mFUSE: Function Sequencer for MATLAB

Help Manual

LANL/UCSD Engineering Institute

LA-CC-10-033
LA-UR 10-01264

July 29, 2010

Contents

Part I
What is mFUSE?

mFUSE: Function Sequencer for MATLAB is a Java based graphical user interface for use with MATLAB. mFUSE facilitates the development of analytical processes by allowing users to quickly and intuitively connect MATLAB functions as steps in a sequence. mFUSE will help you easily develop and compare similar processes. The software also supports collaboration on projects through the use of session files (.ses) and packages. mFUSE features a modern graphical display, drag-and-drop support, advanced m-file parsing, numerous saving options, and a reconfigurable function library.

Using mFUSE to enhance your experience with MATLAB can simplify your thinking while increasing productivity. mFUSE encourages you to develop a set of modular functions to reuse for all your projects. Modular design entails standardizing the header format and data structures used in your functions. As you begin to develop a database of functions which can be reused in new processes, you will find yourself repeating code less frequently therefore saving time. Even users unfamiliar with MATLAB can make use of your modular MATLAB functions through the informative interface mFUSE provides.

1 Features

mFUSE 0.2.00 (Beta) includes the following features:

- Works with any MATLAB function m-files

- Reconfigurable function library

- Displays useful function information from standardized headers

- Detects function headers that do not meet mFUSE standards

- Repairs cmommon code structure errors in functions

- Supports functions with multiple call methods

- Provides helpful hints and messages throughout interface

- Allows Quick Steps to be used in place of entire m-files

- Status indicators make it easy to see if a step has been configured

- Help messages provide detailed information about where errors occur in sequences

- Results can be viewed, plotted, and saved from within mFUSE

- Extensive m-file saving options

- Automatically generate m-files with standardized headers and comments

- Save sequences as a package to move them to another system

- Combine modular functions in a single file

- New function wizard makes creating functions with standardized headers easy

- Save and share sessions with other users

- Automatically recovers previous session if program closes unexpectedly

- Backwards compatibility with previous mFUSE saved sessions

- Fully supports debugging of sequence functions in MATLAB

5

2 Version Numbers

mFUSE versions are identified by a version number of the form *X.Y.ZZ*. The current version is 0.2.00.

X denotes the primary version. Primary version 0 is a beta for testing purposes. Each successive primary version represents major improvements to the software.

Y denotes the feature set. Each primary version will start with feature set 0. A change to the feature set number represents added functionality in the software

ZZ denotes the update level. Each primary version will start with update level 00. When a bug is fixed or minor functionalities are changed, the update level will increase.

3 About This Manual

This manual is divided into 5 parts. Part I is called "What is mFUSE?" and provides background and other basic information about mFUSE. Part II, "Getting Started," tells you all the actions to take before using mFUSE on your system. Part III is called "Welcome to mFUSE" and provides a brief description of different areas of the mFUSE interface as well as basic steps to create and use your first sequence. Part IV, "Advanced Features," covers all the details of how to make full use of the interface. "Appendices" in part V provides an example sequence and troubleshooting information.

4 Author Information

mFUSE: Function Sequencer for MATLAB was developed by Dustin Harvey and Eric Flynn at the Engineering Institute, a collaboration between Los Alamos National Laboratory and the University of California at San Diego's Jacobs School of Engineering. For more information, visit `http://institute.lanl.gov/ei/software-and-data/`

5 Copyright and License Information

Part II

Getting Started

The following sections describe mFUSE's system requirements, the files needed for operation, and platform specific instructions to prepare your system and install mFUSE.

6   System Requirements

Hardware

- Intel or AMD Processor

- 512 MB RAM

- 2 MB Free Disk Space

Software

- Any operating system supported by MATLAB

- MATLAB version 7 (R14 from 2004) or higher

- Java Runtime Environment 5 or later

7   Included Files

The following files must be copied to your system *in the same directory* for mFUSE to function:

- *all2str.m* – converts MATLAB workspace variables to strings for viewing in mFUSE

- *copyright.txt* –contains mFUSE licensing information

- *installmFUSE.m* – adds necessary paths to MATLAB's search path and class path

- *mFUSE.jar* – contains Java application

- *mFUSE.m* – launches Java application from MATLAB

- *mFUSE_Help.pdf* – mFUSE documentation

- *mFUSElibrary.txt* – lists all directories in mFUSE function library

8   Windows Users

The following sections describe how to prepare MATLAB and your Windows system to run mFUSE and the installation procedure.

8.1   Prepare Your System

Before using mFUSE, MATLAB must be configured to use Java 5 (also called 1.5) or higher (Java 6/1.6 preferred).

1. Launch MATLAB and enter

   ```
   version -java
   ```

   in the command window.

2. If MATLAB returns any Java version above 1.5, skip to mFUSE installation. Otherwise, continue following steps.

3. Ensure you have Java 1.5 or higher installed.

   (a) Open a command prompt. Press *Windows Logo Key + R* then type

   ```
   cmd
   ```

   and press enter.

   (b) Type

   ```
   java -version
   ```

   and press enter.

   (c) If your Java version is at least 1.5, continue to step 4, otherwise download and install the latest version of Java from `http://www.java.com/getjava/`

4. Repeat steps 1 and 2. If MATLAB still reports a Java version below 1.5, you will need to create an environment variable to set MATLAB to use a different Java installation.

   (a) Click *Start* then *Control Panel.*
   (b) Open the *System* control panel.
   (c) Select the *Advanced* tab then click *Environment Variables.*
   (d) In the *User Variables* section, click *New.*
   (e) For variable name, use

   ```
   MATLAB_JAVA
   ```

   and for variable value use the directory where Java's "bin" folder is installed. For example,

   ```
   C:\Program Files\Java\jre1.6
   ```

   (f) Press *OK* three times. Your system is now ready to install mFUSE.

8.2   Installation

1. Launch MATLAB.

2. Change current directory to the location of the mFUSE files.

3. Run the *installmFUSE.m* script.

4. Restart MATLAB.

5. To launch mFUSE, enter

   `mFUSE`

   in the command window.

9 Macintosh Users

Note: mFUSE 0.2.00 is not fully compatible on Macintosh systems. Later releases of the software will include Mac support.

The following sections describe how to prepare MATLAB and your Mac OS system for and the installation procedure.

Note: Anywhere in the software where you are instructed to right-click, Mac users with a single-button mouse should control-click.

9.1 Prepare Your System

Before using mFUSE, MATLAB must be configured to use Java 5 (also called 1.5) or higher (Java 6/1.6 preferred).

1. Launch MATLAB and enter

   ```
   version -java
   ```

   in the command window.

2. If MATLAB returns any Java version above 1.5, skip to mFUSE installation. Otherwise, continue following steps.

3. Ensure you have Java 1.5 or higher installed. Open the Java Preferences application. If your Java version is at least 1.5, continue to step 4, otherwise download and install the latest version of Java through Software Update.

4. In the Java Preferences application, ensure that the more recent Java version installed on your system is the A version (first in the list). If it is not, reorder the list to move the newest Java versions to the top.

9.2 Installation

1. Launch MATLAB.

2. Change current directory to the location of the mFUSE files.

3. Run the *installmFUSE.m* script.

4. Restart MATLAB.

5. To launch mFUSE, enter

   ```
   mFUSE
   ```

   in the command window.

10 Linux Users

Linux support will be available in a later version.

Part III
Welcome to mFUSE

Before you begin, it's important to understand the structure of mFUSE. When you run mFUSE, the interface displays the state of the current session. A session includes a single sequence as well as all of the options you have selected for execution, saving, and plotting. Sessions can be saved and opened later through the use of .ses files.

A sequence is a series of steps that perform MATLAB commands. Sequences can be executed through mFUSE and saved as MATLAB functions or scripts. Each step is made of a function from an m-file included in the function library (unless it's a Quick Step, see section 19). Steps may have multiple call methods, but only one may be used at a time. They also contain information about the input and output parameters used in the selected call method.

The following sections will introduce you to the mFUSE interface display and the basic steps to create, execute, and save a sequence.

11   Display Areas

The mFUSE display contains six main sections. Three in the top half, and three in the bottom tabs. The sections are labeled in Figure 1 in the following order:

- *A* – Function Library (section 11.1)

- *B* – Sequence List (section 11.2)

- *C* – Selected Step (section 11.3)

- *D* – Function Documentation Tab (section 11.4)

- *E* – Sequence M-File Tab (section 11.5)

- *F* – User Inputs/Tagged Outputs Tab (section 11.6)
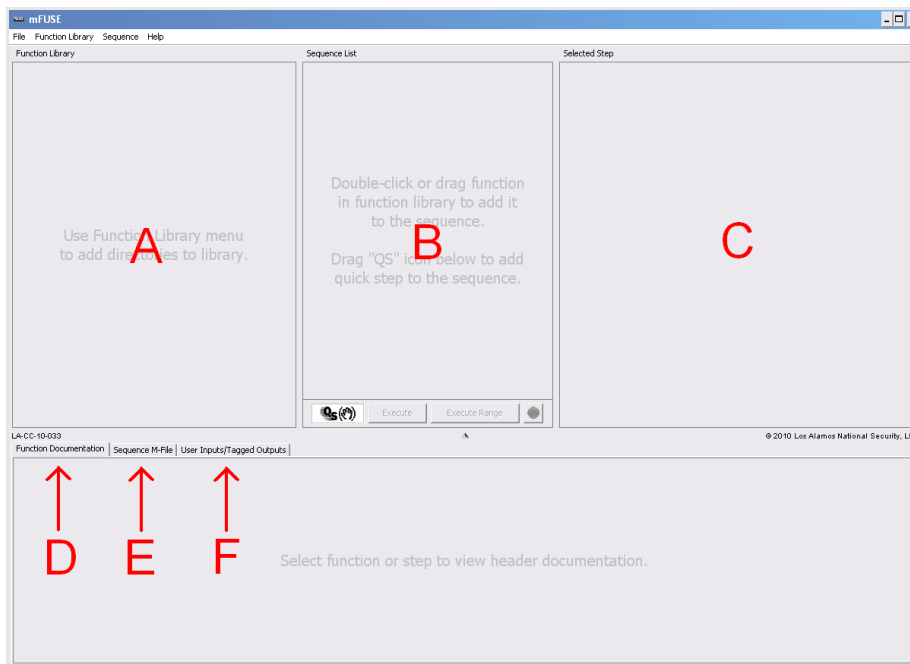
Figure 1: mFUSE interface at first launch.

11.1   Function Library

The function library displays all the functions available to add to a sequence. Each directory in your function library is displayed along with all subdirectories and m-files. Directories are indicated by a blue folder. M-Files are shown with a green paper icon. See section 12 for more information.

Any MATLAB functions can be used with the mFUSE interface; however, if the functions contain headers following the mFUSE header standard (see section 26) the interface can provide more options and useful information about the function. For example, Figure 2 shows an m-file called *sample.m*. Since the function header defines the function name to be *Sample Function*, the function library uses the more descriptive name to list the function.
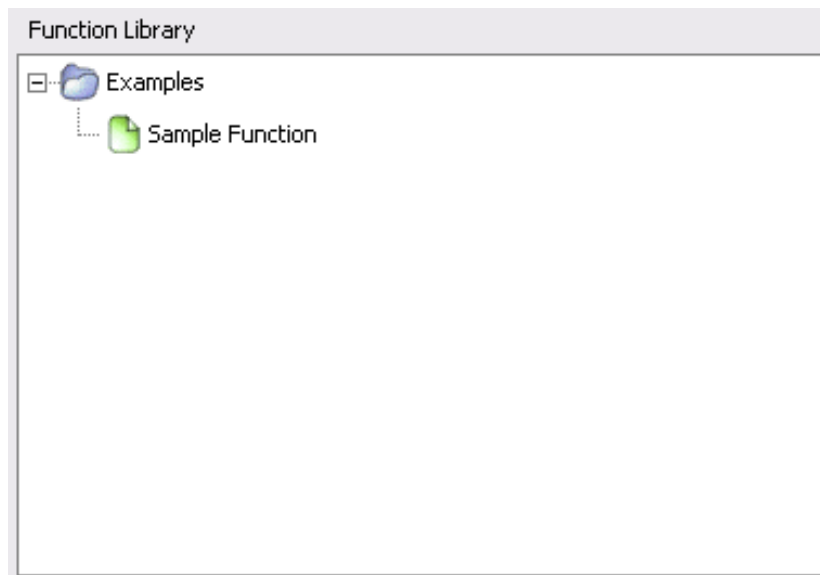
Figure 2: Function Library area.

11.2 Sequence List

The sequence list area shows all the steps in the sequence along and allows you to move, copy, and remove steps as well as execute the sequence and create Quick Steps (see section 19). Each step is listed with a colored circle, step number, and function name. The color of the circle indicates the lowest level status of the inputs in that step (see section 11.3).

11.3 Selected Step

The selected step area displays all the options for the step currently selected in the sequence list area (see section 11.2). The *Comments* box allows you to add comments that will be printed in the sequence m-file with the step. If the function used to create the step has multiple call methods, the desired function call can be selected in the *Call Methods* box.

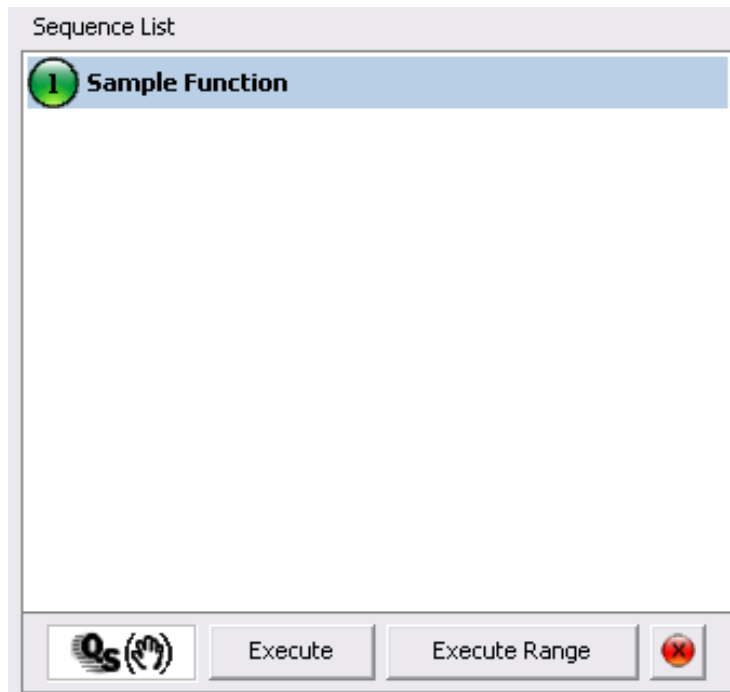All of the inputs used in the selected call method are listed in the *Inputs* table. each input is listed with a status icon, variable name, and selected value. The value can either be the default value, a user entered value, or the output of a previous step in the sequence. For example, Figure 3 shows a step with two inputs. Input 1 is set to the default value (in this case the number 1), and Input 2 has a user entered value of 3.

The color of the status icon for each input can be green, yellow, or red depending on the value of that input. A red icon indicates that no value is chosen or the input has an empty user value. You will not be able to execute or save the sequence as an m-file if it contains an input in red status. Yellow status indicates that the input is set to default value, but the function may or may not provide a default value for that input. If a default value is defined in the function header or if you assign default value to the input, the status will be green. Likewise, a non-empty user value or the output of a previous step will be shown with a green status.

The final box, *Outputs*, allows you to indicate which outputs from the selected step are important in your sequence by tagging them. Tagged outputs can be viewed, plotted, and saved from the *User Inputs/Tagged Outputs* tab (see section 11.6) and selected as function outputs for the entire sequence (see section 20.1).
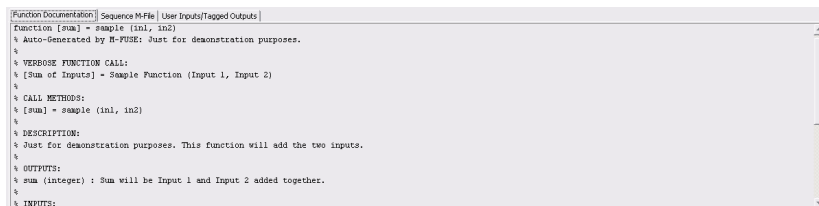
Figure 3: Selected Step area.

11.4 Function Documentation Tab

The *Function Documentation* tab shows the function header for the most recently selected function in the *Function Library* or *Sequence List*.
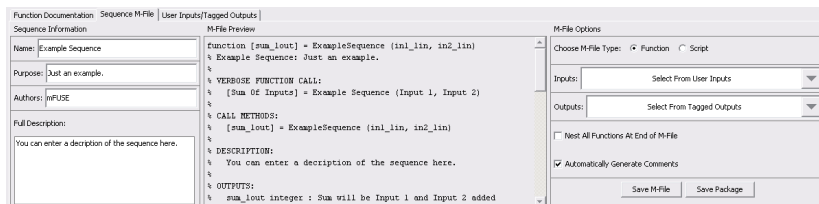
11.5 Sequence M-File Tab

The *Sequence M-File* tab provides all the saving options for sequence m-files as well as a preview of the m-file to be saved. See section 16 for more information.

11.6 User Inputs/Tagged Outputs Tab

The *User Inputs/Tagged Outputs* tab provides a convenient location to edit user values for inputs to all steps in the sequence as well as a list of all *Tagged Outputs*. From here, you can view the value of *Tagged Outputs*, plot them, open them in the MATLAB variable editor, and save them to .mat files. You can select the type of plot to use by clicking *Line*, *Points*, *Semilog*, or *Bar*. See section 15 for more information.

21

12   Add a Directory to Function Library

Before you start building a sequence, you need to add a directory of MATLAB functions to your function library. It is recommended that you create a directory to store all of your modular functions as you develop them for use with mFUSE, but any directory that contains m-files can be used.

1. Select *Function Library* from the menu bar.

2. Choose *Add Library Path.*

3. A warning will appear indicating that mFUSE will modify your MATLAB search path. Click *OK.*
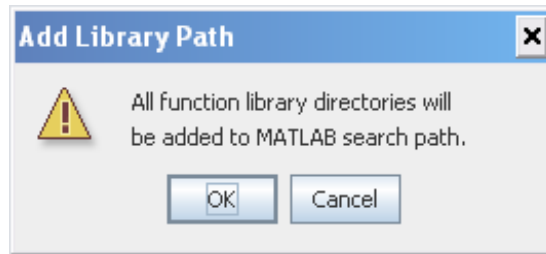
4. A file dialog will appear. Navigate into the directory you wish to add to your function library and click *Save.*

13   Creating a Sequence

The following sections cover the steps necessary to add steps to a sequence and configure each step.
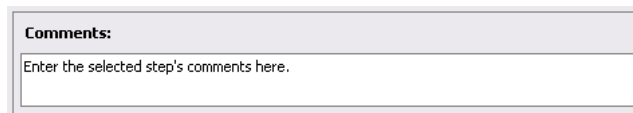
13.1   Adding Steps to Sequence

There are two methods to add steps to a sequence.

- *Method A*: Double-clicking a function in the *Function Library* will create a new step at the end of the sequence.

- *Method B*: Dragging a function from the *Function Library* will create a new step in the sequence at the location where the function is dropped in the *Sequence List*.

  1. Click and hold the mouse on the function in the *Function Library* you wish to add to the sequence.

  2. Continue holding the mouse button down and move the pointer to the location in the *Sequence List* where you wish the new step to be added.

  3. Release the mouse button.

13.2   Add Comments to Steps

To add or modify a step's comments, select the step in the *Sequence List* then enter the new comments in the *Comments* box at the top of the selected step area.

13.3   Select Call Method

If a function defines multiple call methods in its header, you can select the desired call method with these steps.

1. Select the step in the *Sequence List*.

2. Click the arrow to the right of the current call method in the *Call Methods* box of the selected step area.

3. A box will appear listing all available call methods with the current selection indicated. To change the selection, click on the desired call method.

Call Methods:

[Sum Of Inputs] = Sample Function (Input 1, Input 2)
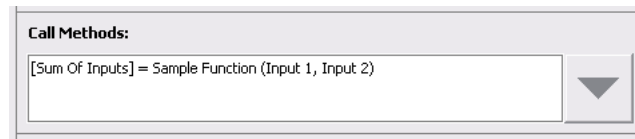
13.4   Modify Input Values

When a function is first added to a sequence, all of the new steps inputs will be set to default value. To enter a user value or select an output from a previous step, right-click on the value column of the input you wish to modify. A box will appear allowing you to click on one of the following options:

- *Default Value*: set input to use the default value defined in the function

- *Enter User Value*: manually enter the value to use for the input

- *Reset User Value*: reset the manually entered value to the format of the input as specified in the function header

- *Step Number and Name*

  1. Previous steps are listed by user and name. If there are many previous steps, they may be displayed in groups. Move the mouse over the name of the step you wish to connect to the input.

  2. A list of outputs from that step will appear. Click the output you wish to connect to the input.

Figure 4 shows many of the possible input choices and corresponding statuses.

Inputs:

Status	Variable	Value (Right-Click to Edit)
❌	No Value	Right-Click to Edit Value
⚠️	Unassigned Default	Default (Unassigned)
✔️	Assigned Default	Default
✔️	Default From Header	4 (Default)
❌	Blank User Value	Enter User Value
✔️	User Value	6

Figure 4: Input values and statuses.

24

13.5   Tag Outputs

To indicate important outputs in your sequence, tag them by following these
steps.

1. Select the step in the *Sequence List*.

2. Click the arrow to the right of the *Tag Outputs to View, Plot, and Save*
   box in the *Outputs* box of the selected step area.

3. A box will appear listing all outputs for the step with checkboxes indicat-
   ing which outputs are tagged.
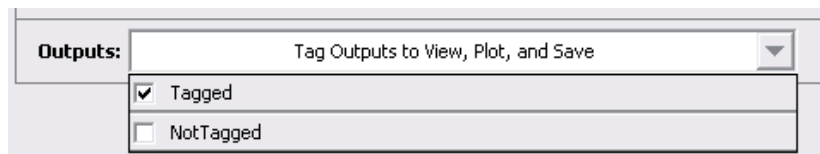
14 Executing a Sequence

Once all steps are added to your sequence and correctly configured, click the *Execute* button under the *Sequence List* to run the sequence in MATLAB. A message will appear to notify you that the sequence is being executed. You can click the *Cancel* button at any time to stop sequence execution.
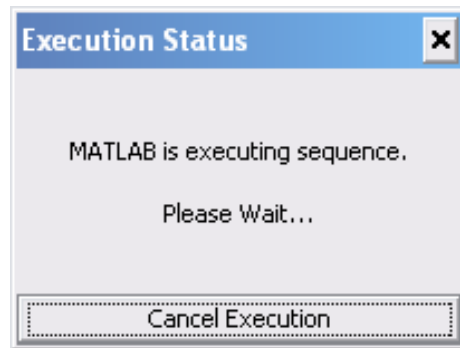
mFUSE will monitor MATLAB's status to detect if the sequence executed successfully, failed, or entered debug mode. If the sequence fails, an error message will appear providing basic error information.

15 Viewing Results

After a sequence has been executed, you can view, plot, and save tagged outputs from the *User Inputs/Tagged Outputs* tab.

- To view the value of an output, click on it in the *Tagged Outputs* list. Remember, values will only be updated after you reexecute the sequence. The value shown is generated by the *all2str.m* function included with mFUSE. You can modify how values are displayed in mFUSE by changing that function.

- To plot an output, click on it in the *Tagged Outputs* list. Select a *Plot Type*. Click the *Plot* button. A MATLAB figure window will appear.

- To open an output in the MATLAB Variable Editor, click on it in the *Tagged Outputs* list. Then, click the *Open* button.

- To save an output as a .mat file, click on it in the *Tagged Outputs* list. Then, click the *Save* button. A file dialog will appear allowing you to choose the save location and file name.
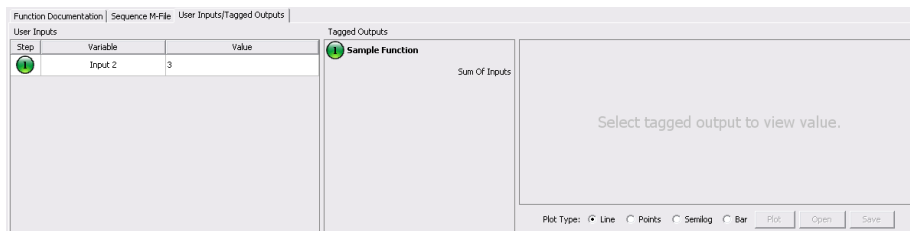
16   Saving Sequence M-File

To save a sequence as an m-file, open the *Sequence M-File* tab. You can enter a name for the sequence as well as other information in the *Sequence Information* box on the left. This information will be used to generate the header of the new m-file. The *M-File Options* box on the right provides all the available saving options.

There are two main types of m-files to choose from. M-files can be scripts that simply execute a series of MATLAB statements, or they can be functions that accept input arguments and produce output (like the functions used to create the sequence). Click either *Function* or *Script* in the *Choose M-File Type* box to make a selection.

If you choose to save a function, you will need to select which inputs and outputs from the sequence are used as function inputs and function outputs for the new m-file. The *Inputs* box allows you to select which user inputs in the sequence will be used as function inputs. Likewise, the *Outputs* box allows you to select which tagged outputs in the sequence will be used as function outputs. If you do not see an input or output in the list, go back to the step and make sure the input is set as a user value or the output is tagged.

Section 20 provides more information about other advanced saving options. You can preview the m-file to be saved in the *M-File Preview* box. In most cases, the m-file display will be automatically kept up-to-date. If the *Refresh M-File* button is displayed, the m-file is not being automatically updated, so you will need to click the *Refresh M-File* button to update the display. Once you are ready to save the sequence m-file, click the *Save M-File* button. A file dialog will appear allowing you to choose the save location and file name.
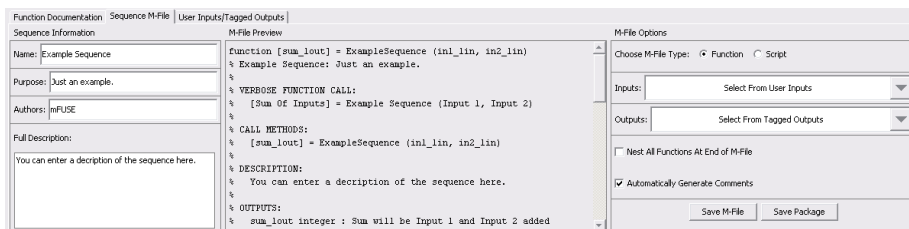
17   Saving mFUSE Session

If you would like to come back to a session at a later time to continue working on it, you can save the session as a .ses file.

1. Click *File* in the menu bar.

2. Click *Save Session.*

3. A file dialog will appear allowing you to choose the save location and file name.

You can open the session file later on by selecting *Open Session* from the file menu and choosing your saved .ses file.
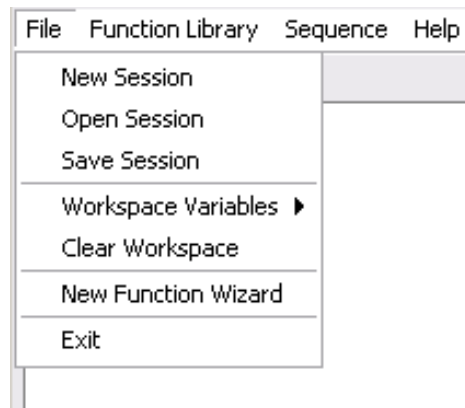
Part IV
Advanced Features

The following sections provide instructions and explanations for the many features in mFUSE not covered in part III.

　　　　　　　　30

18   Modifying a Sequence

18.1   Reordering Steps

Steps can be reordered in a sequence using a drag-and-drop operation. If you need to reorder the steps in a sequence, follow these steps.

1. Select the steps you wish to move. Multiple selections are allowed using the *Shift* key.

2. Click on selected step(s) and drag mouse while holding mouse button.

3. Move mouse to desired location in sequence.

4. Release mouse button.

After the drag-and-drop operation is complete, you may be prompted to confirm the move if it will cause connections between inputs and outputs in the sequence to be broken.
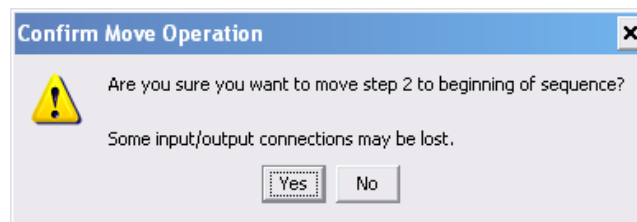
18.2   Copying Steps

Steps can be copied through a context menu accessed by right-clicking on the step you wish to copy. The copy operation creates a new step at the end of the sequence using the same function as the original step. The new step will have the default input and output configuration.
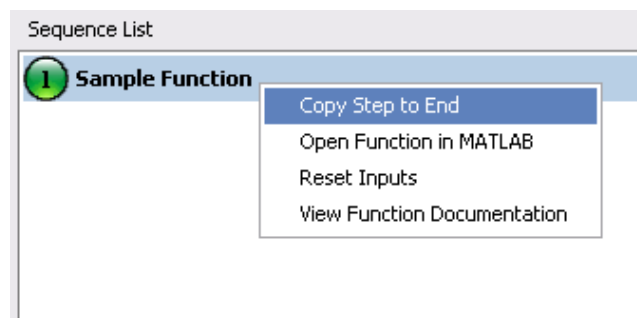
Note: The copy operation performs differently for Quick Steps. See section 19.3 for more information.

18.3   Removing Steps

Steps can be removed from a sequence using the remove button or *Delete* key.

1. Select the steps you wish to remove. Multiple selections are allowed using the *Shift* key.

2. Click the remove button shown with a red circle and an *X*, or press the *Delete* key.
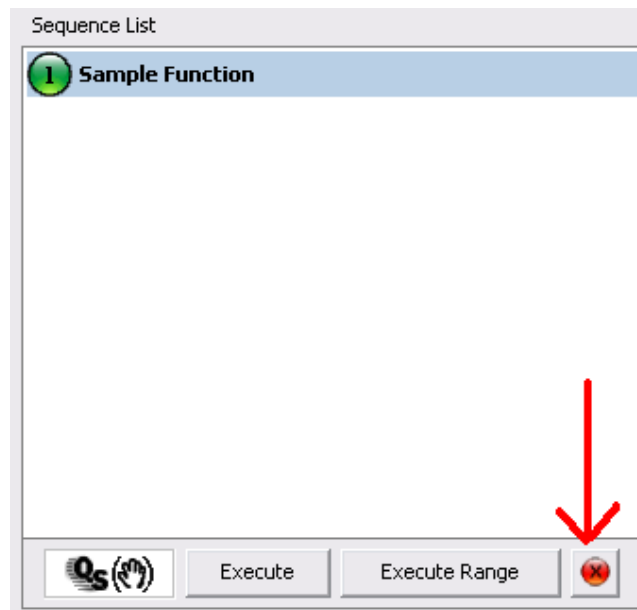
19 Quick Steps

Ideally, as you build your database of modular functions, you will be able to connect them seamlessly in any sequence. However, this may not always be the case. Quick Steps allow you to fill in the gaps between steps with a single line of code instead of creating an entirely new function. Each Quick Step contains a body line which you enter through the interface. mFUSE then analyzes the body to identify inputs to and outputs from the Quick Step. These inputs and outputs can be configured as with any other step.

Common uses for Quick Steps include but are not limited to the following:

- Indexing a variable

- Plotting commands

- Loading data files

- Saving variables to files

- Performing basic calculations

- Calling functions built-in to MATLAB or another toolbox

19.1 Adding a Quick Step To a Sequence

Follow these steps to create a new Quick Step using a drag-and-drop operation.

1. Click and hold the mouse on the *QS* icon beneath the *Sequence List*.

2. Continue holding the mouse button down and move the pointer to the location in the *Sequence List* where you wish the new Quick Step to be added.

3. Release the mouse button.
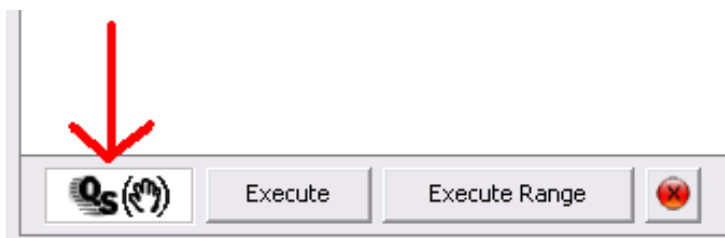
19.2 Writing Quick Step Bodies

After adding a new Quick Step to a sequence, the selected step area will display a *Body* box allowing you to enter the line of code for the Quick Step. Type the code in the box and click *OK*. The body can be edited later by clicking the *Edit Body* button. Input and output settings will be maintained after editing if variables names do not change.
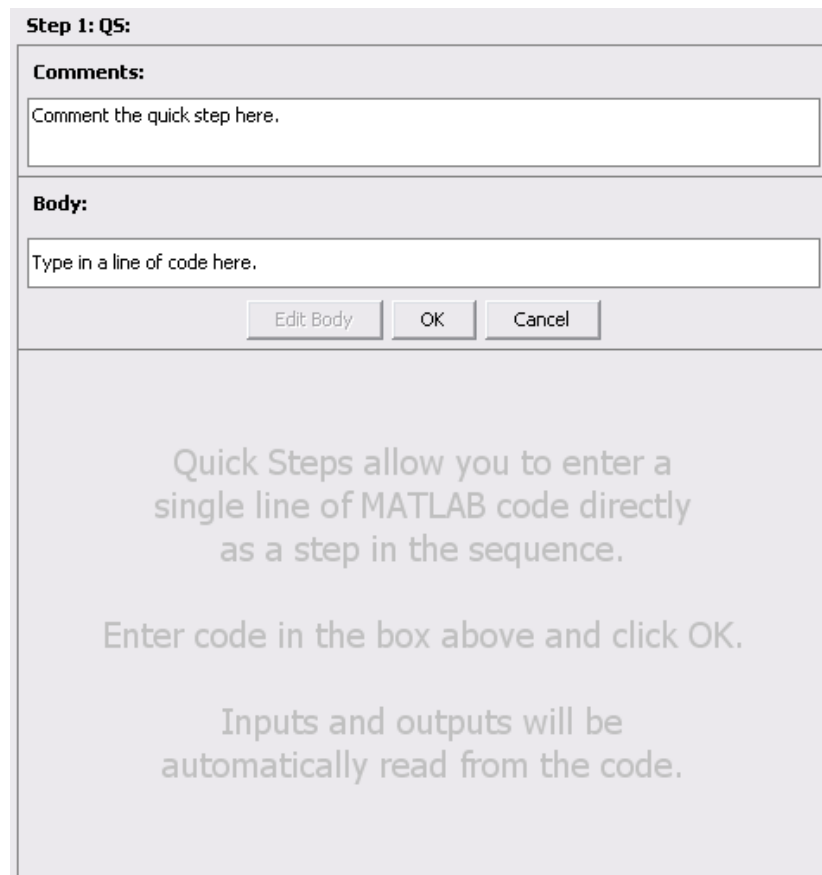
You may find certain commands that mFUSE does not interpret as you would expect, but there is usually an alternate form of the command that will work. For example, mFUSE will interpret the command to clear the workspace variables

```
clear all
```

to be two inputs name *clear* and *all*. In this case, an alternate form

```
clear('all')
```

will work as intended.

19.3   Copying Quick Steps

Quick Steps can be copied following the same method as other steps (see section 18.2) with an additional option. After selecting *Copy Step to End* from the Quick Step's context menu, the dialog shown in Figure 5 will appear asking if you want to link the new step to the old step. If you choose *Yes*, the two Quick Steps will continue to use the same body when either is edited. Linking does not affect the input and output settings for either step. If you choose *No*, the new Quick Step will initially have the same body as the original, but editing either body will not affect the other.
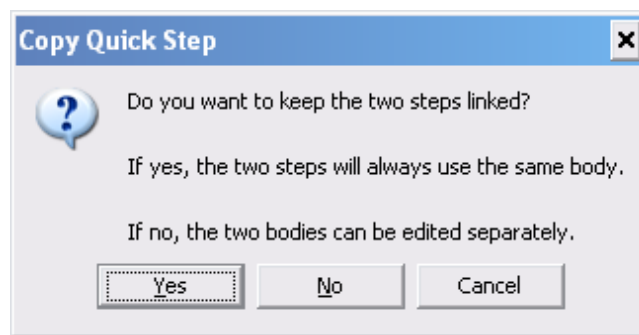
Figure 5: You have the option to link quick steps when copying them.

20 M-File Saving Options

When saving a sequence as an m-file, mFUSE provides many options in the saving process. The available options are different depending on your selection of save type, function or script.

20.1 Function Saving Options

When saving a function, the *Inputs* and *Outputs* boxes appear allowing you to choose which parameters will be used as the inputs and outputs to the new function. The current user value will be used as the default value in the new function for inputs.

The *Nest All Functions At End of M-File* option allows you to create a single m-file to execute the sequence without needing the functions from your function library. All the needed functions will be nested in the new function at the end of the file. mFUSE will attempt to correct any structural problems in the code that would prevent correct nesting. If problems are detected, you may see a message notifying you of the functions causing problems. Most often problems that occur when nesting functions are due to "end" statements. Because this option creates a lengthy m-file, automatic updating of the m-file preview is disabled when it is chosen. You will need to click the *Refresh M-File* button to update the preview.

The *Automatically Generate Comments* option adds a function header that follows the mFUSE function header standards as well as comments in cells for each step. The comments you have entered through the interface for each step will only appear in the m-file if this option is chosen.

20.2 Script Saving Options

When saving a script, the *Choose Variables to Save in Workspace* box appears allowing you to choose which parameters will remain in the MATLAB workspace after the script is executed. This option adds additional code at the beginning and end of each step to clear any variables that do not meet your selection. If *All* is chosen, no extra code will be added and all variables created during execution will remain in the MATLAB Workspace. If *Step Outputs* is chosen, all inputs will be cleared. If *Tagged Outputs* is chosen, all variables that are not tagged outputs will be cleared.

The *Combine Used Functions in Single M-File* option allows you to create a single m-file to execute the sequence without needing the functions from your function library. The body of each function used from the function library will be copied directly into the new m-file for each step. mFUSE will attempt to correct any structural problems in the code. If problems are detected, you may see a message notifying you of the functions causing problems. Most often problems that occur are due to "end" statements. Because this option creates a lengthy m-file, automatic updating of the m-file preview is disabled when it is chosen. You will need to click the *Refresh M-File* button to update the preview.

Note: The Combine Used Functions in Single M-File *option will not function correctly if any functions used in the sequence contain nested functions as nested functions are not allowed in MATLAB scripts.*

The *Automatically Generate Comments* option adds a function header as well as comments in cells for each step. The comments you have entered through the interface for each step will only appear in the m-file if this option is chosen. For scripts, the comments and header are formatted for use with MATLAB's publishing command.

The *Publish M-File* button only appears for scripts. This button will save the m-file in a location you choose through a file dialog then generate an HTML version of the script using MATLAB's publish command. The HTML file will include a table of contents, comments for each step, code, and any outputted results or plots.

M-File Options

Choose M-File Type: ○ Function   ● Script

Choose Variables to Save in Workspace:

● All   ○ Step Outputs   ○ Tagged Outputs

☐ Combine Used Functions in Single M-File

☑ Automatically Generate Comments

Save M-File   Save Package   Publish M-File

20.3   Saving a Package

Packages allow you to easily move a sequence and all the necessary files to another system. Packages are especially useful when collaborating on a project. The *Save Package* button performs the same actions as the *Save M-File* button with two additions. When saving a package, the functions used in the sequence from the function library will be copied into the same directory as the new m-file. Also, a session file will be saved in the directory with the m-file and functions. It is recommended to create a new directory to save a package in as packages can generate a large number of files.

Note: If the Nest All Functions At End of M-File *or* Combine Used Functions in Single M-File *option is chosen, the* Save Package *option will not be available as no additional functions will be required to use the new m-file.*

21   Menu Bar

The menu bar provides access to many tools and convenience methods to help you use mFUSE more effectively. The menu is divided into four sections, *File*, *Function Library*, *Sequence*, and *Help*.

21.1   File Menu

Sessions can be saved and opened using the *New Session*, *Open Session*, and *Save Session* options in the *File* Menu. Before creating a new session or opening a saved session, you will be asked to save your current session to avoid losing work. Only one session may be open at a time.

The *Workspace Variables* options allow you to choose which parameters will remain in the MATLAB workspace after the sequence is executed. If *All Variables* is chosen, no variables will be cleared. If *Step Outputs* is chosen, all inputs will be cleared. If *Tagged Outputs* is chosen, all variables that are not tagged outputs will be cleared.

The *New Function Wizard* generates new functions with headers that follow the mFUSE function header standard. See section 25 for more information.

Exit closes the mFUSE interface after prompting you to save the current session.
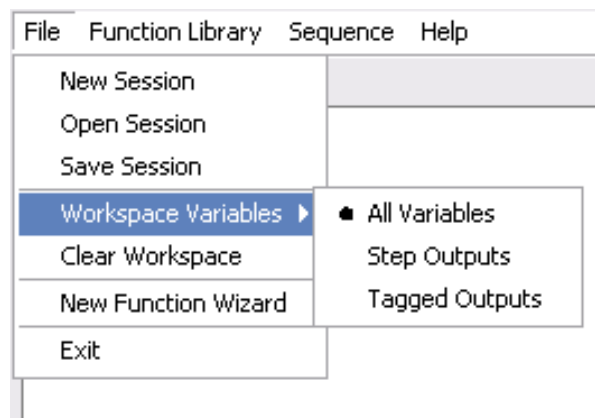
21.2   Function Library Menu

The *Function Library* menu provides convenience methods to update and edit your function library.

Refresh Function Library will reload all directories in your function library. Any steps that use functions from the function library will be updated to reflect changes in the function m-file. Input and output settings will not be lost if variable names remain the same.

Add Library Path will open a file dialog to allow you to choose a directory to add to your function library. Navigate into the directory you wish to add and click *Save.*
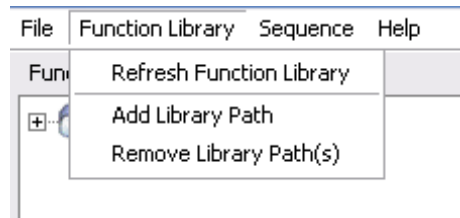
Remove Library Path will open a list of directories in your function library. Check the boxes in the *Remove* column next to directories you wish to remove and click *OK.*
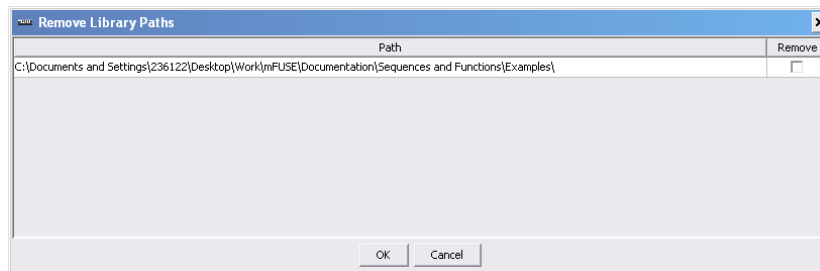
21.3   Sequence Menu

The *Sequence* menu provides convenience methods to edit all steps in a sequence at one time.

Reset All Inputs will set all the inputs in the sequence back to their original value. *Reset all user values* will return any user values to their default state.

Tag all outputs and *Untag all outputs* will set the tagged state of all the outputs in the sequence.
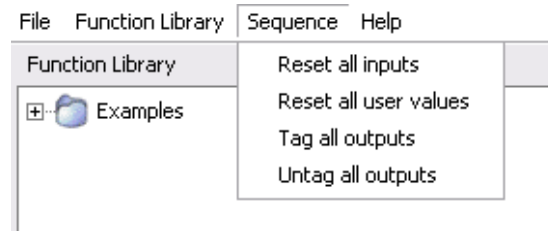
21.4　Help Menu

The *Help* menu provides information about your function library, access to this help document, and general information about mFUSE.
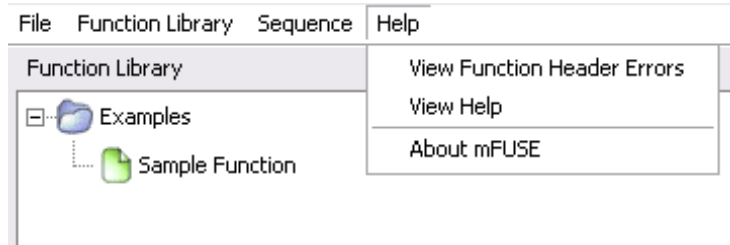
View Function Header Errors opens a window listing any missing or misformatted header sections detected in the functions in your function library along with hints to help you fix detected problems.

View Help opens this help document in your system's standard PDF viewer.

About mFUSE opens the about window which lists your mFUSE version and other general information.
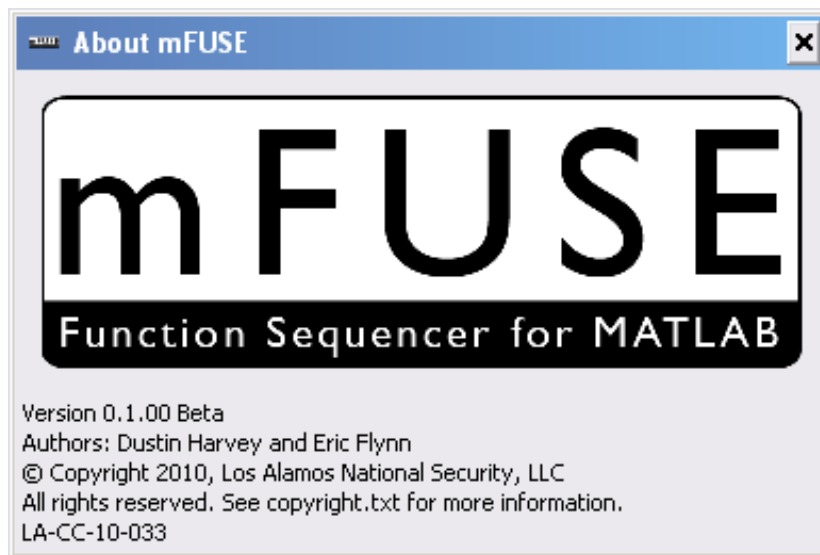
22　Partial Sequence Execution

When working with sequences that have long execution times, it is sometimes convenient to reexecute a small portion of the sequence to view changes. The *Execute Range* button allows you to execute only the selected steps in the *Sequence List*. The selection of multiple steps is allowed using the *Shift* key. It is recommended to set the *Workspace Variables* option to *All Variables* or *Step Outputs* in the *File* menu (see section 21.1) when using partial sequence execution.
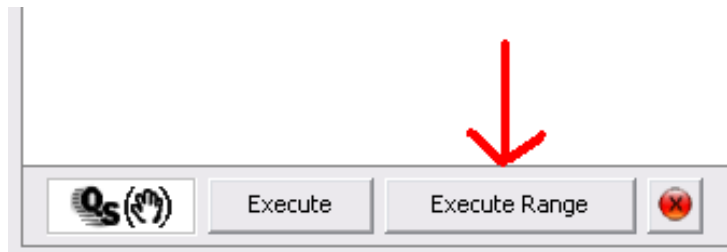
23   Hiding Tabbed Area

To save space when mFUSE is used on a small resolution display, the bottom tabbed area can be hidden by clicking the upward pointing arrow in the middle of the bar that contains the copyright notice. Once the tabs are hidden, they can be shown by clicking the downward pointing arrow in the same location.
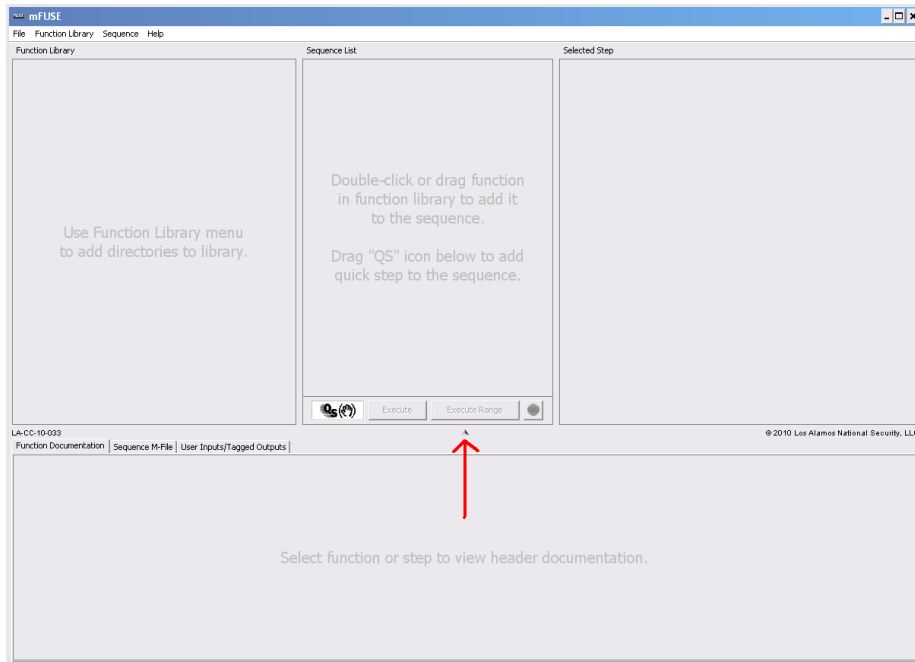
24   Editing Function M-Files

Follow these steps to edit the functions in your function library:

1. Right-Click on the function in the *Function Library* you wish to edit to open the context menu.

2. Click *Open Function in MATLAB* to open MATLAB's Editor.

3. Make edits in MATLAB's Editor and save m-file.

4. Right-Click in *Function Library* to open context menu.

5. Click *Refresh Function Library* to update your functions.

   Note: You can also refresh the Function Library *by following the steps in section 21.2.*
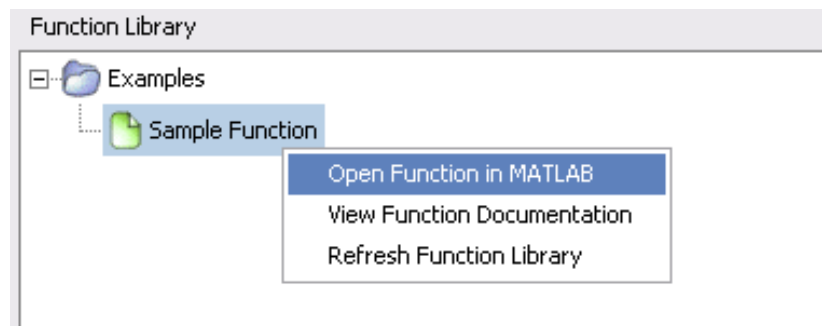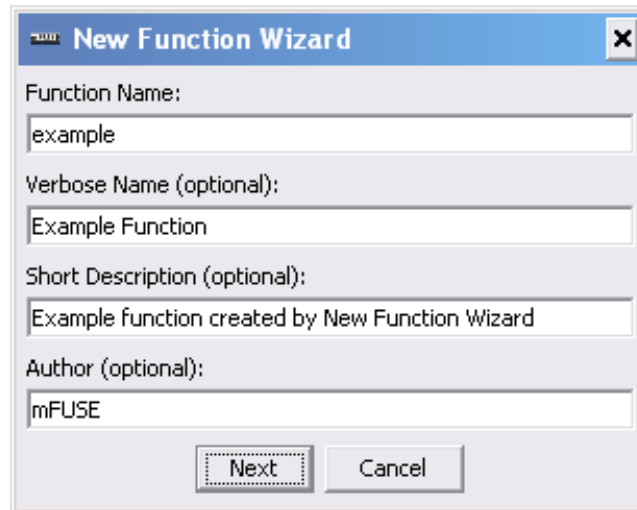
25   New Function Wizard

The *New Function Wizard* provides a series of dialogs to enter information about a new function. mFUSE will then use the entries to generate a new m-file that follows the mFUSE function header standard. To start the wizard, select *New Function Wizard* from the *File* menu.

25.1   Basic Function Information

In the first dialog, enter the name of the new function in the *Function Name* box and a more descriptive name for the function in the *Verbose Name* box. The function name must follow MATLAB's rules for naming functions while the verbose name can include any characters. You can also enter a short description for the function and the author's name.
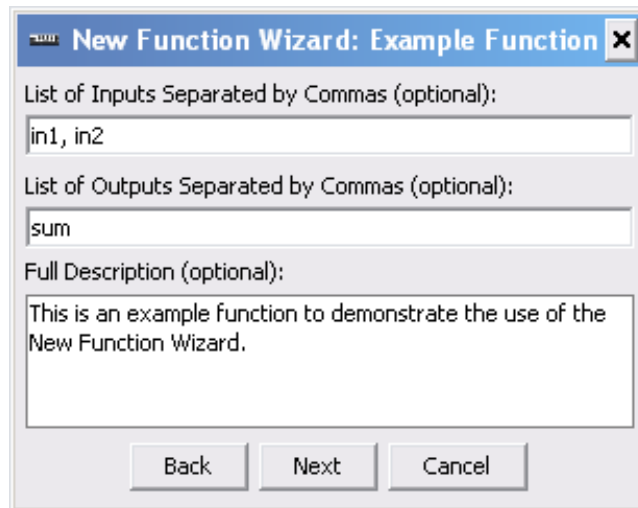
25.2   Parameters and Full Description

In the second dialog, enter comma separated lists for the function's inputs and outputs. Each variable must follow MATLAB's rules for naming variables. You may also enter a more detailed description of the function in the *Full Description* box.

25.3   Detailed Inputs Information
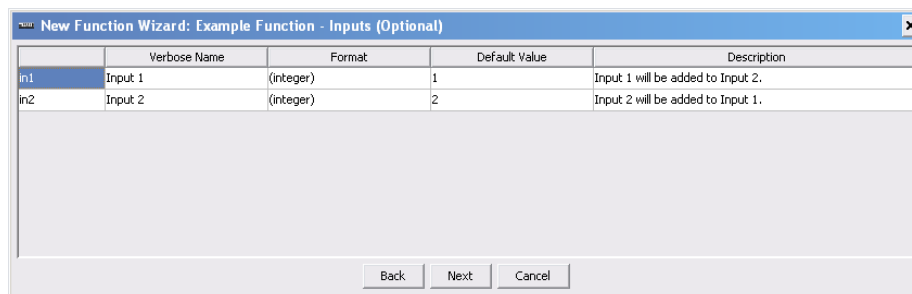
If you entered any inputs for the new function, the next dialog will contain a table allowing you to enter more information about each input.

In the *Verbose Name* column enter a descriptive name for the input.

In the *Format* column, enter the format for the input in parentheses. For example, "(integer)" or "(string)". The format column will be used by mFUSE to provide the initial user value for that input.

In the *Default Value* column, enter the default value for the input if one exists. The default value will be coded into the new function immediately following the header.

In the *Description* column, enter a description for the input.

25.4   Detailed Outputs Information

If you entered any outputs for the new function, the next dialog will contain a table allowing you to enter more information about each output.

In the *Verbose Name* column enter a descriptive name for the output.

In the *Format* column, enter the format of the output in parentheses. For example, "(integer)" or "(string)".

In the *Description* column, enter a description of the output.

	Verbose Name	Format	Description
sum	Sum of Inputs	(integer)	Sum will be Input 1 and Input 2 added together.

New Function Wizard: Example Function - Outputs (Optional)

Back   Create M-File   Cancel
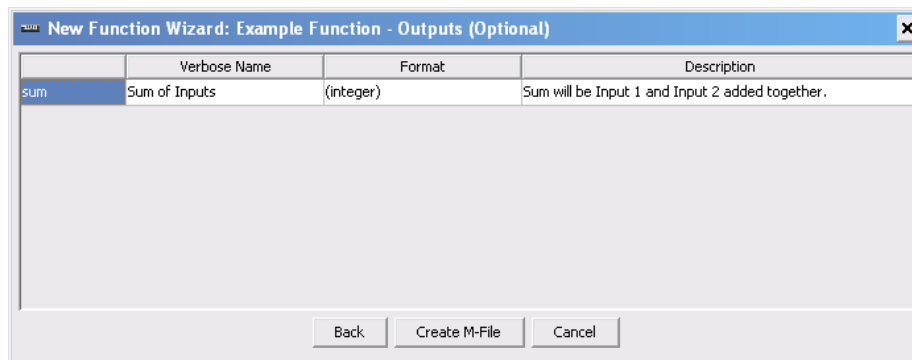
25.5   Wizard Complete

When the wizard is complete, a file dialog will appear allowing you to save the new function m-file. After saving, a message will appear to allow you to open the new function in MATLAB's Editor.

New Function Wizard Complete

Would you like to edit the new function in MATLAB's editor?

Yes   No
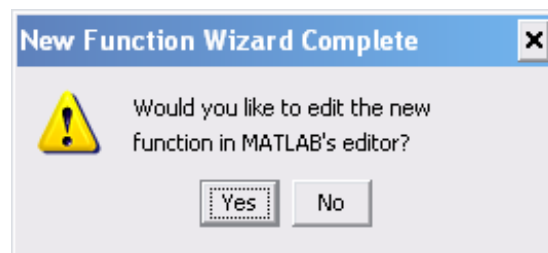
25.6   New Function

The following is an example of a function created by the *New Function Wizard*
that follows the mFUSE function header standard:

```
function [sum] = example (in1, in2)
% Auto-Generated by mFUSE: Example function created by New Function Wizard
%
% VERBOSE FUNCTION CALL:
%    [Sum of Inputs] = Example Function (Input 1, Input 2)
%
% CALL METHODS:
%    [sum] = example (in1, in2)
%
% DESCRIPTION:
%    This is an example function to demonstrate the use of the New Function Wizard.
%
% OUTPUTS:
%    sum (integer) : Sum will be Input 1 and Input 2 added together.
%
% INPUTS:
%    in1 (integer) : Input 1 will be added to Input 2.
%    in2 (integer) : Input 2 will be added to Input 1.
%
% DEFAULT VALUES:
%    in1 = 1
%    in2 = 2


% AUTHOR:
%    mFUSE
%
% DATE CREATED:
%    1/20/2010


% Set Input Defaults:
if isempty(in1), in1 = 1; end
if isempty(in2), in2 = 2; end



end
```

26   Function Header Standard

The mFUSE function header standard specifies various sections of the header that mFUSE will recognize. Each section is optional, and the sections can be listed in any order though the following order is preferred:

1. Verbose Function Call

2. Call Methods

3. Description

4. Outputs

5. Inputs

6. Default Values

7. See Also

8. Authors

9. Date Created

10. Modifications

11. References

12. Sample Data Set

13. Data Acquisition

14. MATLAB Functions Called

mFUSE only uses the information in the first 6 sections listed above. The other sections are defined to allow you to include other information without causing problems in mFUSE. Any information in the header should be contained in one of the above sections. A blank line should be entered after the See Also section for use with MATLAB's help command.

All lines in the header must start with "%" to indicate a MATLAB comment line. Sections begin with a section title in all capital letters followed by a colon. For example the title line of the Verbose Function Call section should look like this:

```
% VERBOSE FUNCTION CALL:
```

See section 25.6 for an example of a complete function header that follows the mFUSE function header standard.

Note: mFUSE will ignore any comments in the header after a blank line (no "%"). You can include anything in the header after a blank line.

26.1   Short Description

The first line of a MATLAB function must contain the function call which is not a commented line and starts with the word "function." mFUSE allows you to provide a short description in the second line (first line of the header) without a section title. The short description should follow a colon (":"). The space before the colon can be used to indicate other information about the function such as a group of functions the function belongs to.

In the example below, the first line contains the function call, and the second line indicates that the function was automatically generated by mFUSE along with a short description after the colon. The short description of the function is used throughout the mFUSE interface to describe the function.

```
function [sum] = example (in1, in2)
% Auto-Generated by mFUSE: Example function created by New Function Wizard
```

26.2   Verbose Function Call

The verbose function call provides more descriptive names for the inputs, outputs, and the function itself. It should match the format of the function call in the first line with the same number of inputs and outputs in the same order. Verbose function call entries may use multiple lines. Inputs and outputs should be entered as comma-separated lists as spaces are allowed in each parameter name.

```
% VERBOSE FUNCTION CALL:
%   [Sum of Inputs] = Example Function (Input 1, Input 2)
```

26.3   Call Methods

The call methods section is only necessary for functions that have multiple call methods. If multiple call methods are listed, mFUSE will allow you to choose the desired call method from within the mFUSE interface when using the function. The first entry should be identical to the function call in line 1 with each optional call method defined on a new line. Call method entries may use multiple lines.

```
% CALL METHODS:
%   [sum] = example (in1, in2)
```

26.4   Description

The description section allows you to enter a description for the function that covers multiple lines.

```
% DESCRIPTION:
%   This is an example function to demonstrate the use of the New Function Wizard.
```

26.5   Outputs and Inputs

The Outputs and Inputs sections use identical formats. Each entry contains
the name of the variable, the data format, and a description of the variable.
The variable name must match the name used in the function call in line 1
or one of the call methods exactly. The format follows the variable name and
is placed inside parentheses followed by a colon. The description follows the
colon. Information provided in the header about inputs and outputs is used
throughout the mFUSE interface.

```
% OUTPUTS:
%   sum (integer) : Sum will be Input 1 and Input 2 added together.
%
% INPUTS:
%   in1 (integer) : Input 1 will be added to Input 2.
%   in2 (integer) : Input 2 will be added to Input 1.
```

26.6   Default Values

If a default value is defined for an input within the function, it should be specified
in the default values section of the header. Entries in the default value section
notify mFUSE that the default value exists and allows you to see the default
value from within the interface. Each entry should start with the name of an
input identical to an input in the function call in line 1 or one of the call methods.
The input name is followed by an equals sign ("=") and the default value.

```
% DEFAULT VALUES:
%   in1 = 1
%   in2 = 2
```

26.7   Other Allowed Sections

The following sections are allowed by the mFUSE header standard, but the information contained in each section is not used by the mFUSE interface. Each section should start with a title line following the same specifications as other sections.

- See Also

- Authors

- Date Created

- Modifications

- References

- Sample Data Set

- Data Acquisition

- MATLAB Functions Called

Note: The see also section has special meaning in MATLAB. Each function referenced in this section should be entered on its own line.

```
% SEE ALSO:
%    function1
%    function2
%    function3
```

Part V
Appendices

A   Troubleshooting

This section describes common problems encountered while using mFUSE and solutions to those problems as well as other support information.

A.1   Problems and Solutions

A.1.1   *PROBLEM: mFUSE crashed MATLAB and I lost all my work.*

SOLUTION: mFUSE has been found to be more stable when run in Java 6/1.6. Update your Java installation to the latest version available to increase stability. Also, while your work in mFUSE will be automatically recovered after a crash, other work in MATLAB may not be. It is not recommended to leave mFUSE open when not in use.

A.1.2   *PROBLEM: Sequence execution is taking much longer than I expected.*

SOLUTION: It is possible that MATLAB has finished executing your sequence, but mFUSE hasn't removed the Execution Status dialog. Try using the Cancel Execution button on the dialog. If it does not respond, you will need to force MATLAB to close then reopen it.

A.1.3   *PROBLEM: I can't see the bottom tabbed area of the interface.*

SOLUTION: The bottom tabbed area can be hidden using the arrow located between the top and bottom sections. Click the arrow to show the tabs. See section 23

A.1.4   *PROBLEM: I received an error during installation similar to "Error writing to classpath.txt: Obtain write permission for following file before installing."*

SOLUTION: For mFUSE to be installed, MATLAB must be able to write to the classpath.txt file. If your account does not have write permissions to that file, you will need to obtain permissions before running the installation script. You can locate the claasspath.txt file using the following command:

```
which ('classpath.txt')
```

A.2   Contact mFUSE Team

Visit http://institute.lanl.gov/ei/software-and-data/ for updates to mFUSE and to contact the mFUSE team with comments or suggestions. Technical support questions can not be answered on an individual basis, but bugs will be fixed in future releases.

B   Example Saved Files

This section provides examples of the m-files that mFUSE generates and a saved session file. The sequence itself is trivial in this example, but serves as a demonstration of the saving options in mFUSE. The sequence uses two functions which are shown in the next section. The add function takes two inputs and adds them together. The multiply function multiplies two inputs and has a default value of 4 for the second input. Quick Steps are used to define A and B at the beginning of the sequence. Additionally, a Quick Step is used to divide by a factor of two. The sequence performs the following steps given two inputs (A and B) to calculate the value of C. The calculation involved is

$C = ((A * B)/2) + 4 * A$

The sequence follows these steps to perform the calculation.

1. Define A and B to be 1 and 2 respectively.

2. Multiply A and B.

3. Divide the product by 2.

4. Multiply A by 4.

5. Add the quotient and second product.

B.1 Functions Used

B.1.1 Add Function

```
function [sum] = Add (in1, in2)
% Auto-Generated by mFUSE: Calculates the sum.
%
% VERBOSE FUNCTION CALL:
% [Sum] = Add Two Numbers (Input 1, Input 2)
%
% CALL METHODS:
% [sum] = Add (in1, in2)
%
% DESCRIPTION:
% Calculates the sum of two numbers.
%
% OUTPUTS:
% sum (integer) : Sum of Inputs
%
% INPUTS:
% in1 (integer) : First Input
% in2 (integer) : Second Input
%
% DEFAULT VALUES:
%
% AUTHOR:
% mFUSE
%
% DATE CREATED:
% 1/20/2010


sum = in1 + in2;


end
```

B.1.2 Multiply Function

```
function [product] = Multiply (in1, in2)
% Auto-Generated by mFUSE: Calculates the product.
%
% VERBOSE FUNCTION CALL:
% [Product] = Multiply Two Numbers (Input 1, Input 2)
%
% CALL METHODS:
% [product] = Multiply (in1, in2)
%
% DESCRIPTION:
```

```
% Calculates the product of two numbers.
%
% OUTPUTS:
% product (integer) : Product of Inputs
%
% INPUTS:
% in1 (integer) : First Input
% in2 (integer) : Second Input
%
% DEFAULT VALUES:
% in2 = 4
%
% AUTHOR:
% mFUSE
%
% DATE CREATED:
% 1/20/2010

% Set Input Defaults:
if isempty(in2), in2 = 4; end

product = in1 + in2;

end
```

B.2 Sequence M-File As Function

```
function [Sum_6out] = ExampleSequenceForSavedFiles (integer_1in, integer_2in)
% Example Sequence for Saved Files: Performs Basic Arithmetic
%
% VERBOSE FUNCTION CALL:
% [Sum] = Example Sequence for Saved Files (integer, integer)
%
% CALL METHODS:
% [Sum_6out] = ExampleSequenceForSavedFiles (integer_1in, integer_2in)
%
% DESCRIPTION:
% Performs Basic Arithmetic
%
% out = ( (in1 * in2 ) / 2) + 4 * in1
%
% OUTPUTS:
% Sum_6out (integer) : Sum of Inputs
%
% INPUTS:
% integer_1in  : Quick Step Input
% integer_2in  : Quick Step Input
%
% DEFAULT VALUES:
% integer_1in = 1
% integer_2in = 2
%
% AUTHOR:
% mFUSE
%
% DATE CREATED:
% 1/20/2010

% Set Input Defaults:
if isempty(integer_1in), integer_1in = 1; end
if isempty(integer_2in), integer_2in = 2; end



%% Step 1: QS: A=integer;

% Set Function Inputs:
integer = integer_1in;

% Step Body:
A=integer;
```

```
% Save Function Outputs:
A_1out = A;



%% Step 2: QS: B=integer;

% Set Function Inputs:
integer = integer_2in;

% Step Body:
B=integer;

% Save Function Outputs:
B_2out = B;



%% Step 3: Multiply Two Numbers
% Calculates the product.

% Function Call: [Product] = Multiply Two Numbers (Input 1, Input 2)
[Product_3out] = Multiply(A_1out, B_2out);



%% Step 4: QS: Quotient = integer/2;

% Step Inputs:
integer_4in = Product_3out;

% Set Function Inputs:
integer = integer_4in;

% Step Body:
Quotient = integer/2;

% Save Function Outputs:
Quotient_4out = Quotient;



%% Step 5: Multiply Two Numbers
% Calculates the product.
```

```matlab
% Function Call: [Product] = Multiply Two Numbers (Input 1, Input 2)
[Product_5out] = Multiply(A_1out, []);




%% Step 6: Add Two Numbers
% Calculates the sum.

% Function Call: [Sum] = Add Two Numbers (Input 1, Input 2)
[Sum_6out] = Add(Quotient_4out, Product_5out);

% End ExampleSequenceForSavedFiles

end
```

B.3   Sequence M-File As Function In Single File

```
function [Sum_6out] = ExampleSequenceForSavedFiles (integer_1in, integer_2in)
% Example Sequence for Saved Files: Performs Basic Arithmetic
%
% VERBOSE FUNCTION CALL:
% [Sum] = Example Sequence for Saved Files (integer, integer)
%
% CALL METHODS:
% [Sum_6out] = ExampleSequenceForSavedFiles (integer_1in, integer_2in)
%
% DESCRIPTION:
% Performs Basic Arithmetic
%
% out = ( (in1 * in2 ) / 2) + 4 * in1
% All required functions nested at end of file.
%
% OUTPUTS:
% Sum_6out (integer) : Sum of Inputs
%
% INPUTS:
% integer_1in  : Quick Step Input
% integer_2in  : Quick Step Input
%
% DEFAULT VALUES:
% integer_1in = 1
% integer_2in = 2
%
% AUTHOR:
% mFUSE
%
% DATE CREATED:
% 1/20/2010

% Set Input Defaults:
if isempty(integer_1in), integer_1in = 1; end
if isempty(integer_2in), integer_2in = 2; end



%% Step 1: QS: A=integer;

% Set Function Inputs:
integer = integer_1in;

% Step Body:
```

```
A=integer;

% Save Function Outputs:
A_1out = A;



%% Step 2: QS: B=integer;

% Set Function Inputs:
integer = integer_2in;

% Step Body:
B=integer;

% Save Function Outputs:
B_2out = B;



%% Step 3: Multiply Two Numbers
% Calculates the product.

% Function Call: [Product] = Multiply Two Numbers (Input 1, Input 2)
[Product_3out] = Multiply(A_1out, B_2out);



%% Step 4: QS: Quotient = integer/2;

% Step Inputs:
integer_4in = Product_3out;

% Set Function Inputs:
integer = integer_4in;

% Step Body:
Quotient = integer/2;

% Save Function Outputs:
Quotient_4out = Quotient;



%% Step 5: Multiply Two Numbers
% Calculates the product.
```

```
% Function Call: [Product] = Multiply Two Numbers (Input 1, Input 2)
[Product_5out] = Multiply(A_1out, []);



%% Step 6: Add Two Numbers
% Calculates the sum.

% Function Call: [Sum] = Add Two Numbers (Input 1, Input 2)
[Sum_6out] = Add(Quotient_4out, Product_5out);

% End ExampleSequenceForSavedFiles

end



%% Start Nested Function: Multiply Two Numbers
%
% Calculates the product.

function [product] = Multiply (in1, in2)
% Auto-Generated by mFUSE: Calculates the product.
%
% VERBOSE FUNCTION CALL:
% [Product] = Multiply Two Numbers (Input 1, Input 2)
%
% CALL METHODS:
% [product] = Multiply (in1, in2)
%
% DESCRIPTION:
% Calculates the product of two numbers.
%
% OUTPUTS:
% product (integer) : Product of Inputs
%
% INPUTS:
% in1 (integer) : First Input
% in2 (integer) : Second Input
%
% DEFAULT VALUES:
% in2 = 4
%
% AUTHOR:
% mFUSE
```

```
%
% DATE CREATED:
% 1/20/2010

% Set Input Defaults:
if isempty(in2), in2 = 4; end

product = in1 + in2;

end




%% Start Nested Function: Add Two Numbers
%
% Calculates the sum.

function [sum] = Add (in1, in2)
% Auto-Generated by mFUSE: Calculates the sum.
%
% VERBOSE FUNCTION CALL:
% [Sum] = Add Two Numbers (Input 1, Input 2)
%
% CALL METHODS:
% [sum] = Add (in1, in2)
%
% DESCRIPTION:
% Calculates the sum of two numbers.
%
% OUTPUTS:
% sum (integer) : Sum of Inputs
%
% INPUTS:
% in1 (integer) : First Input
% in2 (integer) : Second Input
%
% DEFAULT VALUES:
%
% AUTHOR:
% mFUSE
%
% DATE CREATED:
% 1/20/2010

sum = in1 + in2;
```

end

B.4   Sequence M-File As Script

```
% Example Sequence for Saved Files: Performs Basic Arithmetic
%
% DESCRIPTION:
% Performs Basic Arithmetic
%
% out = ( (in1 * in2 ) / 2) + 4 * in1
%
% AUTHOR:
% mFUSE
%
% DATE CREATED:
% 1/20/2010



%% Step 1: QS: A=integer;

% Step Inputs:
integer_1in = 1;

% Set Function Inputs:
integer = integer_1in;

% Step Body:
A=integer;

% Save Function Outputs:
A_1out = A;



%% Step 2: QS: B=integer;

% Step Inputs:
integer_2in = 2;

% Set Function Inputs:
integer = integer_2in;

% Step Body:
B=integer;

% Save Function Outputs:
B_2out = B;
```

```
%% Step 3: Multiply Two Numbers
% Calculates the product.

% Function Call: [Product] = Multiply Two Numbers (Input 1, Input 2)
[Product_3out] = Multiply(A_1out, B_2out);



%% Step 4: QS: Quotient = integer/2;

% Step Inputs:
integer_4in = Product_3out;

% Set Function Inputs:
integer = integer_4in;

% Step Body:
Quotient = integer/2;

% Save Function Outputs:
Quotient_4out = Quotient;



%% Step 5: Multiply Two Numbers
% Calculates the product.

% Function Call: [Product] = Multiply Two Numbers (Input 1, Input 2)
[Product_5out] = Multiply(A_1out, []);



%% Step 6: Add Two Numbers
% Calculates the sum.

% Function Call: [Sum] = Add Two Numbers (Input 1, Input 2)
[Sum_6out] = Add(Quotient_4out, Product_5out);

% End ExampleSequenceForSavedFiles
```

B.5 Sequence M-File As Script In Single File

```
% Example Sequence for Saved Files: Performs Basic Arithmetic
%
% DESCRIPTION:
% Performs Basic Arithmetic
%
% out = ( (in1 * in2 ) / 2) + 4 * in1
%
% AUTHOR:
% mFUSE
%
% DATE CREATED:
% 1/20/2010



%% Step 1: QS: A=integer;

% Step Inputs:
integer_1in = 1;

% Set Function Inputs:
integer = integer_1in;

% Step Body:
A=integer;

% Save Function Outputs:
A_1out = A;



%% Step 2: QS: B=integer;

% Step Inputs:
integer_2in = 2;

% Set Function Inputs:
integer = integer_2in;

% Step Body:
B=integer;

% Save Function Outputs:
B_2out = B;
```

```
%% Step 3: Multiply Two Numbers
% Calculates the product.

% Step Inputs:
Input1_3in = A_1out;
Input2_3in = B_2out;

% Set Function Inputs:
in1 = Input1_3in;
in2 = Input2_3in;

% Copy of Function Body:

if isempty(in2), in2 = 4; end

product = in1 + in2;

% Save Function Outputs:
Product_3out = product;



%% Step 4: QS: Quotient = integer/2;

% Step Inputs:
integer_4in = Product_3out;

% Set Function Inputs:
integer = integer_4in;

% Step Body:
Quotient = integer/2;

% Save Function Outputs:
Quotient_4out = Quotient;



%% Step 5: Multiply Two Numbers
% Calculates the product.

% Step Inputs:
Input1_5in = A_1out;
```

```
Input2_5in = [];

% Set Function Inputs:
in1 = Input1_5in;
in2 = Input2_5in;

% Copy of Function Body:

if isempty(in2), in2 = 4; end

product = in1 + in2;

% Save Function Outputs:
Product_5out = product;



%% Step 6: Add Two Numbers
% Calculates the sum.

% Step Inputs:
Input1_6in = Quotient_4out;
Input2_6in = Product_5out;

% Set Function Inputs:
in1 = Input1_6in;
in2 = Input2_6in;

% Copy of Function Body:

sum = in1 + in2;

% Save Function Outputs:
Sum_6out = sum;

% End ExampleSequenceForSavedFiles
```

B.6   Saved Session File

```
% mFUSE Saved Session File
VERSION:-> 0.2.00
NAME:-> Example Sequence for Saved Files
% Sequence Information
SHORT DESCRIPTION:-> Performs Basic Arithmetic
AUTHORS:-> mFUSE
LONG DESCRIPTION:->
Performs Basic Arithmetic

out = ( (in1 * in2 ) / 2) + 4 * in1
<-:END LONG DESCRIPTION
% Selected Options
VARIABLES TO WORKSPACE:-> STEP
SAVE TYPE:-> FUNCTION
SCRIPT VARIABLES:-> ALL
COMBINE FUNCTIONS:-> FALSE
GENERATE COMMENTS:-> TRUE
PUBLISH M-FILE:-> FALSE
% Sequence Steps

STARTSTEP:-> 1
QUICKSTEP:-> TRUE
BODY:-> A=integer;
INPUTS:->
integer<U>1<V>USER
<-:INPUTS
OUTPUTS:->
A<T>FALSE
<-:OUTPUTS
COMMENTS:->

<-:END COMMENTS
<-:ENDSTEP

STARTSTEP:-> 2
QUICKSTEP:-> TRUE
BODY:-> B=integer;
INPUTS:->
integer<U>2<V>USER
<-:INPUTS
OUTPUTS:->
B<T>FALSE
<-:OUTPUTS
COMMENTS:->
```

```
<-:END COMMENTS
<-:ENDSTEP

STARTSTEP:-> 3
QUICKSTEP:-> FALSE
FILENAME:-> Multiply.m
USED CALL METHOD:-> [product] = Multiply (in1, in2)
INPUTS:->
in1<U>(integer)<V>A_1
in2<U>(integer)<V>B_2
<-:INPUTS
OUTPUTS:->
product<T>FALSE
<-:OUTPUTS
COMMENTS:->
Calculates the product.
<-:END COMMENTS
<-:ENDSTEP

STARTSTEP:-> 4
QUICKSTEP:-> TRUE
BODY:-> Quotient = integer/2;
INPUTS:->
integer<U><V>product_3
<-:INPUTS
OUTPUTS:->
Quotient<T>FALSE
<-:OUTPUTS
COMMENTS:->

<-:END COMMENTS
<-:ENDSTEP

STARTSTEP:-> 5
QUICKSTEP:-> FALSE
FILENAME:-> Multiply.m
USED CALL METHOD:-> [product] = Multiply (in1, in2)
INPUTS:->
in1<U>(integer)<V>A_1
in2<U>(integer)<V>DEFAULT
<-:INPUTS
OUTPUTS:->
product<T>FALSE
<-:OUTPUTS
COMMENTS:->
```

```
Calculates the product.
<-:END COMMENTS
<-:ENDSTEP

STARTSTEP:-> 6
QUICKSTEP:-> FALSE
FILENAME:-> Add.m
USED CALL METHOD:-> [sum] = Add (in1, in2)
INPUTS:->
in1<U>(integer)<V>Quotient_4
in2<U>(integer)<V>product_5
<-:INPUTS
OUTPUTS:->
sum<T>TRUE
<-:OUTPUTS
COMMENTS:->
Calculates the sum.
<-:END COMMENTS
<-:ENDSTEP
FUNCTION INPUTS:->
integer_1
integer_2
<-:FUNCTION INPUTS
FUNCTION OUTPUTS:->
sum_6
<-:FUNCTION OUTPUTS
ENDFILE
```