeStream**.

    *A set of child categories are shown, including **Stream Steps** and **Model Nodes**.*

16. Right-click on the **Stream Steps** category and select **New** from the pop-up menu.

    *The **Select Application** dialog is displayed, as shown below.*



17. Select the appropriate **TRACE** application from the available list and press the **OK** button.

    A new TRACE step is added to the stream. This is the step that will run the TRACE input represented by the model.

18. Set the following properties on the new TRACE step:

    Name: **StandPipe**
    Interactive Step: **On**
    Start Paused: **On**

    ***Interactive Step** and **Start Paused** will ensure that the TRACE job is run in interactive mode and, after a short initialization phase, waits before executing.*

19. Expand the **Model Nodes** category in **StandPipeStream**.

    *A single "model node", **TRACE model 1**, will appear in the category. This node represents the StandPipe model.*

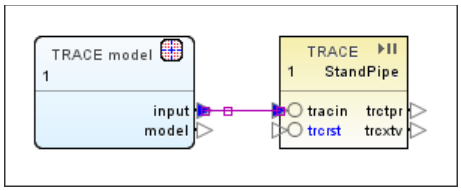
20. Right-click on **StandPipeStream** and select **Create View** from the pop-up menu.

    A view will be created with the model node and the TRACE step.

21. Reposition the elements of the new view so that the blue model node is to the left of the red TRACE node.

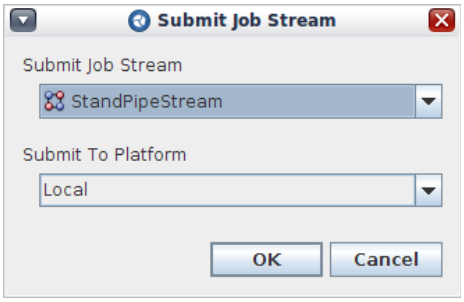
22. Using the Connection Tool in **StandPipeStream View**, connect the job stream model's **input** point to the **StandPipe** step's **tracin** point.

    The stream step will change color from red to yellow, indicating that its required inputs have been connected.



    This simple process represents the most fundamental concept of job streams: connecting the outputs *of a model, file, or step to the* inputs *of another step. When this stream is executed, the input created by the stream model will be run by the specified TRACE application as its* ***tracin*** *input.*

23. Select "**Tools → Submit Job**" from the main menu.



    The stream submission dialog is displayed, as shown.

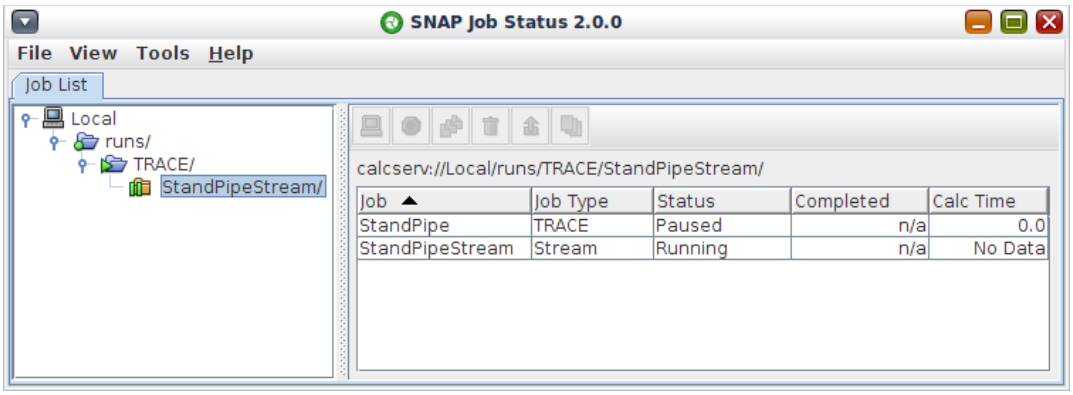
24. In the stream submission dialog, make sure that **StandPipeStream** and **Local** are selected, then press the **OK** button.

    The job will be submitted to the Local Calculation Server. Job Status will appear shortly thereafter, and is described below. Once the stream begins execution, it will perform its necessary initialization, then pause until told to proceed.

The following steps introduce Job Status. This application provides functionality for accessing and monitoring streams submitted to a calculation server. In particular, the next several steps will examine the paused TRACE task created above and resume its calculation.

25. Wait for Job Status to appear after submitting the stream. If Job Status does not appear, select from the Start menu: **All Programs → SNAP → Job Status**.

    Job Status is shown in the image below. On the left is a list of known Calculation Servers and their mounted-folder hierarchies. The area on the right is a list of the jobs residing in the selected folder.



26. Expand **Local**, then the **runs** folder, then the **TRACE** folder, and select the **StandPipeStream** folder.

    Note that this folder is a different color than those above it. This indicates that the directory was created specifically to house the contents of the stream. Selecting the stream shows the tasks running in that stream in the table to the right. Notice that ***StandPipe*** *is listed and that its status is* ***Paused****.*

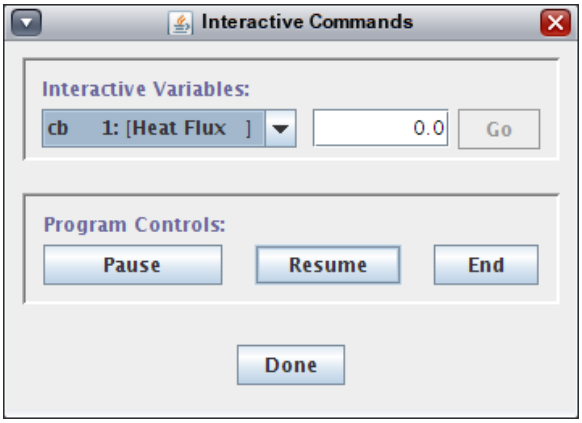
This final series of steps will issue an interactive command to the TRACE task, resuming its calculation. The interface for this process is the **Interactive Commands** dialog, which allows pausing, resuming, and completely halting jobs. The dialog also allows changing the values of interactive variables.

Note: Terminating the run outside of the Interactive Commands dialog terminates the process, which may produce corrupted output files.

27. Select **StandPipe** row in the table to the right.

28. Press the **Interactive Commands** button () on the toolbar above the job list.

    The Interactive Commands dialog is displayed, as shown in the figure to the right.



29. Press the **Resume** button.

    The ***Status*** *of* ***StandPipe*** *changes from* ***Paused*** *to* ***Interactive****, and the calculation resumes.*

30. Close the Interactive Commands dialog by pressing the **Done** button, then wait for the job to complete.

31. Close Job Status.

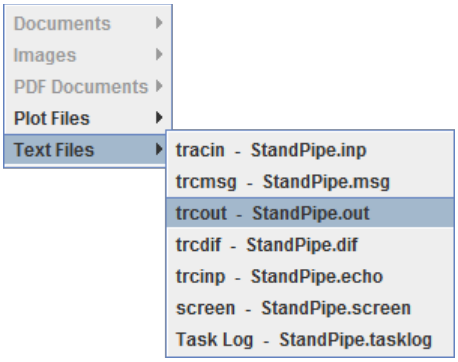
Exercise 9. Working with Output Files

This exercise describes how to use Job Status to view calculation results. The following steps describe how to use the File Viewer to examine the results of the **StandPipe** job submitted in a previous exercise.

1. Open Job Status. On Windows, select **All Programs → SNAP → Job Status** from the Start menu.
2. In the Job List, expand **Local**, and select the **/runs/TRACE/StandPipeStream** folder.

   This folder contains the results of the standpipe stream submitted in a previous exercise.
3. In the table, select the task named **StandPipe**.
4. Press the **View Output** button ( ).

   *This will open a pop-up menu of available files related to the job. Files are broken down into several types, as shown in the image below. Selecting any items in the **Text Files** menu will open the Output Viewer, described below.*



   *The TRACE **StandPipe** job provides several files which can be opened by the File Viewer. Some notable files include:*

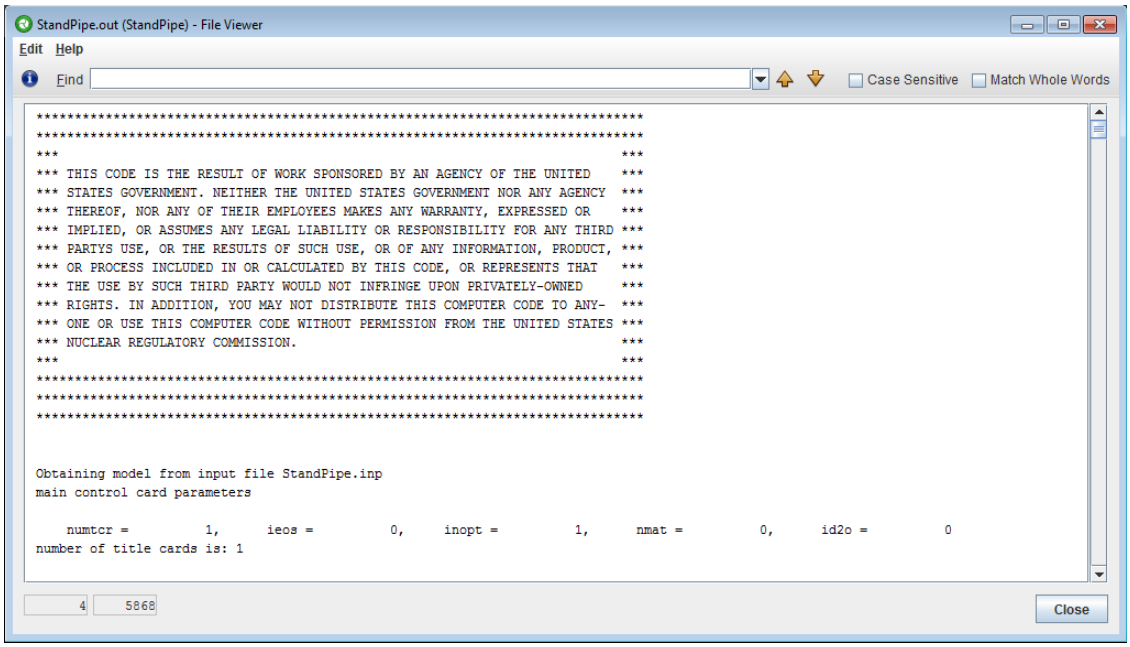
   ***tracin - StandPipe.inp**. The input file submitted to the TRACE executable.*

   ***trcout - StandPipe.out**. The results of the calculation in ASCII form.*

   ***trcmsg - StandPipe.msg**. Condensed output on the behavior of the numerical calculation and warning messages.*

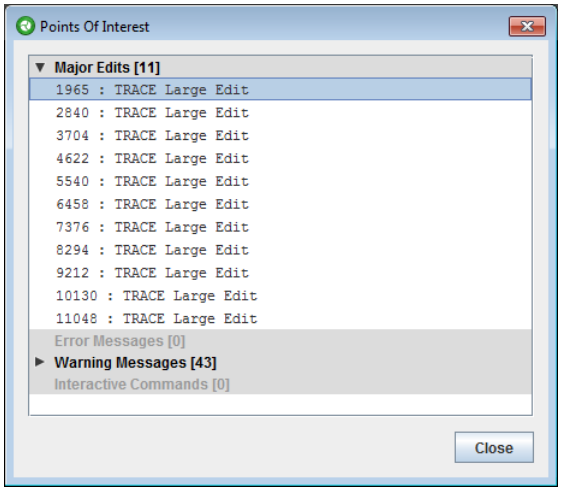
   ***screen - StandPipe.screen**. A collection of messages written to the console during the run.*
5. From the pop-up menu, select the **trcout** entry.

   The File Viewer will appear and display the contents of the file. The File Viewer was designed to allow very large text files on a remote Calculation Server to be viewed and searched over a relatively slow connection.



6. Press the Points of Interest ( ) button.

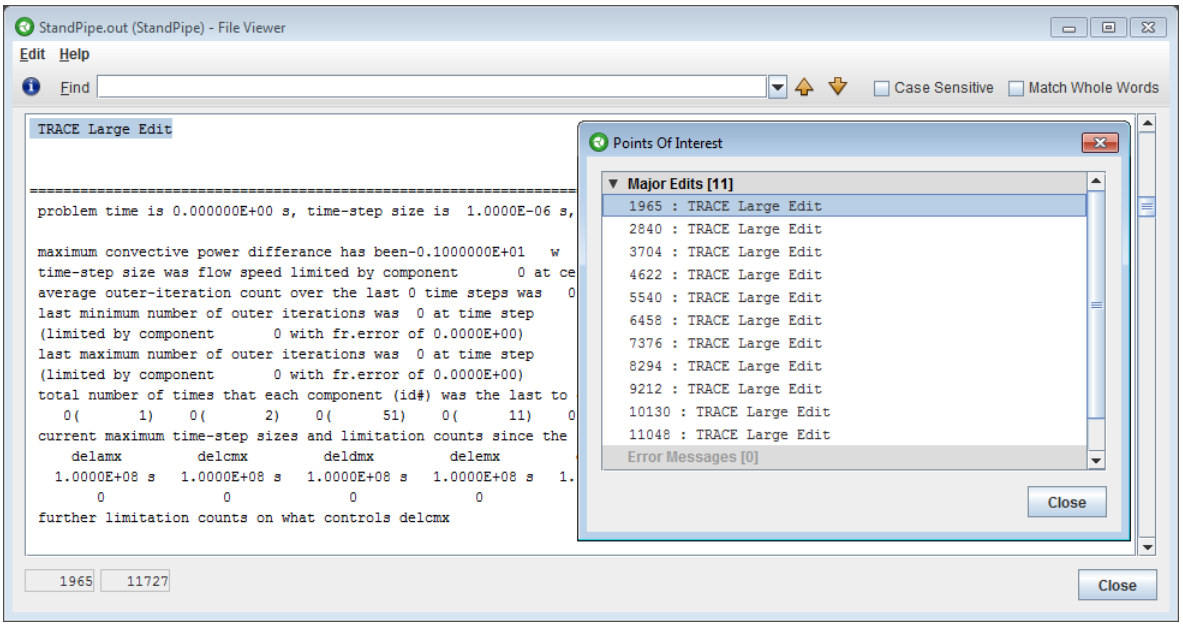
   *The **Points of Interest** search dialog will be displayed. The points of interest in the file are separated by type into expandable sections. Expanding each section will show the list of points of interest of that type in the file and the line number on which is occurs.*



   For TRACE output files the points of interest include major edits, error messages, warning messages, and interactive commands sent from SNAP.

7. Click on the **Major Edits** row to expand the list of major edits.

   This will expand the Major Edits row to display the list of major edits in the file and will automatically select the first major edit in the list.



Notice that the File Viewer is now positioned at the beginning of the first Major Edit. Each of the provided Points of Interest items will navigate to the point in the output where the item can be found.

8. Close the **Points of Interest** window.

The remaining steps demonstrate the other available search functions: Find and Goto.

9. Click in the **Find** field at the top of the File Viewer.

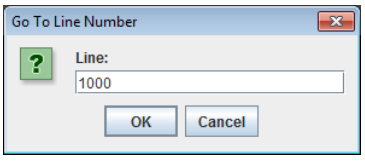
   The Find field can be used to search for arbitrary text in the file.

10. Enter the word "edit" and press Enter.

    The File Viewer will scroll to the next location in the file that contains the text "edit". Pressing Enter again will repeat the search forward from the current location. The "Find Next"( ) and "Find Previous"( ) buttons can be used to search forward and backward in the file. Previously entered search terms can be selected from the drop-down list to the right of the Find field.

11. Select the **Edit → Goto** main menu item.

    This will open the Go To Line Number window. This window is used to scroll the output to a specific line.



12. Enter "**1000**" for the Line and press **OK**.

    The dialog will close and the File Viewer will scroll to line 1000.

13. Close the File Viewer.

Exercise 10. Restart Editing

This exercise describes how to build restart cases in a Job Stream. These steps build on the stand pipe model built in previous exercises. The **StandPipe** job will be used as a source for initial conditions. A new restart case will be create that contains a model end time change, a pipe renodalization, and a set of retrieved initial conditions.

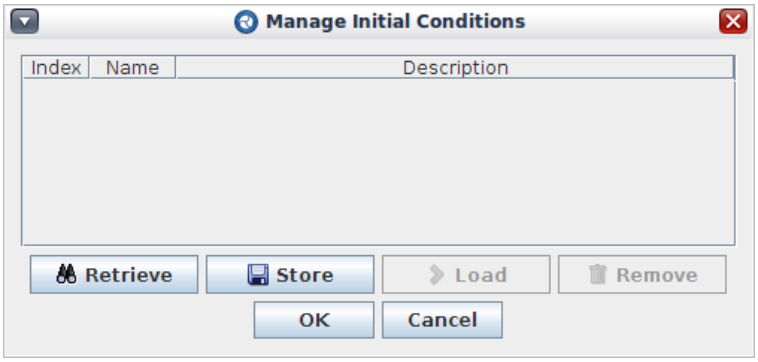
The following steps will guide you through the exercise.

1. Open **SNAP_Exercises/StandPipe4.med**, included with these exercises, and make the following modifications:
   - 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).
   - Select the TRACE step in **StandPipeStream** and make sure its **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. In the Navigator or 2D View, select **Pipe 2** and open its ASCII View.

   ***Pipe 2** is located in the Navigator under the **Hydraulic Components → Pipes** category.*
3. Right-click on the **StandPipe4.med** model node in the Navigator and select **Manage Initial Conditions** from the pop-up menu.

   The Manage Initial Conditions dialog displayed below is opened. This dialog is used to retrieve, store, load, and discard sets of initial conditions retrieved from completed jobs.



4. Press the **Store** button.

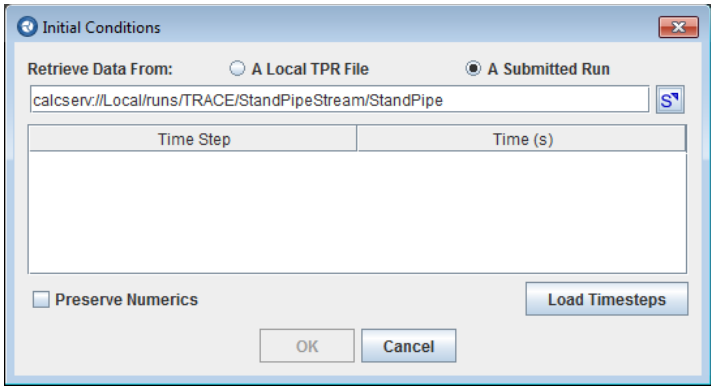
   *A new row is created in the table, representing a set of initial conditions that have just been stored for later use. The contents of this initial conditions set were pulled from component values in the model. At any time in the future, the model's initial conditions can be reset to these values by selecting this row and then pressing the **Load** button.*

5. Double-click in the **Name** column of the newly created **unnamed** row and change the name to "**Original**".

   *The **Name** and **Description** of the row are the only means of indicating the contents of the initial conditions, so it is important that they accurately describe the row.*

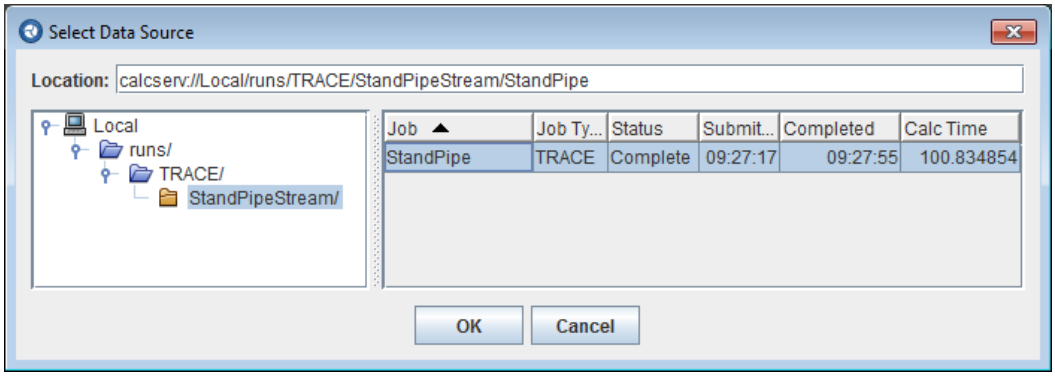
6. Press the **Retrieve** button.

   *The **Initial Conditions** dialog will appear, as displayed below. This dialog is used to retrieve initial conditions from completed calculations.*



7. Select the **A Submitted Run** radio button.
8. Press the **S** to the right to select the job from which to retrieve initial conditions.

   This will open the job selection dialog shown below.



9. Expand **Local**, then the **runs** folder, then the **TRACE** folder, and select the **StandPipeStream** folder.
10. Select **StandPipe** in the table to the right and press **OK** to close the dialog.

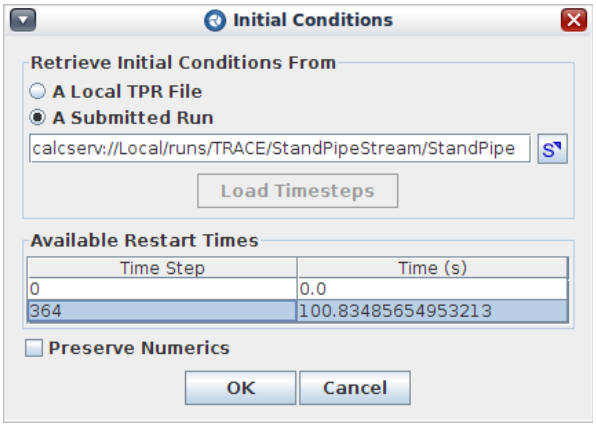
    The previously submitted run is now the source of data from which initial conditions will be retrieved.

11. Press the **Load Timesteps** button.

    *Two **Available Restart Times** will become available.*

12. Select the last restart time in the list: Timestep **364** at **100.834** s.
13. Press the **OK** button.

    *This will retrieve the initial conditions at the last timestep and assign them to each component in the model. Notice the initial conditions of **Pipe 1** displayed in the ASCII view have changed to reflect the data imported from the plot file.*



14. Press the **Store** button.

    *As with storing the **Original** initial conditions, the initial conditions currently in the model will be represented by a new, **unnamed** row in the table.*

15. Double-click in the **Name** column of the newly created **unnamed** set and change the name to "**Steady State**".
16. Select the **Original** initial conditions row and press the **Load** button.

    *The contents of the stored **Original** conditions will be loaded into the model. The **Steady State** contents will be used in a later step.*

17. Press the **OK** button in the Manage Initial Conditions dialog to close the dialog.

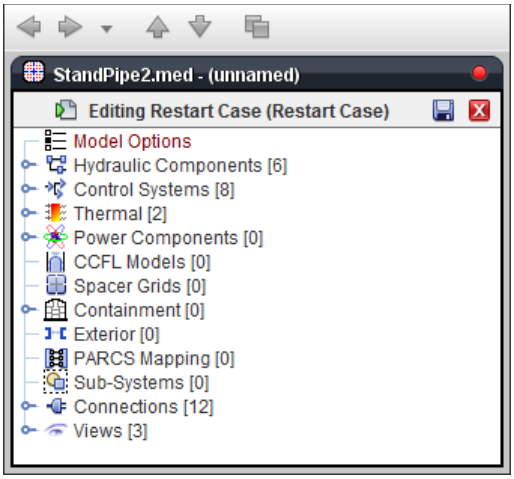
With the initial conditions set, the following steps edit the model as a restart.

18. Create a new restart case in the **Cases** category.
19. Select the **Initial Conditions** by pressing the **S** button in the editor.

    A selection dialog will appear. This is used to select a set of initial conditions that the restart case will be based on. This has the effect that the selected initial conditions will be loaded automatically when editing a restart case in a virtual model (as shown in the next several steps).

20. Select **Steady State** and press the **OK** button.
21. Set the **Editing Mode** property of the new case to "**Graphical**".
22. Begin editing the new Restart Case by pressing the **E** button for the **Restart Model** property.

A virtual model will be created, where the restart changes are made. Notice the red dot in the model node and the Restart Panel under it. The red dot is used to indicate models currently in Restart Editing mode and is used for all plug-ins that support restarts, while the panel is used to indicate that a restart case is being edited.

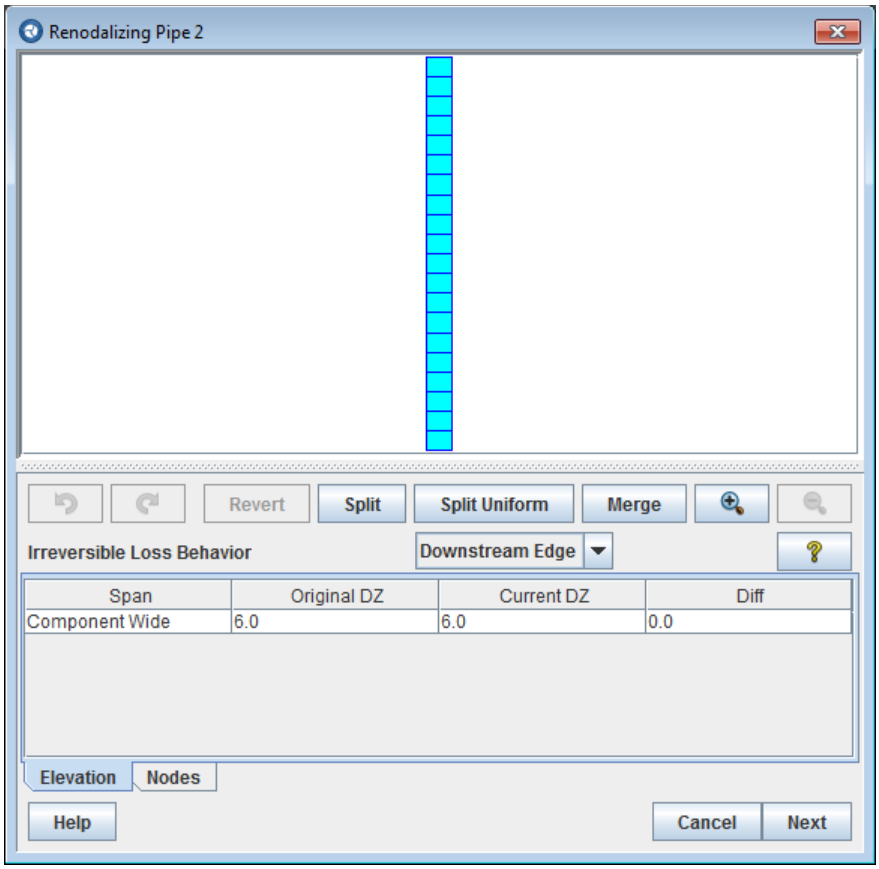


23. Select **Model Options** in the Navigator.
24. Press the **E** to the right of the **Timestep Data** property to open the timestep table.

    *The **Timestep Data** attribute is near the bottom of the **General** attribute group in the Property Editor.*

    *The **Edit Timestep Data** dialog will be displayed. The central table of this dialog can be used to add, edit, and remove timestep data for the model.*
25. Change the **End Time** from **100** seconds to **1000** seconds, then press **OK**.
26. Select **Pipe 2** in the View or in the Navigator.
27. Right-click on **Pipe 2** and select **Renodalization → Renodalize** from the pop-up menu.

    *The **Component Renodalization** dialog is displayed, as shown. As its name suggests, this dialog is used to change the nodalization of the pipe. In this case, cells 1 through 20 will be split uniformly into two cells each.*



28. In the renodalization dialog, select cells 1 through 20.

    To select the cells, click on the first cell in the top of the dialog, then hold the SHIFT key and click on the last cell. The selection should match the image above.

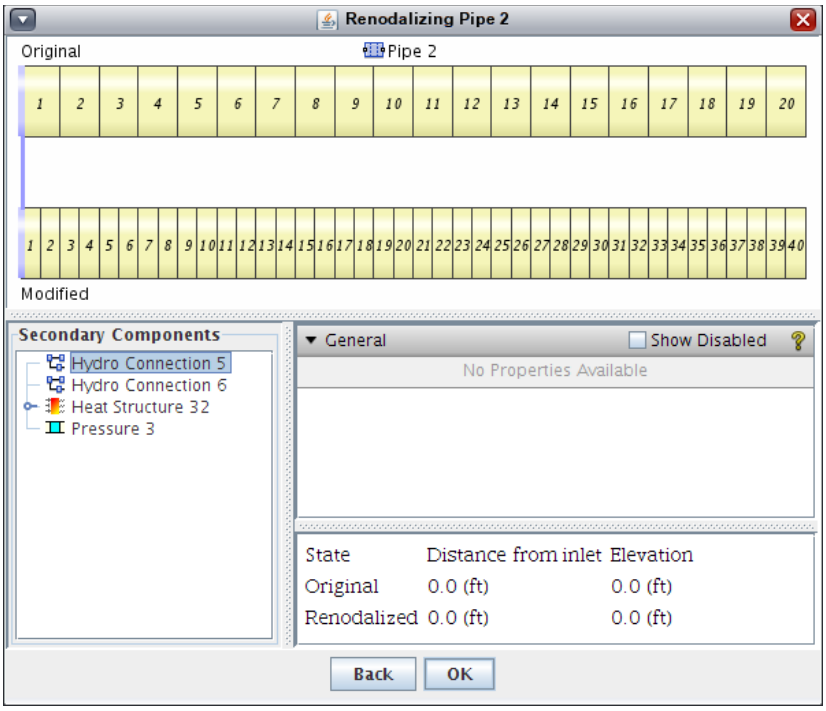
29. Press the **Split Uniform** button.

    A prompt is displayed, requesting the number of cells each selected node should be split into.

30. Type "**2**" at the prompt at press the **OK** button.

    The contents of the dialog are updated to indicate the impending renodalization. Note that the 2D representation displays the new cells. The table display below shows any change in elevation introduced by the renodalization.

31. Press the **Next** button to continue the renodalization process.



This will continue to the renodalization results display. The effect of the renodalization on other components in the model can be seen in this dialog before committing to the change.

32. Press the **OK** button to complete the renodalization.

    This will open a Renodalization Report including detailed descriptions of the effects that the renodalization had on the model.

    *In **Default View**, notice how the 2D view is automatically updated. Any components modified for the restart are highlighted red in the Navigator. Notice that **Pipe 2** and **Heat Structure 32** have restart edits, indicated by red names in the Navigator.*

33. Press **Close** to close the Renodalization Report.

34. In the Restart Case panel, press the **Save Case** button ( ).

    The changes made in the virtual model are imported as a complete restart case.

35. In the Navigator or 2D View, select the **Restart Case** and open its ASCII View.

    An ASCII Viewer is displayed, showing the contents of the restart case.

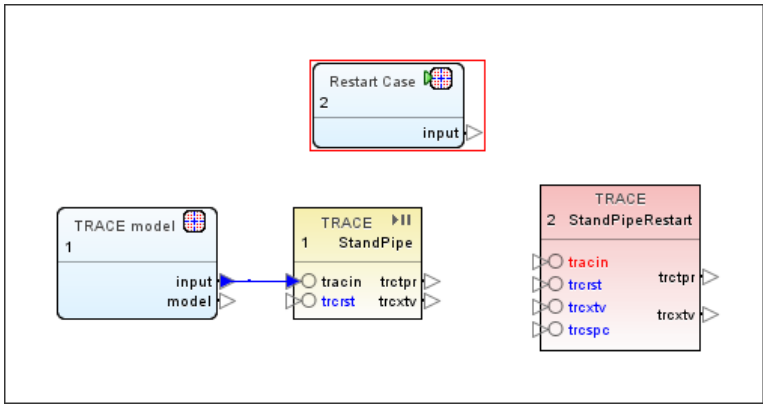
36. Close the ASCII Viewer by pressing the **Close** button.

Now that all intended restart edits are complete, the next several steps submit the model as a restart of the previous job.

37. Open the **StandPipeStream** view if it is not already open.
38. Expand **StandPipeStream** in the Navigator, and add a new **Model Node**.
39. With **TRACE model 2** selected, select the check box in the **Restart Case** property editor.

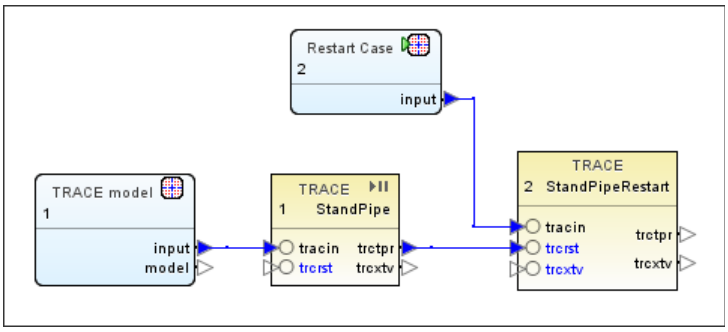
    The property is now enabled, indicating that this model node represents a restart case.
40. Press the **S** button in the **Restart Case** property editor, and select **Restart Case** in the selection dialog.

    *This model node now represents a specific restart, as indicated by its name changing to **Restart Case 2**. It will be used as the basis of a restart TRACE step in the job stream.*
41. Press the **OK** button to select the restart case.
42. Create another TRACE **Stream Step**, as described in the previous exercise.
43. Name the new step "**StandPipeRestart**".



44. Drag the new **StandPipeRestart** step from the Navigator into the "**StandPipeStream View"** to the right of the **StandPipe** step.
45. Drag the "Restart Case" model node from the Navigator into the "**StandPipeStream View"** just above the **StandPipe** step.

46. Using the Connection Tool, connect the **input** point of **Restart Case** to the **tracin** point of **StandPipeRestart**.
47. Connect the **trctpr** output of **StandPipe** to the **trcrst** input of **StandPipeRestart**.



*The restart stream is now complete. When this stream is run, the **StandPipe** task will be run as before, using the base model. Once the **StandPipe** job is complete, the **StandPipeRestart** task will run TRACE with the given restart case as its input and the TPR file created in the previous task as its restart input.*

48. Select the **StandPipe** step and set its **Start Paused** property to **Off**.
49. Select the **StandPipeRestart** step and set its **Interactive Step** and **Start Paused** properties to **On**.

    ***Start Paused** will not be available until **Interactive Step** is set to **On**.*
50. Right-click **StandPipeStream** in the Navigator and select **Submit Stream to Local** from the pop-up menu.
51. Press **OK** to confirm the stream submission.

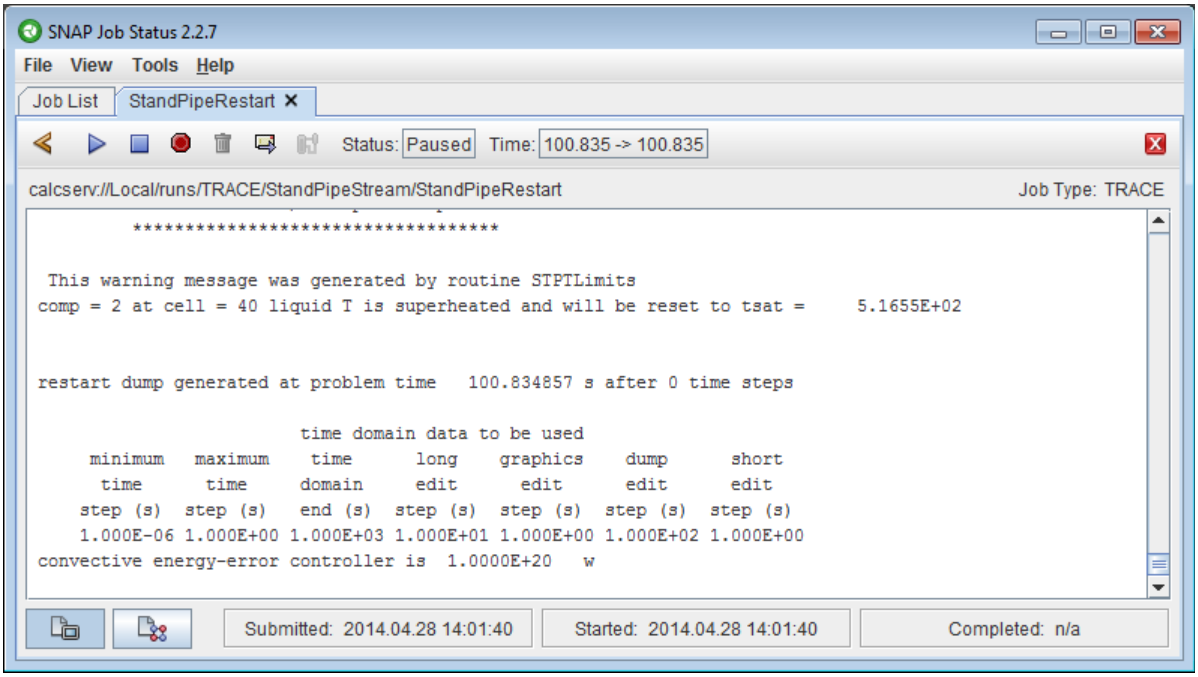
    The stream is submitted, and Job Status will be displayed after the stream starts.
52. In Job Status, expand **Local**, then the **runs** folder, then the **TRACE** folder, and select the **StandPipeStream** folder.

    The submitted stream steps will be displayed in the table to the right.
53. Wait for the **StandPipe** task to complete, then for the **StandPipeRestart** job to initialize and switch to the **Paused** status.

    Note the distinction in terminology here. When creating a stream in the Model Editor, applications added to a stream are referred to as "steps". When actually running the stream, these application runs in the stream are referred to as "jobs" or "tasks".

54. Select **StandPipeRestart** in the table.
55. Press the **Job Console** button (🖳) in the Job List toolbar, or double-click **StandPipeRestart** in the table.

    *The job console for **StandPipeRestart** will open in a new tab, as shown below. A job console will display the console output of a task and can be used to issue commands to interactive jobs, terminate the task, open local files, etc..*



56. Press the **Play** button (▶) in the job console toolbar and wait for the task to complete.
57. Close Job Status.
58. Select **File → Close All** from the main menu.
59. Press the **Discard All** button.

Exercise 11. User-Defined Numerics and Parametrics

This exercise begins with a an example of the user-defined numerics functionality. Numerics provide a means of assigning variables to specific model inputs. This provides a great deal of flexibility: a single value can be referenced in multiple places, edited in a 2D View, adjusted by user-defined functions, and varied for parametric submit.

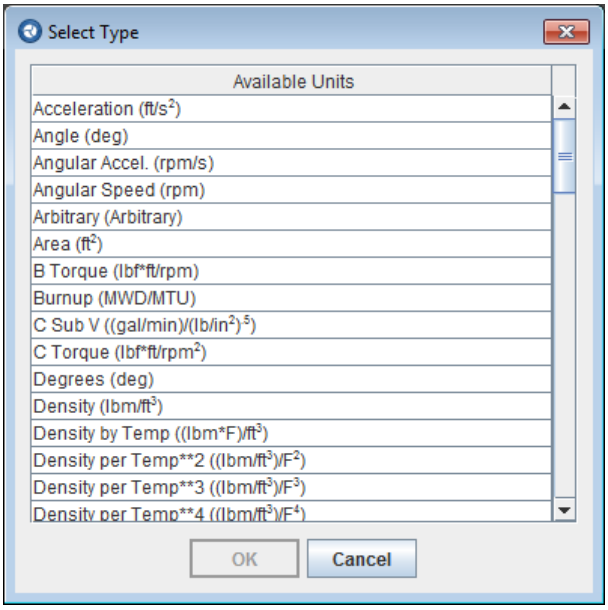
The first series of steps will create a user-defined numeric and a reference to it.

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 numeric. Real numerics can only be referenced from values with a matching unit type.*

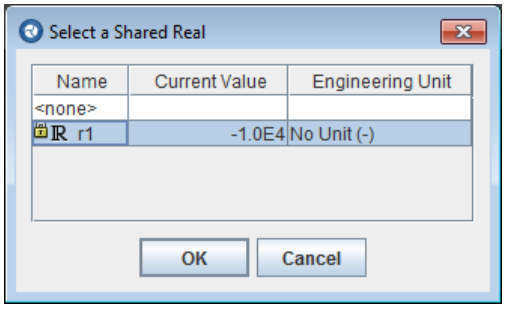


4. Select the "**No Unit (-)**" type, then press the **OK** button.

*A new Real numeric named **r1** is created and selected in the Navigator. Note the **Unknown** in the **Value** property: this numeric has no value by default.*

5. Set the **Value** of the new numeric to "**-1.0E4**".
6. In the Navigator, expand the **Control Systems** category, then **Control Blocks**, then **Controller Blocks**.
7. Select **Interactive Variable -1 (heatflux)**.
8. 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.*



9. 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.

10. Open and display **Default View**, if it is not already open.
11. 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.*

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

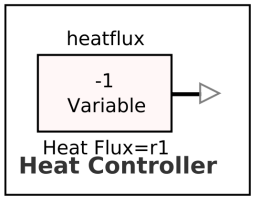
    *The value of **r1** displayed in the View will change to reflect the new variable value.*

13. Undo the previous change.
14. Right-click the **r1** drawn variable in the View to display its pop-up menu.
15. 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.*

16. 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.



17. 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 an animation will prevent inadvertent repositioning of components and will activate any interactive controls contained in the view. Editable drawn variables can only be modified in a locked view.

18. Left-click within the field.

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

19. Type in the value "**-5.0E4**" and press the Enter key.

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

20. Undo the previous change.

The next several steps demonstrate user-defined pythons 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 numeric 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 user-defined 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 user-defined functions are executed. It serves as a stable starting value, allowing functions to reproduce their results with each successive evaluation.*

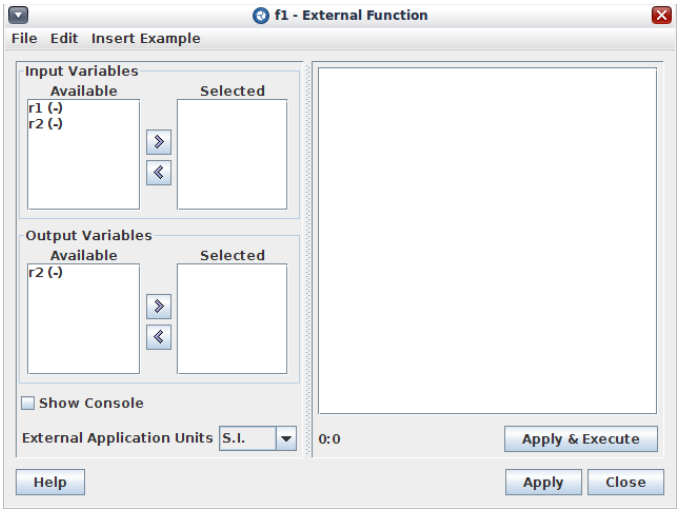
24. Create a new component in the **Functions** category.

*A new function, **f1**, is created and selected in the Navigator. The default 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/.

25. In the Property View, edit the **External Function** property by pressing the **E** button in the editor.

    *The **External Function** window will be shown. The next several steps use this dialog to define the user-defined function in its entirety.*



26. In the **Input Variables** panel, from the **Available** list, select "**r1 (-)**".

    *The **Input Variables** indicate which variable values can be retrieved by the function. All variables in the **Selected** list are available as a source of data.*

27. Press the **Add to Selected** button ( ) in the **Input Variables** panel.

    *The **r1** variable will be added to the **Selected** list.*

28. In the **Output Variables** panel, from the **Available** list, select "**r2 (-)**".

    *The **Output Variables** indicate which variables values can be modified by the function. Note that **r1** does not appear here. Only non-interactive variables can be modified, so only these components are shown.*

29. Add **r2** to the **Selected Output Variables**.

    The next several steps add the Python code that defines the function.

30. In the text area to the right, 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.*

31. 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**.

32. 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 numeric to the specified value, in this case the variable **y**.*

33. Press the **Apply** button, then the **Close** button.

The logic defined in the last several steps will be stored and the dialog closed.

34. Select the **Functions** category.

The properties for this sub-category allow executing all user-defined 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).

35. Press the **Execute Functions** button.

    A dialog will briefly appear, indicating the evaluation of the user-defined 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.

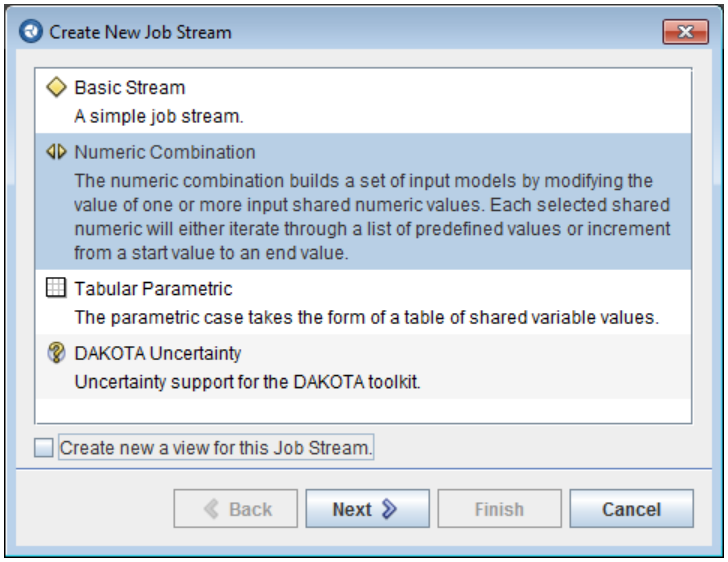
36. Select **r1** in the Navigator and note its **Value** of **-1.0E4**.
37. Select **r2** in the Navigator and note its **Value** of **1.0**.

    *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.

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

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



39. 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.

40. Select the **Numeric Combination** item and press the **Next** button.
41. Select **"An Empty Stream"** from the available options.
42. 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.*

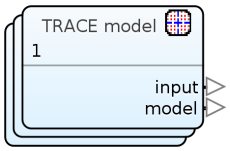
43. Enter the following values for the new stream:

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

44. Expand **ParametricStream** and create a new **TRACE** step.
45. Name the new step "**StandPipeParametric**".
46. Open and display **StandPipeStream View**.
47. From **ParametricStream** in the Navigator, drag both the model node and the TRACE step into the View.
48. Select ParametricStream model node, **TRACE model 1**, in the Navigator.

    *A new property, **Parametric**, will appear in the Property View.*

49. *Set the model node's **Parametric** property to "**True**".*

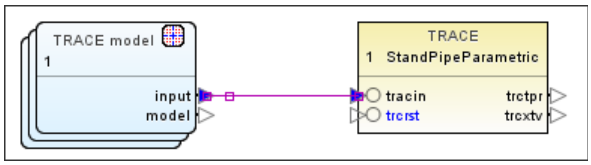
    

    Note that the model node for the parametric stream is rendered differently. This indicates a parametric model node that will export a set of inputs as part of a stream.

50. *Set the model node's **Parametric** property to "**False**".*

    *The model node in the view will now appear similar to the model node from **StandPipeStream**. A parametric stream can use non-parametric model nodes to export the base model without modification.*

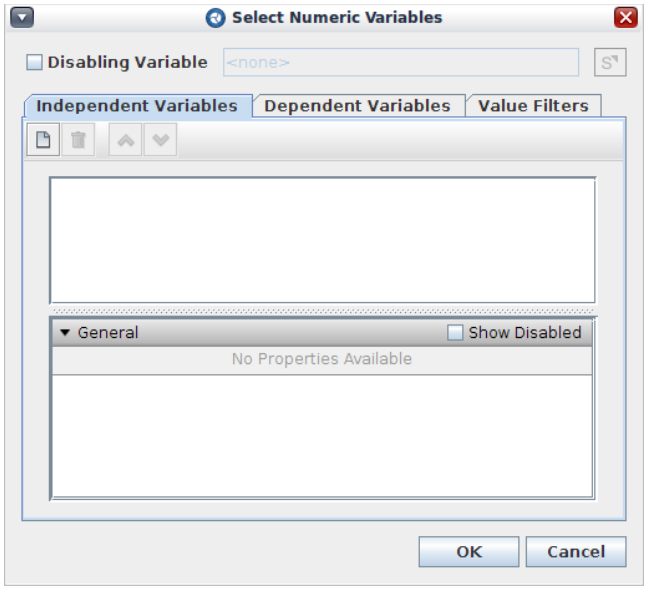
51. Undo the previous change.
52. Using the Connection Tool, connect the **input** point on the model node to the **tracin** point on the **StandPipeParametric** TRACE step.



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

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

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



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

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

56. 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.*

57. 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.*

    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.*

58. Select the **Dependent Variables** tab.

    Note that this tab does not have a property view.

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

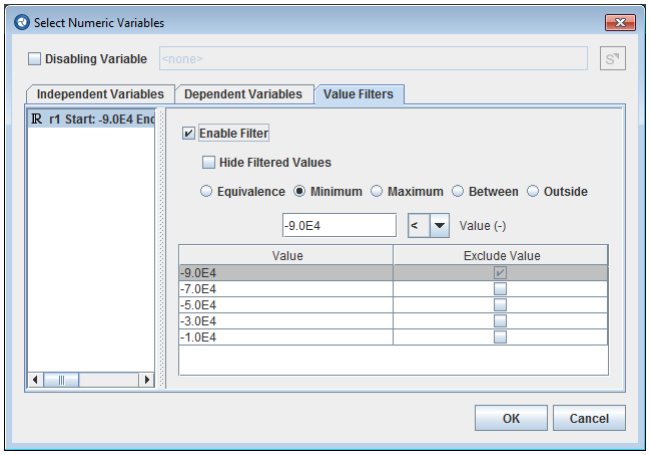
    Unlike an independent variable, a dependent variable is not used to define the parameters of the combination. Instead, its value will be sampled and added to the list of parametric keywords associated with each task. Each set of keywords acts as a signature of sorts for an individual parametric task.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

60. Select the **Value Filters** tab, then select **r1** in the list on the left.

This portion of the stream type configuration describes the process of filtering parametric tasks based on numeric conditions. The next several steps will filter out the -3.0e4 and -7.0e4 tasks.

61. Select the **Enable Filter** check-box

    *The rest of the filter configuration will now be available for the r1 numeric. Note that **-9.0E4** is highlighted in gray. This is because of the default filter. As it stands, the default filter can be read as: "disable tasks unless -9.0E4 is less than the value". As -9.0E4 is equal to the minimum value, it fails the filter criteria.*



62. In the drop-down next to the **-9.0E4** field, change the value from "<" to "<=".

    *The **-9.0E4** task will now pass the filter. This* equivalence *option determines whether the filter is inclusive or exclusive.*

63. Change the equivalence back to the exclusive option, "<".

64. Change the filter type to **Equivalence**.

*Only the **-9.0E4** task will pass the filter. Equivalence, as its name indicates, allows only tasks whose numeric value matches a specific value.*

65. Change the filter type to **Maximum**.

    *The **-1.0E4** task will fail the filter. This task is similar to the Minimum filter, setting a ceiling that values may not exceed.*

66. Change the filter type to **Between**.

    *Both the **-1.0E4** and **-9.0E4** tasks will fail the filter. **Between** essentially enables both a **Minimum** and **Maximum** filter.*

67. Change the filter type to **Outside**, and both equivalence values to the inclusive option, "**<=**".

    *Only the **-1.0E4** and **-9.0E4** tasks will pass the filter. The **Outside** filter only accepts values lying outside the specified range.*

68. Change the **Outside** filter parameters to the following:

    value (-) <= -3.5e4 OR -2.5e4 <= value (-)

    *The **-3.0E4** task will be disabled.*

69. Select the check-box in the **Exclude Value** column for the **-7.0E4** task.

    While the filter can be used to specify contiguous ranges of included or excluded tasks, an option is available for manually disabling specific tasks.

70. Press the **OK** button.

    The parametric properties are now set for the stream.

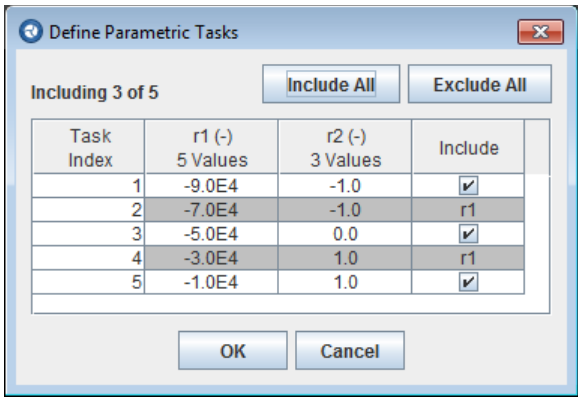
71. Select the TRACE step **StandPipeParametric** in the Navigator or in the View.

    *A new property, **Parametric Tasks**, will appear in the Property View. The value should indicate "**Including 3 of 5**".*

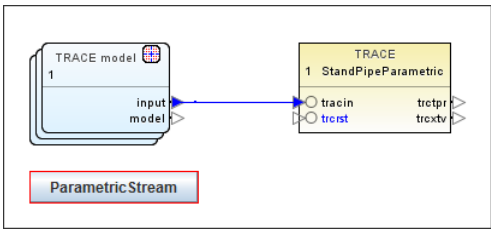
72. Edit the **Parametric Tasks** property by pressing the **E** button in the editor.

    *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. Note that the tasks for **r1** values **3.0E4** and **7.0E4** will be unavailable due to the filtering entered in the parametric properties.*

    *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.*



73. Press the **Cancel** button to close the dialog.
74. From the Navigator, drag ParametricStream into the **StandPipeStream View**, under the **StandPipeParametric** step.

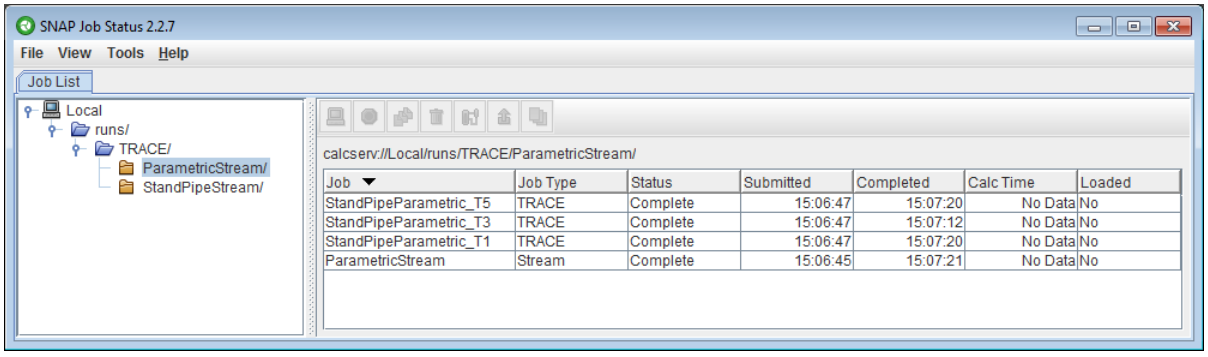


*A button titled **ParametricStream** is added to the View. This button can be used to submit the stream. Like other interactive controls, it can only be used when the View is locked.*

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

*After a moment, Job Status will appear and highlight the stream. The ParametricStream folder will contain three **TRACE** tasks: **StandPipeParametric_T1**, **StandPipeParametric_T3**, and **StandPipeParametric_T5**.*

77. Wait for the tasks to complete.



78. 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.

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

    *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.*
82. Close Job Status.
83. Select **File → Close All** from the main menu.
84. Press the **Discard All** button.

Optional Exercise: Create a **Tabular** parametric stream that exports the same values.

Exercise 12. 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.*

2. Expand the **Cases** category and select **Restart Case**.

3. Pres the **E** button for the **Restart Model** property to begin editing the case.

   *The restart case virtual model will be shown in the Navigator after a moment. This restart case edit will be discarded. The reason this is opened is to copy **Pipe 2** with the renodalization that will be animated in the next section of the exercise.*

4. If **Default View** is not open or displayed, open and select it now.

5. Unlock the View.

6. 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).

7. With all components in the view selected, right-click the selection and select **Copy** from the pop-up menu.

8. Create a new **Animation** model.

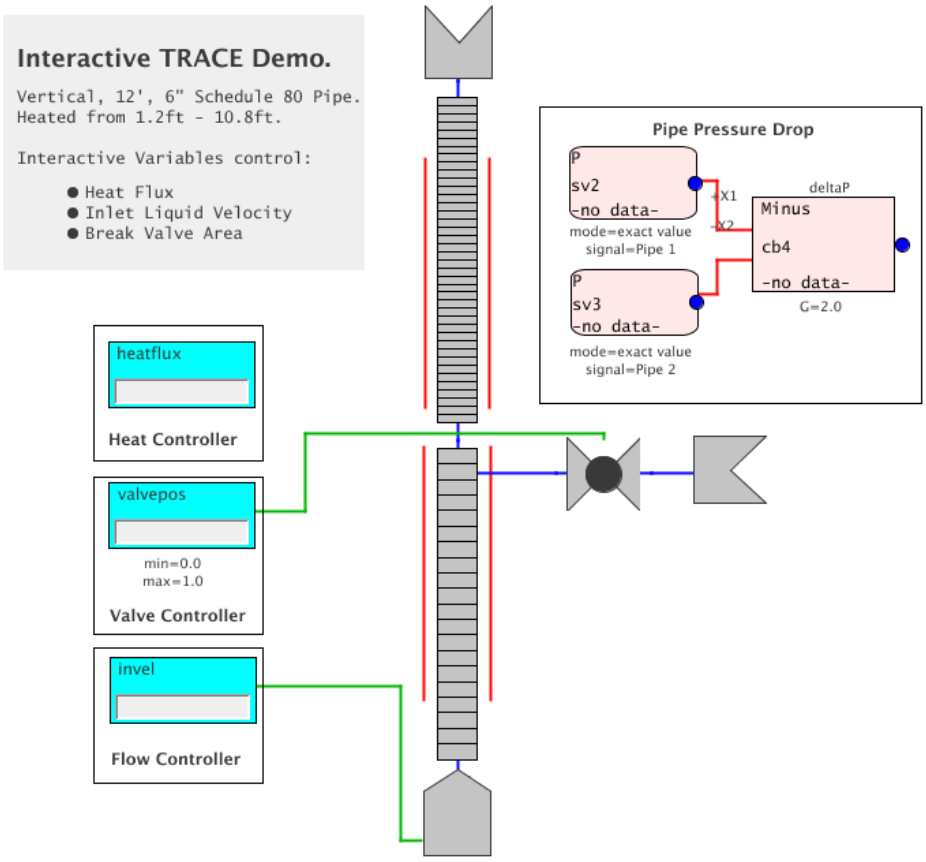
   *The steps are the same as creating a TRACE model, but select "**Animation model**" at the **Select Model Type** dialog.*

   *Once, the new model is created and displayed in the Navigator, a blank **Default View** will be displayed.*

9. 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.



10. In the Navigator, switch back to **StandPipe6.med**.
11. Press the **Close Case** button (X) in the Restart Case panel. Select **No** at the prompt to save changes.
12. 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.

13. In the Navigator, locate and expand the **Data Sources** node of the animation model.
14. Select the data source labeled "**Master (NewSource)**".
15. 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.*

16. Expand **Local**, then **runs**, then **TRACE**, then select the **StandPipeStream** folder.

    A list of calculations is displayed on the right.

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

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

18. 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 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.

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

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

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

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

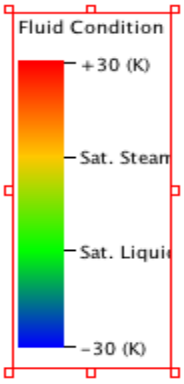
    SNAP uses Color Maps to translate component data and thermal-hydraulic conditions to color.

22. 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).

23. 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.



24. 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**

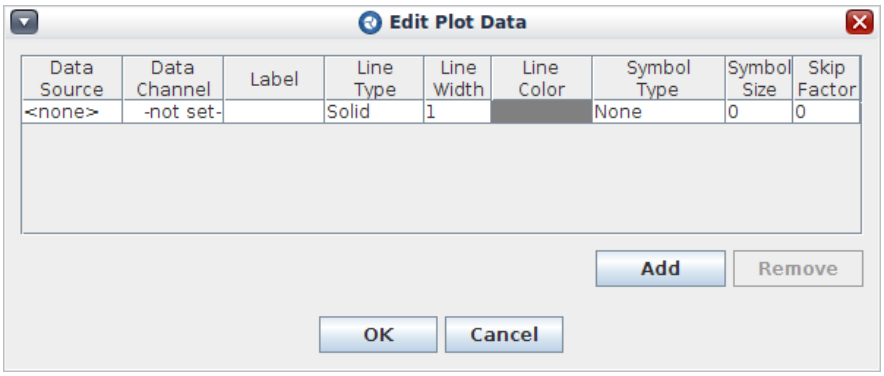
Data values use the format string standard from the C programming language for all numerical formats.

25. 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**

26. 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.*



27. 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.

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

    Data Source: **Master**
    Data Channel: **tln-2A40**
    Label: **Pipe 2**
    Line Color: **90,90,255** in the **RGB** tab.

29. Press the **OK** button.

30. Test the animation using the animation controls on the main menu bar.

    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.

31. Lock the animation using the **Lock** button.

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

32. Test the animation controls contained in the view.
33. Close the model.

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 13. 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:

   - 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).
   - 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. Under the **Cases** category, select **Restart Case**.
5. 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.

6. Select the Model Options node in the Navigator.
7. Edit the **Timestep Data** property by pressing the **E** button in the editor.
8. 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.

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

    This exercise will be animating the results of this step.

11. In the Property View, enable the **Animation Model** property by selecting its property editor check box.
12. Press the **S** button in the **Animation Model** editor.

    A file browser will open, used to select the model animated by this step when it runs.

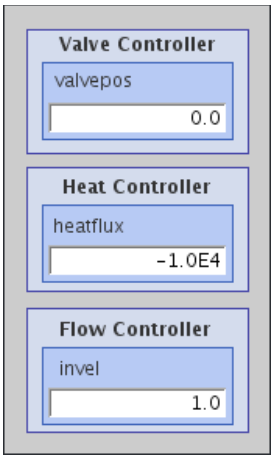
13. In the file browser, select the **StandPipe_Anim_interactive.med** file included with this exercise, then press the **Select** button.
14. Set the **Open Animation** property to **Immediately**.

    *Now, whenever **StandPipeStream** is submitted, the indicated animation model will be opened and automatically connected to the appropriate data source as soon as animation data is available.*
15. Submit the stream: right-click **StandPipeStream** in the Navigator and select **Submit Stream to Local** from the pop-up menu.
16. Press **OK** to confirm the submission.

    *Job Status will open. Once the **StandPipeRestart** initializes, the Model Editor will switch to the animation model and connect to the data source.*
17. Close Job Status.
18. Resume the calculation by selecting **Yes** at the prompt.

    This may need to be done twice: once to synchronize the animation with the calculation, and another time to resume it.

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.



19. 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.
20. Repeat the last step with a value of "**-1.0E4**".
21. 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.
22. Repeat the last step with a value of "**0.0**".

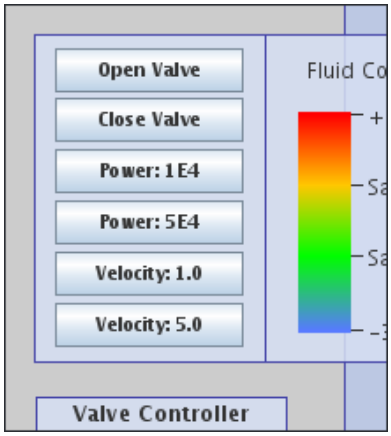
    The valve changes back to red, illustrating a close.
23. Enter "**5.0**" into the **Flow Controller** interactive variable and observe the results.

    The control block to the right of the field will slowly increase to the new flow vaue.
24. Repeat the last step with a value of "**1.0**".
25. 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.

26. Switch back to the **StandPipe6.med** model.
27. Resubmit **StandPipeStream**, and wait for the **Resume Calculation** prompt.
28. Resume the calculation, and wait for the **Void Profile** to stabilize.
29. 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.*



30. 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.

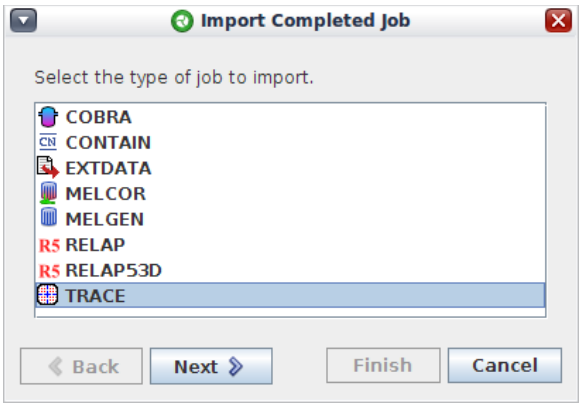
31. Halt the job with the **Stop** button ( ).
32. Close both models and exit the Model Editor.

Exercise 14. Importing A Completed Job

This exercise details how to import a completed plot file into the calculation server using Job Status.

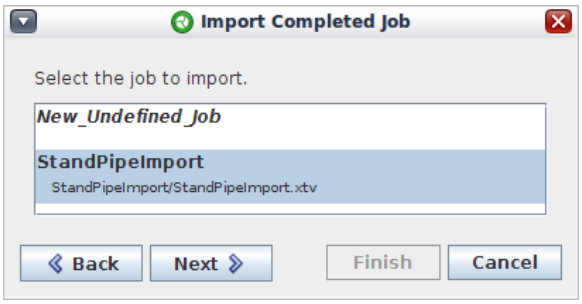
1. Using Windows Explorer, copy the directory "**runs/StandPipeImport**", included with this exercise, to the **runs/TRACE** folder created in Exercise 2.
2. Open Job Status. On Windows, select from the Start menu: **All Programs → SNAP → Job Status**.
3. Expand **Local**, the **runs** folder, and select the **TRACE** folder.
4. Right-click the **TRACE** folder and select **Import Completed Job** from the pop-up menu.

   *The **Import Completed Job** dialog appears, as displayed below. This dialog is used to import the job in a multi-step, guided process. The first step in the import is used to specify the type of job imported by the dialog.*



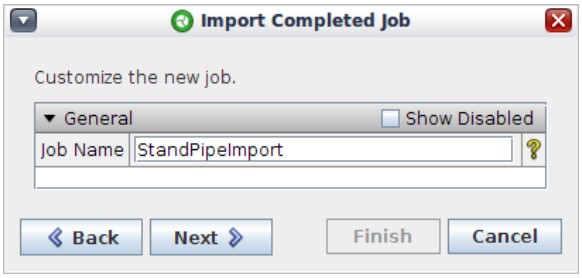
5. Select **TRACE** in the list and press the **Next** button.

   *The second step in the import dialog appears, as shown below. This step is used to specify which job is imported. The TRACE job importer recognizes a potential job named **StandPipeImport**, as the directory created earlier, along with the copied XTV file, follow the basic organization of a completed TRACE job.*



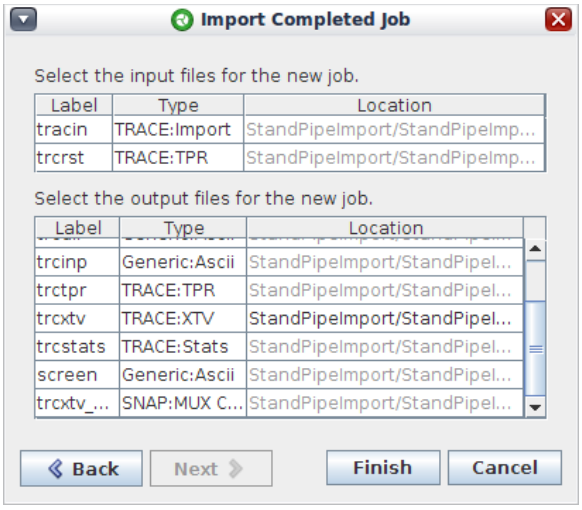
6. Select **StandPipeImport** in the list and press the **Next** button.

   The third step in the import dialog appears, as shown below. This step is used to specify any plug-in specific properties required by the job importer. The TRACE importer only allows specifying the name of the imported job. This exercise will use the default name ***StandPipeImport****.*



7. Press the **Next** button.

   The fourth and final step in the import dialog appears, as shown below. This step allows the user to specify the files recognized as part of the imported job. Note the grayed-out locations. These rows indicate the expected names of files that could be included with the task, but were not found by the importer. This exercise will use the default values.

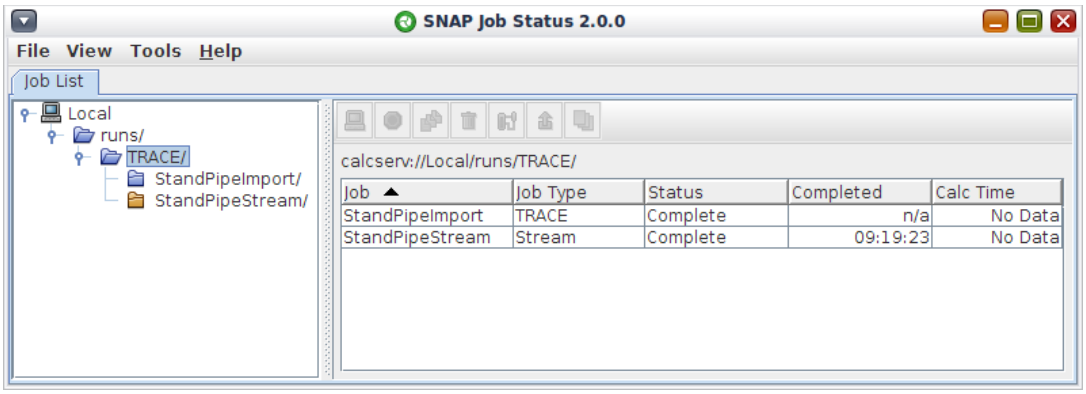


8. Press the **Finish** button.

   Notice that StandPipeImport has been added to the job list as a TRACE job.

With the job imported, it can be used as a data source for an animation, which is demonstrated in the remaining steps.

9. Open the stand pipe animation model used a previous exercise: **SNAP_Exercises\SandPipe_Anim_interactive.med**



10. In the Navigator, expand the **Data Source** node and select the **Master** data source.
11. Set the **Source Run URL** to the newly imported job.

    The steps are the same as those in previous exercises. The imported job is in the ***runs/TRACE*** *folder.*
12. Press the **Connect** button to connect to the job.

    If another animation model is still connected, then simply press the ***Connect*** *button twice: once to disconnect the other model and again to connect the current model.*
13. Animate the model via the runtime controls.
14. Close Job Status.
15. Close the animation model.

Exercise 15. Using AptPlot and the ACS Plug-in

This exercise is designed to familiarize the analyst with the basic functionality of AptPlot and the ACS plug-in. A TRACE plot-file will serve as a source of data for several graphs on a plot. In addition, some introductory graph and data set formatting will be performed.

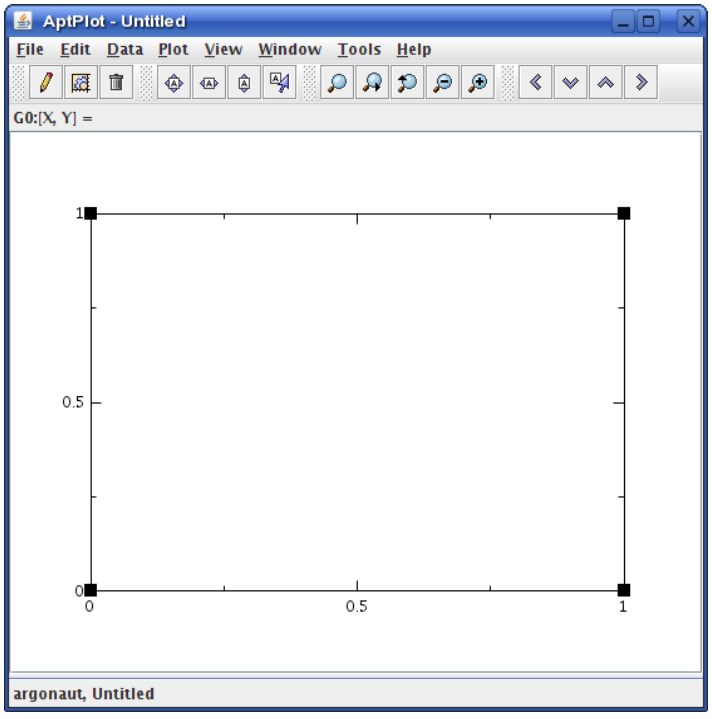
The following steps will guide the user through the exercise.

1. Open AptPlot. From the Windows Start menu, select **All Programs → Plotting Tools → AptPlot**.

   This will open AptPlot with a new plot, as shown below. From here, data will be added to the plot from a TRACE plot file.

   First, a quick rundown of the window. The white area is the plot. Anything appearing within this "canvas" is part of the plot and will appear in print-outs and exported images. The rectangle within the plot is a graph. The data itself is represented by Data Sets: linked columns of data contained within a graph. From the bottom up: data sets form lines on the graph, graphs create a window into their data, and plots are the root container for everything.

   The new plot in the window contains a single graph without any data sets. The first several steps of this exercise will add a data set to the graph by reading values from a TRACE plot file.



2. From the AptPlot main menu, select **File → Read → TRACE data...**.

   A file selection window is displayed.

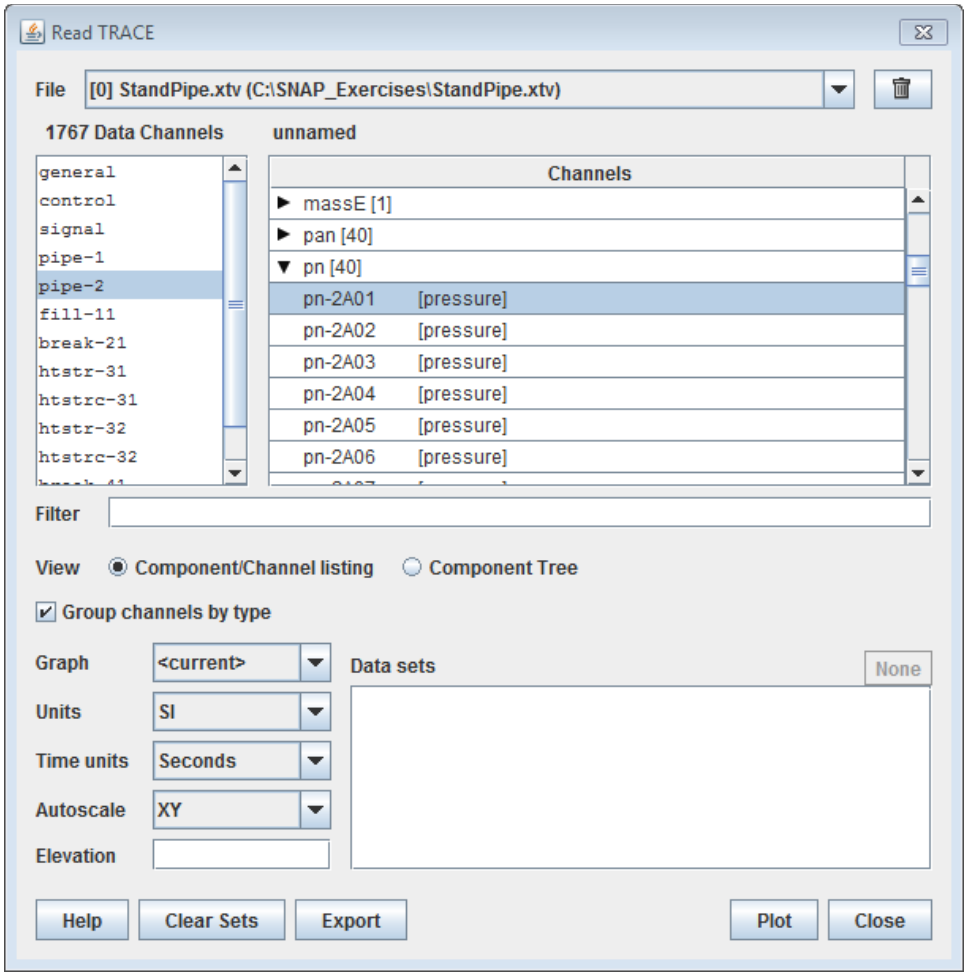
3. In the file dialog, select **Next Available** for the **File Index** and **XTV** for the **File Type**.

   The file type tells AptPlot how to attempt to read the selected file. Most TRACE plot files will use XTV, even if the file has been demultiplexed. A demultiplexed file is a version of the plot file reorganized for more efficient plotting. Demultiplexing is explained in detail in the AptPlot documentation.

   As for the file index: each code type supported by the ACS plug-in allows opening up to 30 plot files at any given time. ***File index*** *specifies which of those slots is assigned to the selected file. With* ***Next****, the file is assigned to the next unused plot index.*

4. In the file dialog, navigate to the **SNAP_Exercises** folder provided with this exercise.
5. Select the file **StandPipe.xtv**.
6. Press the **Open** button.

   A channel selection window will appear, similar to the image below, which will be used to read the channel data into the plot.



7. Select **pipe-2** in the list of categories of the left.

   The list of channels will change to those associated with the pipe-2 component.

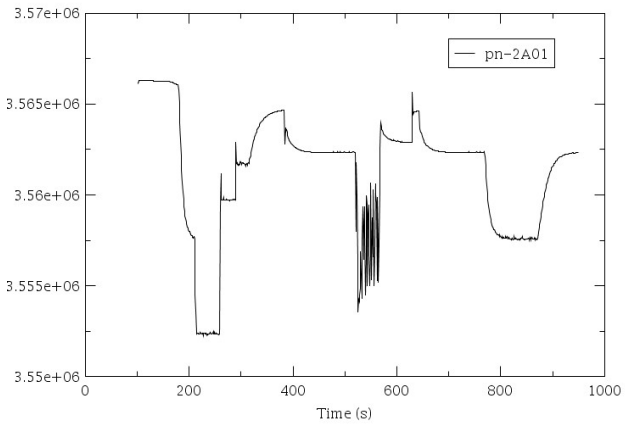
8. Type "**pn-***" (sans quotes) into the **Filter** field and press the Enter key.

   *The list of channels will be filtered to only those channels whose names begin with "**pn-**". The ***** character will match any string of letters (including none). "**pn-2*1***" would match any channel name that starts with "**pn-2**" and ends with "**1**".*

9. Select **pn-2A01** from the list of data channels and press the **Plot** button.

   The channel data will be plotted in the graph, as shown below. Notice that the graph's displayed axis values change to fit the data. Each graph has a set of World Coordinates that define which the range of values it displays. To illustrate, zooming out on a graph widens the world coordinate window.

   Unfortunately, this graph's numeric axis labels reach to the edge of the plot with the default formatting. While it is not currently visible, a vertical unit label "Pressure (Pa)" has been added to the Y axis: the long axis labels have pushed it off the plot. The next several steps will re-size the graph until this label is visible.



10. Close the channel dialog.
11. Double-click and hold on the upper-left corner of the graph, where the small black square is placed.

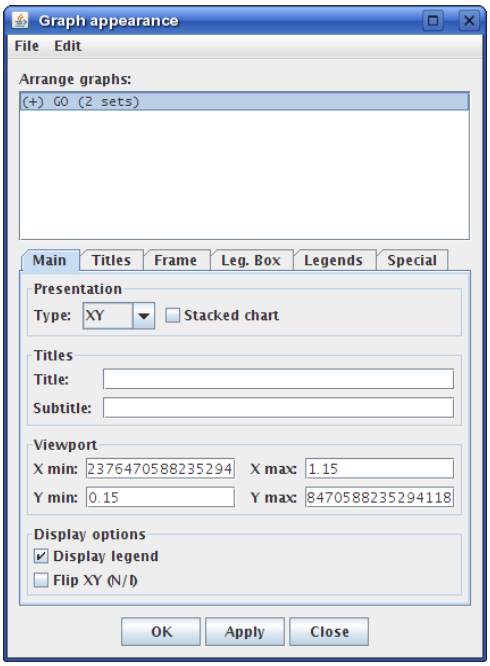
    This will initiate a graph re-size operation, indicated by the red "rubber-bands" displayed on the plot. By moving the mouse around, the rubber-bands indicate the new bounds of the plot. Clicking once will place the corner in its new location and re-size the graph.

12. Make the graph a little smaller by placing the corner slightly to the right of its original location.
13. Repeat the previous two steps until the "**Pressure (Pa)**" label is completely visible.

The current graph displays large values in a relatively narrow range. Further into the exercise, a channel will be plotted with very small values. If the new data is plotted alongside the current data, the graph will appear as two straight lines with a wide margin between them. To avoid this, the new data will be placed on a second graph.

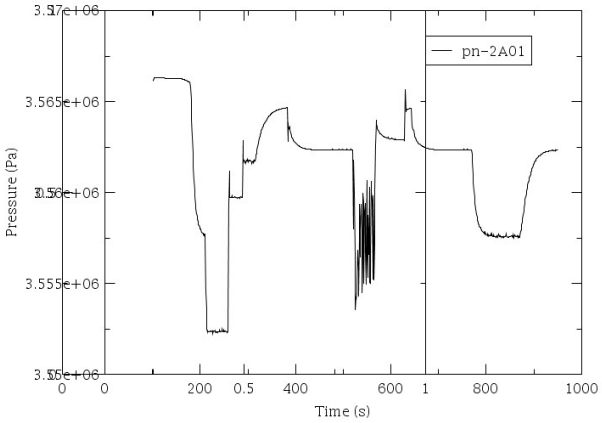
14. From the main-menu, select **Plot → Graph appearance...**.

    *The **Graph Appearance** window is displayed. The top of the Graph Appearance dialog is a graph list. Selecting a graph in the list determines which graph's appearance is being edited below. This type of graph list is used frequently throughout AptPlot, and each one provides identical functionality. This particular dialog was opened specifically to create a new graph from the graph list. In practice, any AptPlot dialog with a graph list could have been used.*



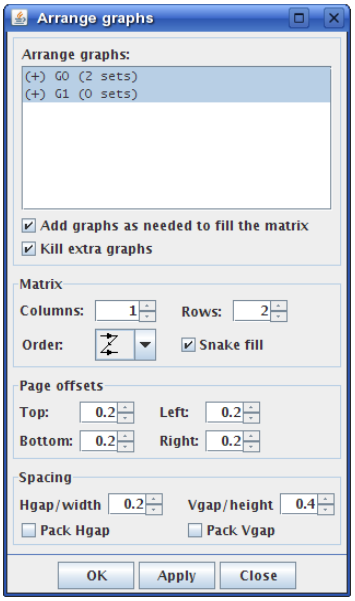
15. Right-click anywhere within the graph-list and select **Create new** from the resulting pop-up menu.

    A new graph is created. The plot should look similar to the one below. The new graph overlaps a significant portion of the original graph. This could be fixed by moving the graph corners around, but this would ignore several options AptPlot makes available for quickly laying out plots.



16. From the main-menu, select **Edit → Arrange graphs...**.

    *The **Arrange Graphs** window will be shown. With this dialog, graphs can be arranged in a grid of any dimensions, automatically assigning each the same size, vertical and horizontal spacing, and overall margin from the edges of the plot. Additionally, the dialog can create and remove graphs as needed to fill the grid.*



17. Select both graphs in the list. To select both graphs, select one, press and hold the Shift key, then select the other.

    Both graphs must be selected to indicate they are the graphs being arranged. If only one graph is selected, AptPlot would create a third graph, as the parameters entered in the next step tell AptPlot to create a graph grid with two rows in one column.

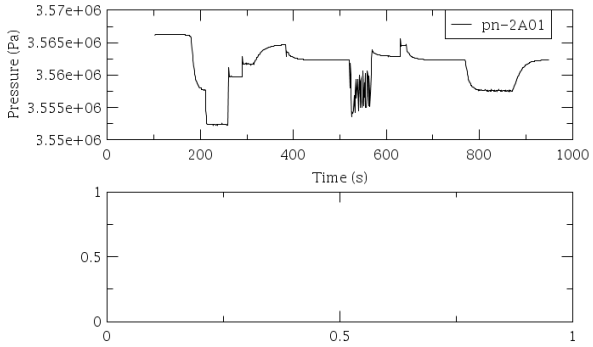
18. Set the following properties in the Arrange Graph window to the indicated values:

Matrix	Columns:	**1**
	Rows:	**2**
Page offsets	Top:	**0.2**
	Bottom:	**0.2**
	Left:	**0.2**
	Right:	**0.2**
Spacing	Hgap/width:	**0.2**
	Vgap/height:	**0.4**

The offsets and spacing above are specified in View Coordinates. A quick explanation: AptPlot creates a coordinate space for graphs that is based on ratios. The shortest side of the plot is 1 viewpoint long; the longer side's length is a ratio to the shorter side. This coordinate space is used to retain relative dimensions regardless of how the plot dimensions are changed. For example, if a user doubles the width of a plot, a square graph remains a square instead of transforming into a rectangle. If both dimensions of that same plot are doubled, the graph expands to correctly fill the space, retaining its original relative dimensions within the plot. The viewpoint coordinate system is the backbone of AptPlot's ability to create plots of any resolution that still appear identical to what's displayed on screen.

19. Press the **OK** button.

*The **Arrange Graphs** window disappears, and the graphs are rearranged. The next several steps will add data to the second graph.*



20. From the main menu, select **Edit → Edit plug-in data → TRACE data...**.

*The channel dialog will reappear. Once a plot file has been opened, its data can always be accessed from the **Edit** menu (unless another file is opened in its file index).*

21. Select **rovn-2A01** in the channel list.

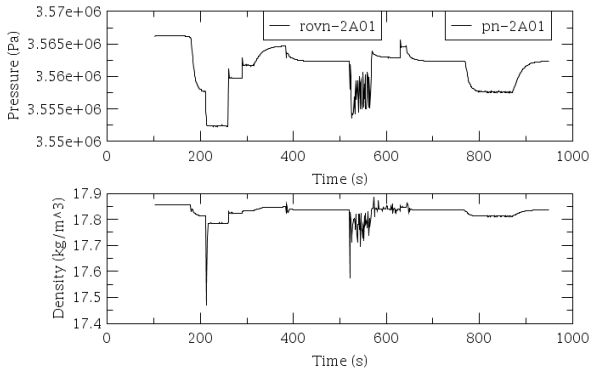
Remember that a channel filter can prune the list.

22. Select "**(+) G1**" from the **Graph** drop-down at the bottom of the window.

    This selects the graph into which the data will be read.

23. Press **Plot** to display the channel data, then close the dialog.

    At this point, the plot should have two graphs with data, axis labels, and legends. Notice that the second graph's legend is placed on the first graph. The next several steps will move the legend to a better location.



24. Click once anywhere on the second graph to select it.

    Many operations in AptPlot depend on the graph selection to determine which graph is affected. Moving the legend in the next step is one of them.

25. Press **Ctrl+L**.

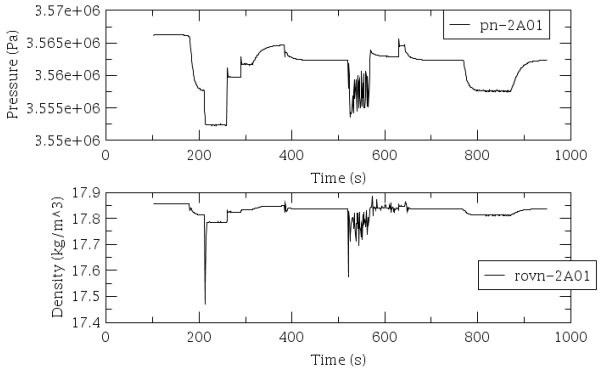
    AptPlot will enter "Move Legend" mode in the plot. This is visually indicated by a change in the mouse cursor.

26. Click and hold on the legend containing the entry **rovn-2A01**.

    This will begin an operation to move the legend, indicated by a shaded preview of the new location. Moving the mouse will move the preview indicator. Clicking once will set the legend in its new location. The move can be canceled by pressing the escape key or right clicking.

27. Reposition the preview indicator to a location on the right-side of the second graph, then click once.

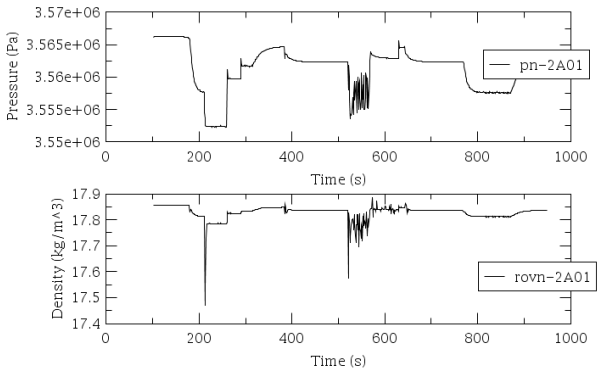
    The legend will be placed in the new location. The plot should look similar to the image below.



28. Press the Escape key or right-click.

    While this exercise moved the legend with the mouse, its coordinates can also be explicitly specified by editing the graph's appearance.

29. Repeat the last five steps to reposition the first graph's legend. When complete, the plot should resemble the following:

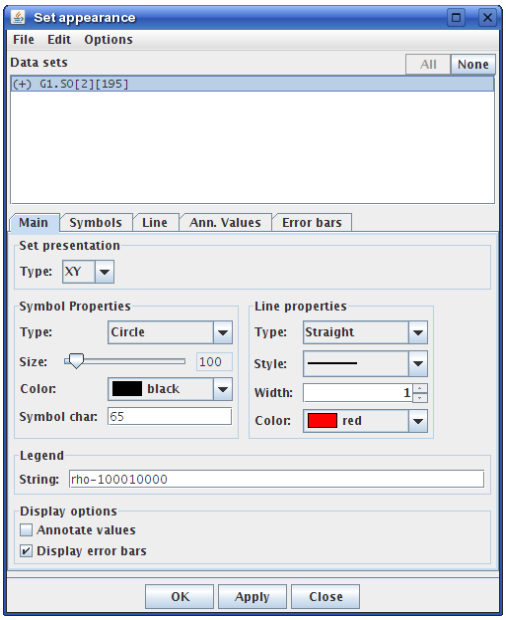


    The final steps of this exercise will modify the appearance of the data set in the second graph.

30. Double-click anywhere within the second graph.

    *The **Set Appearance** window shown below is displayed. Double-clicking within a graph opens the **Set Appearance** dialog for that graph's data sets. This dialog could have also been displayed by selecting "Plot → Set appearance..." from the main menu.*

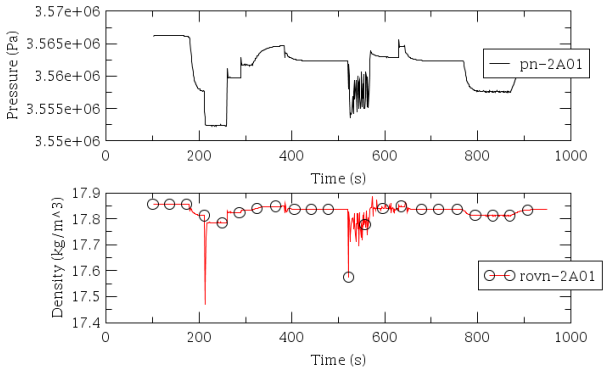
31. Make sure the data set is selected in the set-list the top of the dialog.



Like graph lists, set lists are used throughout AptPlot to select data sets. Also like graph lists, right-clicking on a set list will open a pop-up menu of set-related operations. Unlike how a graph list displays all graphs in the plot, the contents of a set list will always be the sets of the currently selected graph or those of the graph selected in a nearby list.

32. In the **Main** tab, set **Line properties Color** to **Red**.
33. In the **Main** tab, set the **Symbol Properties Type** to **Circle**.
34. In the **Symbols** tab, set the **Extra** property **Symbol Skip** to **35**.
35. Press the **OK** button.

The data set will be adjusted with the new formatting. Notice how the set is now painted as a red line with a circle around every 35th point. The plot should look similar to the image below.



36. From the main menu, select **File → Save**.

    A file dialog is displayed requesting where to save the plot.

37. Navigate to the **SNAP_Exercises** folder in the directory provided with this exercise, and save the file as "**ex15_finalplot.apf**".

38. Leave AptPlot open for the next exercise.

Exercise 16. 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.

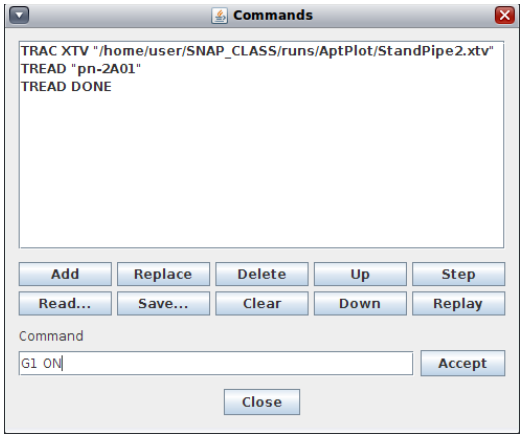
Before proceeding, a note about entering commands: each command in this exercise must be entered exactly as listed. If a command is entered incorrectly, one of two things will happen. The most likely occurrence is AptPlot will warn the user about syntax error or incorrect argument. In this case, close the error console and reenter 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 hit 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 indexes 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:

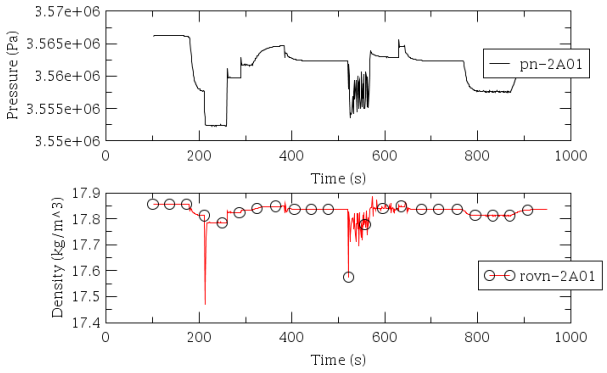
```
FOCUS G1
S0 LINE COLOR 2
S0 SYMBOL 1
G1.S0 SYMBOL SKIP 60
```

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 uses 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 17. AptPlot in Job Streams

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

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

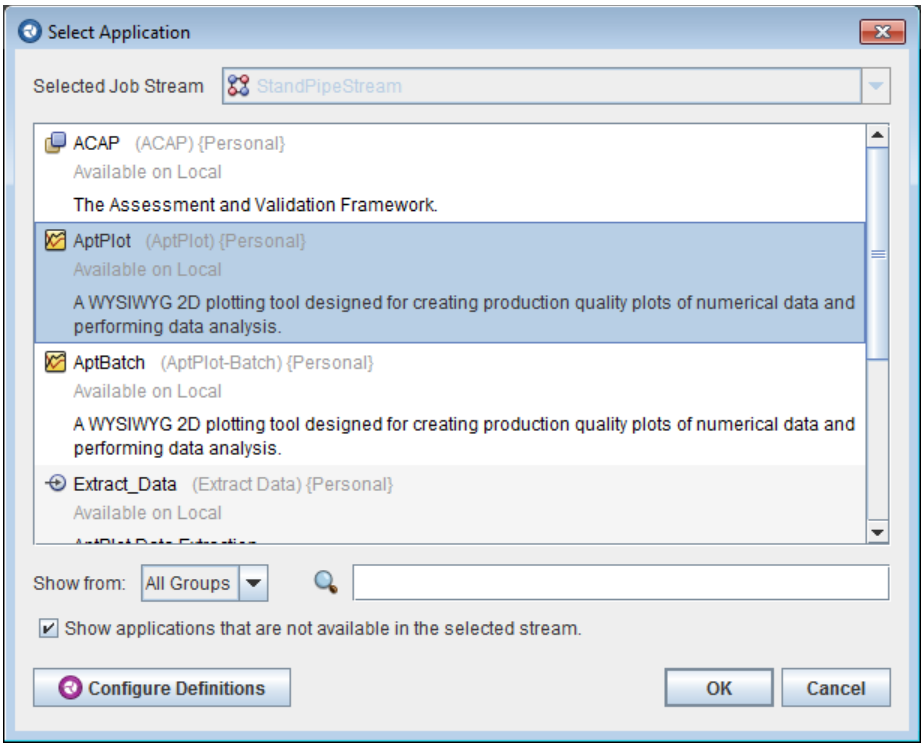
1. Open **SNAP_Exercises/StandPipe7.med**, included with these exercises, and make the following modifications:
    1. Select **StandPipeStream** in the Navigator and make sure its **Platform** is set to "**Local**" and its **Root Folder** to "**runs**" (or the appropriate root folder used in place of 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 sets an extremely long calculation time, so that the interactive commands can be demonstrated without the model running to completion.
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.

    *Notice that 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 created in an earlier exercise.

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

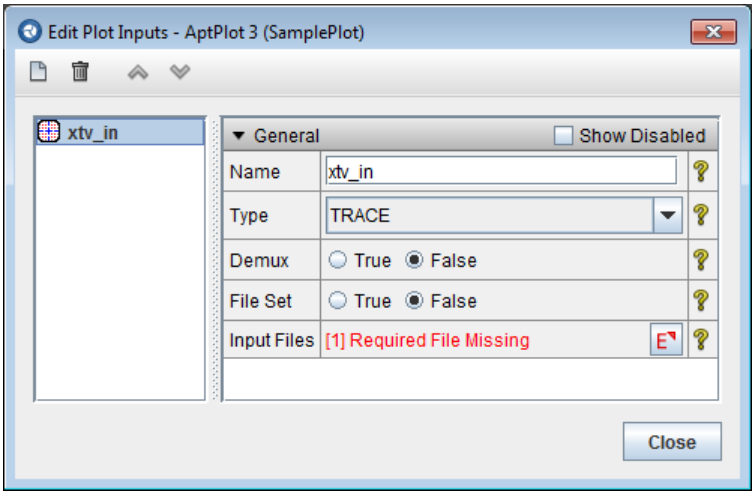
    *A new External File has been created in the stream, representing the parameter file. In addition, this new file definition has been connected to the optional **param** input on the **SamplePlot** step.*

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**

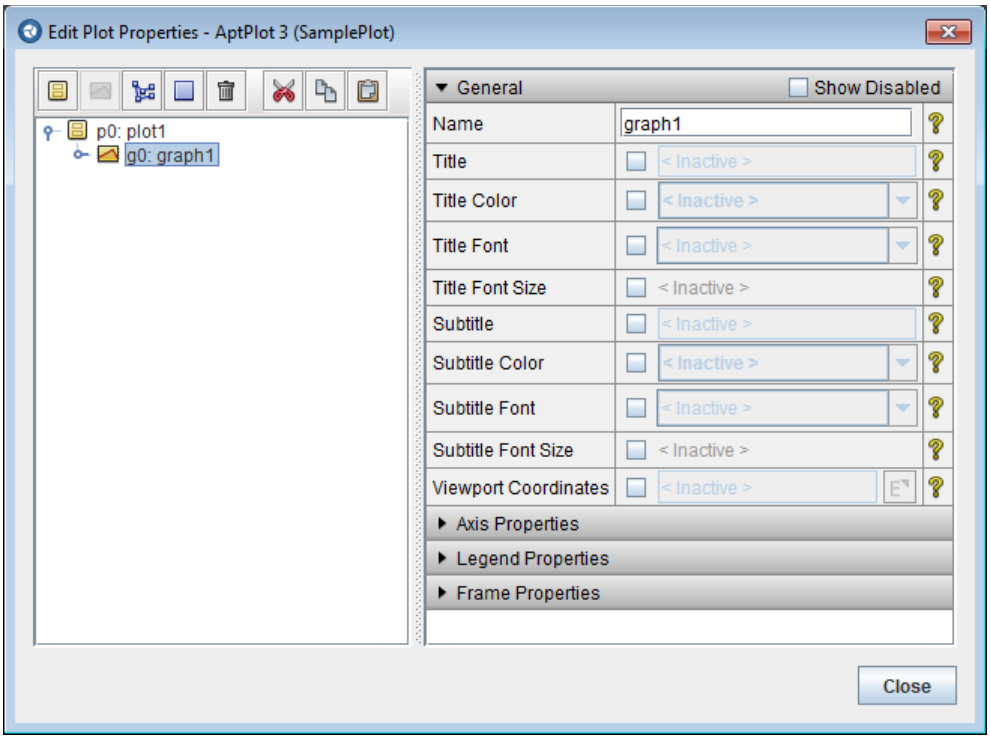
    *Notice that a new input has appeared on the AptPlot step in the view, labeled **xtv_in**. Each input defined in the **Plot Inputs** editor defines another input for the step itself.*

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 that 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. Open the **StandPipeStream** folder, then the **SamplePlot** folder, and finally select the **p0_plot.apf** file.

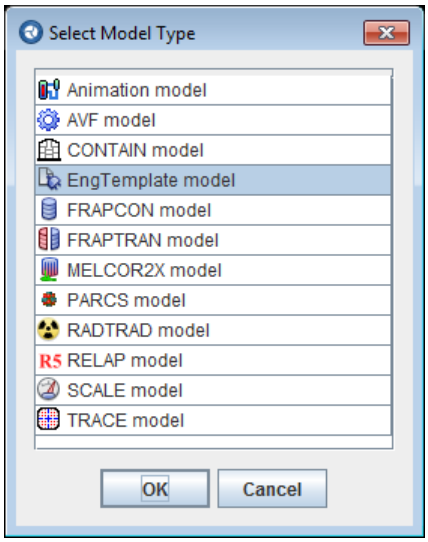
53. Press the **Open** button.

    The plot will be recreated in AptPlot.

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

Exercise 18. 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 "**Reference Models**" category node in the Navigator.
6. Right-click on the "**Reference Models**" node and select the **New** option from the right-click pop-up menu.

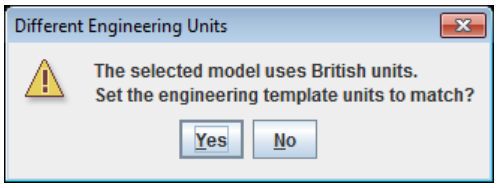
   Engineering templates reference models through Reference Model components. These components reference models by selecting the Model Editor Document (MED) files in which they are contained. The models selected as model references are referred to as "underlying models".
7. Select "**TRACE model**" from 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.
8. Press the **E** button in the **Input Model** property editor.
9. 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.

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



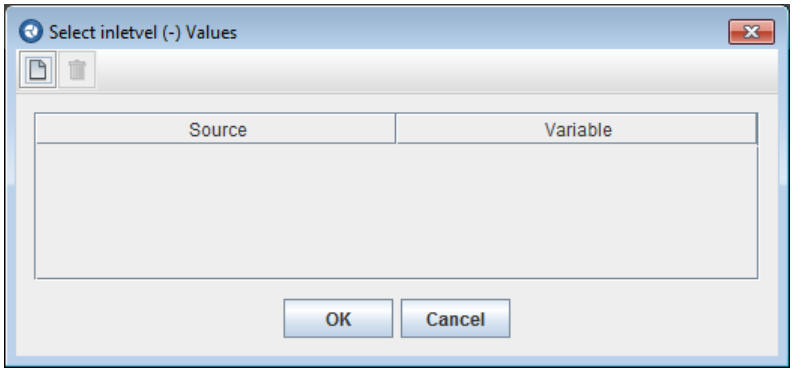
The Engineering Template model can use either British or SI display units. This setting does not affect the units selected by underlying models. All values displayed and/or edited in an engineering template are automatically converted to the appropriate units when they are passed to an underlying model.

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

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

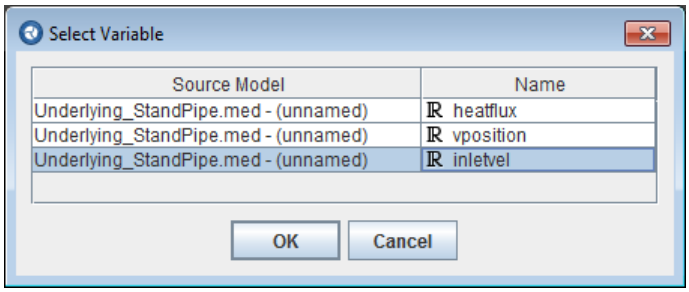
    This will create a new global real using the "No Unit" unit type and select it in the Navigator.
14. Set the **Name** property to "inletvel".
15. Press the **E** button in the **Reference Variables** property editor.

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



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

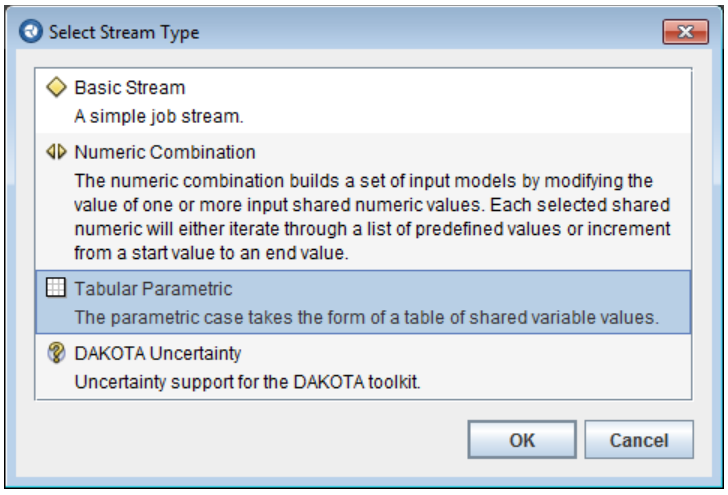
17. Select “inletvel” from the list.



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

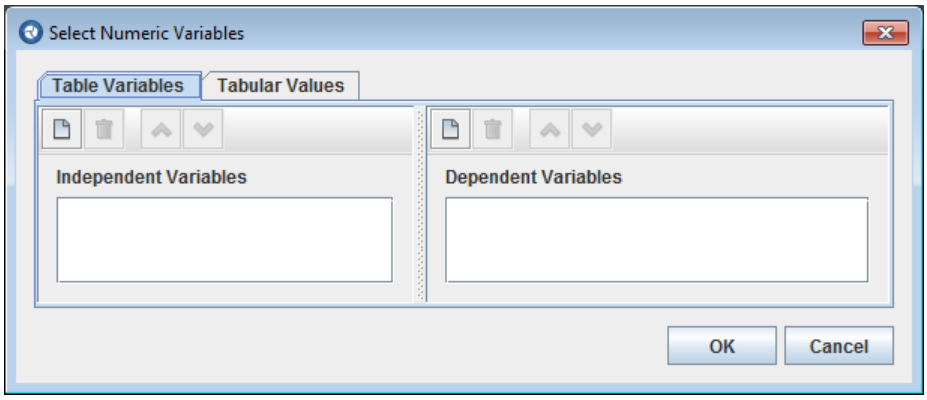
28. Create a new Job Stream by right-clicking the “Job Streams” node and selecting **New** from the pop-up menu.



29. Select the **Tabular Parametric** stream type and press **OK**.

30. Set the **Name** of the new job stream to “Template_Stream”.
31. Press the **E** button in the **Parametric Properties** property editor.

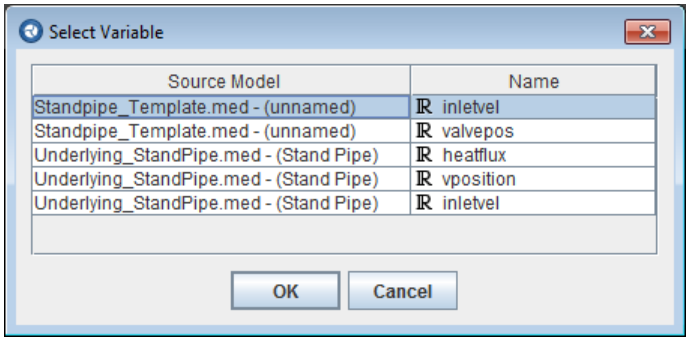
    This will open the Select Numeric Variables dialog used to configure the tabular parametric stream type.



    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.

32. 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.

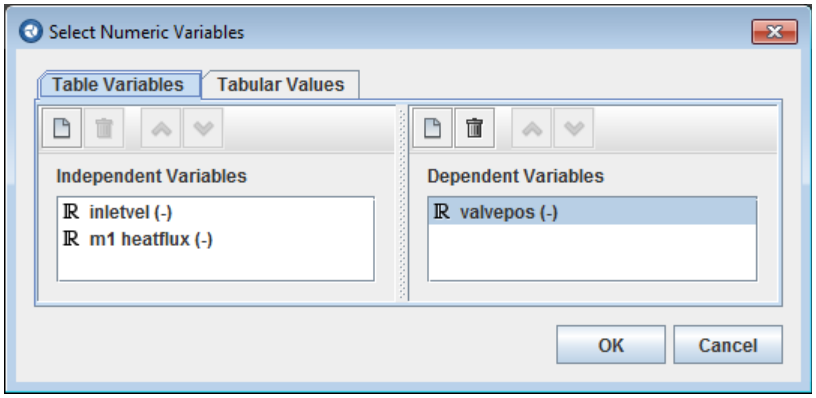
33. Select the global variable “**inletvel**” from the list of variables.
34. Press **OK** to select the variable.

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

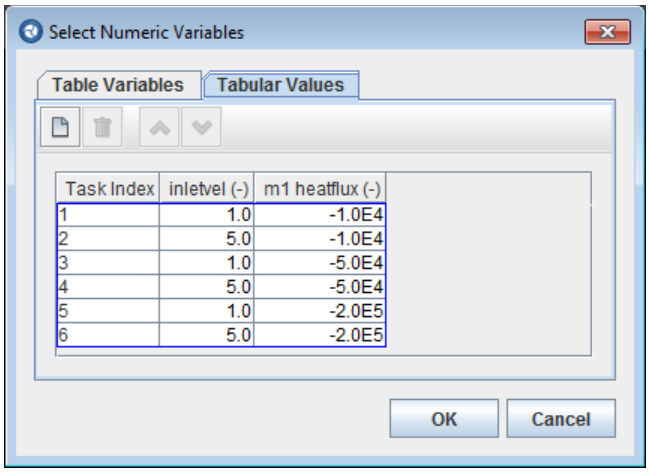
35. Press the **New** button again to add a second independent variable.
36. Select "heatflux" from the list of variables.
37. Press **OK** to select the variable.

    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.

38. Press the Dependent Variables **New** button to add a dependent variable.
39. Select "valvepos" from the list of variables.
40. Press **OK** to select the variable.



41. Select the **Tabular Values** tab.



The table data that makes up the tabular parametric is located in the Tabular Values tab. The table also 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.

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

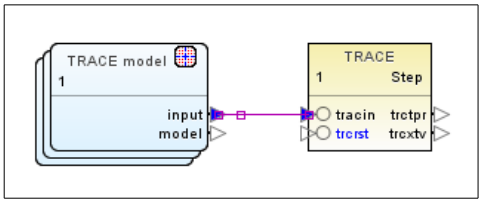
43. Enter the following inletvel and heatflux values.

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

44. Press **OK** to complete the tabular parametric properties.
45. Expand the "**Job Streams → Template_Stream → Model Nodes**" node in the Navigator.
46. Select the "**TRACE model 1**" node in the Navigator.
47. Set the **Parametric** property to "True".
48. Drag "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.
49. Create new job step by right-clicking the "**Stream Steps**" node and selecting **New** from the pop-up menu.
50. 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.
51. Set the **Name** property to "StandPipe".
52. Set the **Interactive** property to "On".
53. Drag the "StandPipe" job step node from the Navigator into the "Default View".



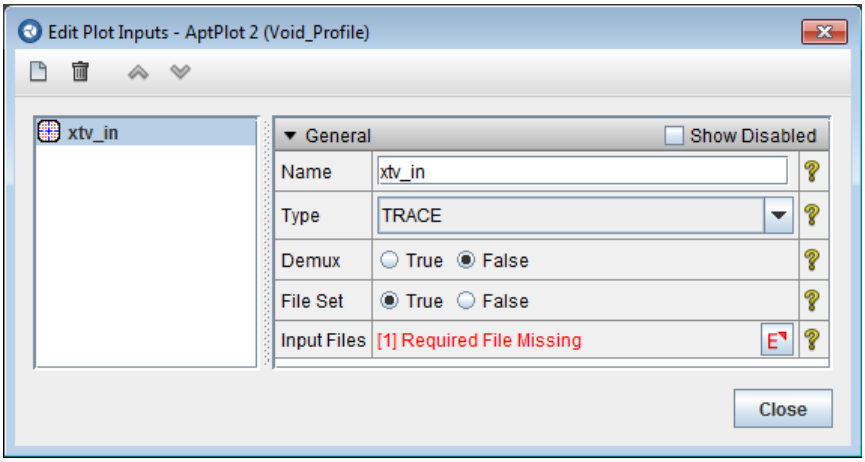
54. In the "Default View", use the Connect Tool to connect the "input" connection point of the TRACE model node to the "tracin" connection point of the "StandPipe" step.

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.

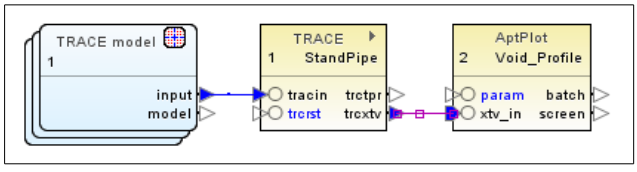
55. Create new job step by right-clicking the “Stream Steps” node and selecting **New** from the pop-up menu.
56. Select **AptPlot** from the list of applications and press the **OK** button.

    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.
57. Set the **Name** property to “Void_Profile”.
58. Press the **E** button in the **Plot Inputs** property editor.

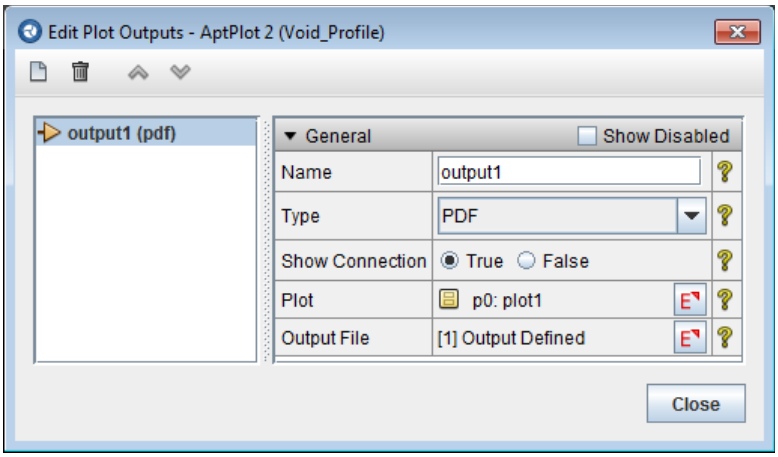
    This will open the plot inputs dialog shown below.



59. Press the **New** button to create a new plot file input.
60. Set the **Name** of the new input to “xtv_in”.
61. Set the **Type** to “TRACE”.
62. Set the **File Set** property to “True”.
63. Press the **Close** button to close the dialog.
64. Drag the “Void_Profile” job step node from the Navigator into the “Default View”.
65. 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.

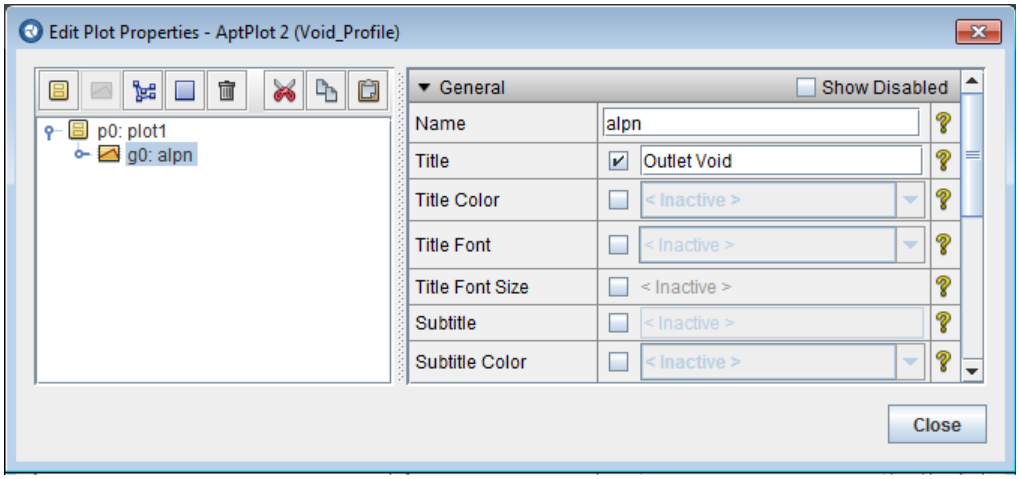


66. Select the “Void_Profile” step in the Navigator or the view.
67. Press the **E** button in the **Plot Outputs** property editor.



68. Press the **New** button to create a new plot output.
69. Select the "p0: plot1" plot definition and press **OK**.
70. Set the output **Name** to "void_profile".
71. Set the output **Type** to "PDF".
72. Press **Close** to close the plot outputs window.
73. 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.

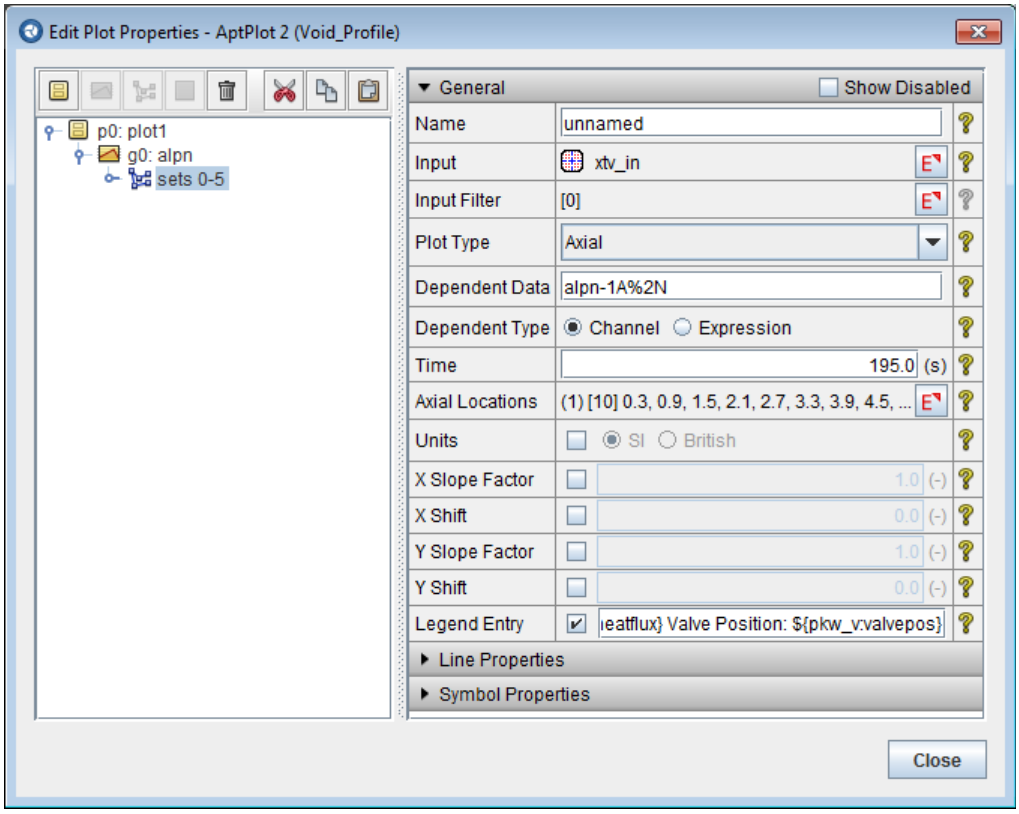


    The tool-bar over the can be used to add or remove plots, graphs, data sets, and annotations. Graphs can only be added when the parent plot is selected; data sets and annotations can only be added when the parent graph is selected. Additionally, cut, copy,

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.

74. Select the "g0: graph1" graph node.
75. Set the following g0 properties:
    - **Name**: alpn
    - **Title**: Outlet Void
    - **X Axis Label**: Elevation (ft)
    - **Y Axis Label**: Void
    - **Y Axis Scaling**: Explicit
    - **Y Axis Bounds**: [-0.1, 1.1]
    - **Legend Coordinates**: [0.3, 0.82]
    - **Legend Text Font Size**: 50
76. 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.



77. 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.

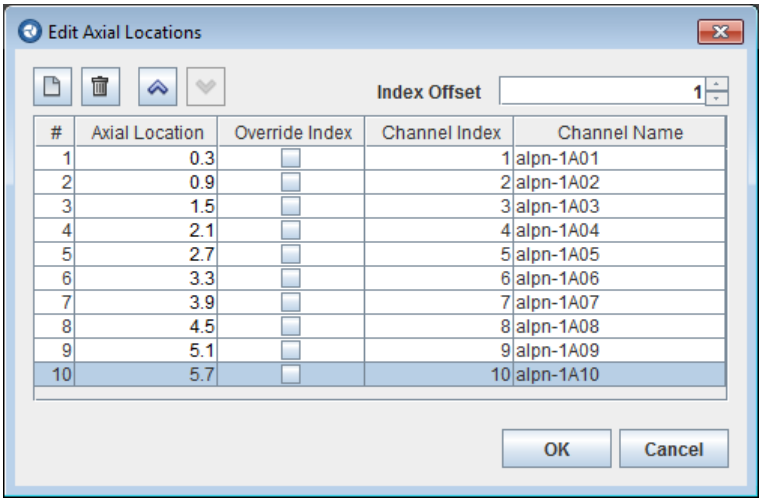
78. Select “xtv_in” and press **OK**.
79. Set the following properties:
    - **Plot Type**: Axial
    - **Dependent Data**: alpn-1A%2N
    - **Time**: 195.0 (s)
80. Set the **Legend Entry** property to the following single line:

    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.

    This legend entry uses three “Parametric Keyword” tokens (${pkw_v: <keyword>}) to to make it clear which velocity, heat flux, and valve position values, are represented by which void profile line.
81. Press the **E** button in the **Axial Locations** property editor.

    This will open the axial locations dialog, shown below.



82. Add 10 rows by pressing the **New** button 10 times.
83. Set the **Offset Index** to 1.

84. Set the following **Axial Location** values:

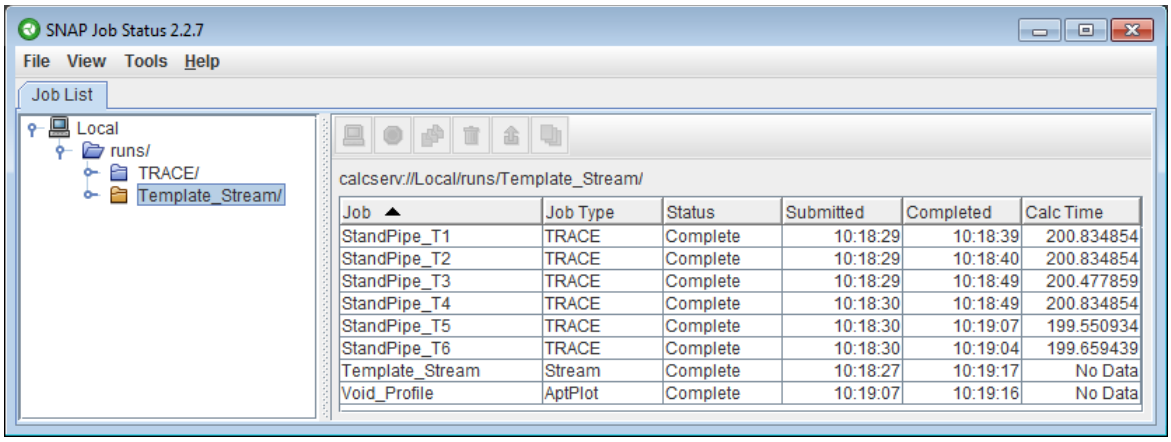
#	Axial Location
1	0.3
2	0.9
3	1.5
4	2.1
5	2.7
6	3.3
7	3.9
8	4.5
9	5.1
10	5.7

85. Press **OK** to close the axial locations dialog.
86. Expand the **sets 0-5: alpn** node.

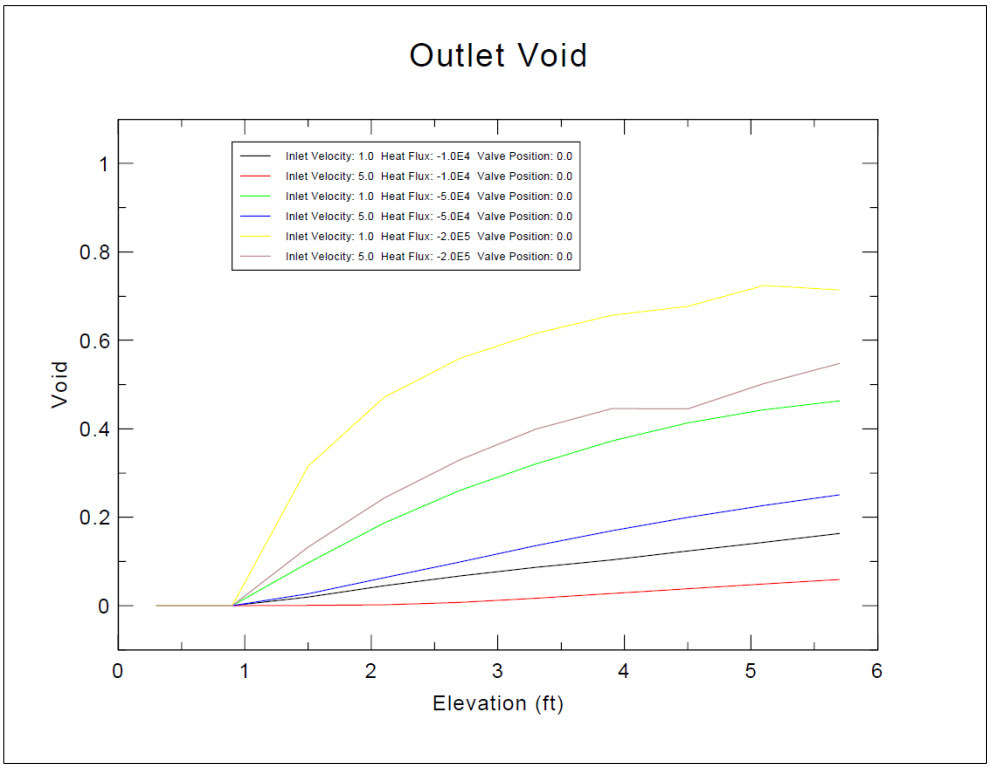
    *Note that the **sets 0-5: alpn** node has 5 children (s0 through s5).*
87. Select the first child node: **s0**

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

    This will submit the stream and open Job Status automatically.
94. Wait for the stream to complete.



95. Select the **Void_Profile** row in the table.
96. Press the **View Files** ( ) button to open the files pop-up menu.
97. 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.

98. Close the PDF viewer.

Exercise 19. 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 that opens 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:

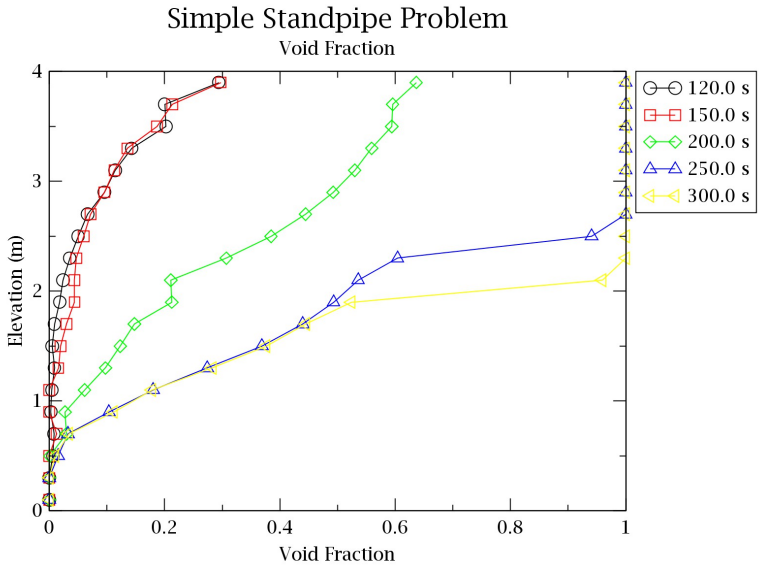
```
REDRAW
HARDCOPY DEVICE "PDF"
PRINT TO "<HOME>\VoidProfile.pdf"
PRINT
```

*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.