Chapter 6

Debugging M-files

6.1 Introduction

This section introduces general techniques for finding *errors* in M-files. *Debugging* is the process by which you isolate and fix *errors* in your program or code.

Debugging helps to correct two kind of errors:

- **Syntax errors** - For example omitting a parenthesis or misspelling a function name.
- **Run-time errors** - Run-time errors are usually apparent and difficult to track down. They produce unexpected results.

6.2 Debugging process

We can debug the M-files using the Editor/Debugger as well as using debugging functions from the Command Window. The debugging process consists of

- Preparing for debugging
- Setting breakpoints
- Running an M-file with breakpoints
- Stepping through an M-file
- Examining values
- Correcting problems
- Ending debugging

6.2.1 Preparing for debugging

Here we use the Editor/Debugger for debugging. Do the following to prepare for debugging:

- Open the file
- Save changes
- Be sure the file you run and any files it calls are in the directories that are on the search path.

6.2.2 Setting breakpoints

Set breakpoints *to pause* execution of the function, so we can examine where the problem might be. There are three basic types of breakpoints:

- *A standard breakpoint*, which stops at a specified line.
- *A conditional breakpoint*, which stops at a specified line and under specified conditions.
- *An error breakpoint* that stops when it produces the specified type of *warning*, *error*, *NaN*, or infinite value.

You cannot set breakpoints while MATLAB is busy, for example, running an M-file.

6.2.3 Running with breakpoints

After setting breakpoints, run the M-file from the Editor/Debugger or from the Command Window. Running the M-file results in the following:

- The prompt in the Command Window changes to

  ```
  K>>
  ```

  indicating that MATLAB is in debug mode.
- The program pauses at the *first* breakpoint. This means that line will be executed when you continue. The pause is indicated by the green arrow.
- In breakpoint, we can examine variable, step through programs, and run other calling functions.

6.2.4 Examining values

While the program is paused, we can view the value of any variable currently in the workspace. Examine values when we want to see whether a line of code has produced the expected result or not. If the result is as expected, step to the next line, and continue running. If the result is not as expected, then that line, or the previous line, contains an *error*. When we run a program, the current workspace is shown in the **Stack** field. Use `who` or `whos` to list the variables in the current workspace.

Viewing values as datatips

First, we position the cursor to the left of a variable on that line. Its current value appears. This is called a *datatip*, which is like a *tooltip* for data. If you have trouble getting the datatip to appear, click in the line and then move the cursor next to the variable.

6.2.5 Correcting and ending debugging

While debugging, we can change the value of a variable to see if the *new* value produces expected results. While the program is paused, assign a new value to the variable in the Command Window, Workspace browser, or Array Editor. Then continue running and stepping through the program.

6.2.6 Ending debugging

After identifying a problem, end the debugging session. It is best to quit *debug mode* before editing an M-file. Otherwise, you can get unexpected results when you run the file. To end debugging, select **Exit Debug Mode** from the **Debug** menu.

6.2.7 Correcting an M-file

To correct errors in an M-file,

- Quit debugging
- Do not make changes to an M-file while MATLAB is in debug mode
- Make changes to the M-file
- Save the M-file
- Clear breakpoints

- Run the M-file again to be sure it produces the expected results.

For details on debugging process, see MATLAB documentation.