

Using the Matlab Debugger (v. 1.1)  
Prepared & copyright by Dr. C. S. Tritt  
Last revised 11/29/11

A debugger is a special mode or program for executing code in a tightly controlled and interactive manor. Debuggers allow you to pause program execution at specific points or under specified conditions. While in this paused state, you can interactively investigate the values of variables and other aspects of the program's state. Debuggers are generally used for finding hard to find errors the cause of which is not immediately obvious after under careful code examination.

For more information see *Matlab > User's Guide > Desktop Tools and Development Environment > Editing and Debugging Matlab Code > Debugging Process and Features* in Matlab help.

Note – If you want to edit an .m file while debugging, it is best to first quit the debug mode and then edit and save changes to the file. If you edit a file while paused in debug mode, you can get unexpected results when you resume execution of the file, the results might not be reliable and you'll probably get very confused.

The same Matlab debugger is integrated into both the Matlab editor and command windows. You can freely switch back and forth between the two interfaces while debugging.

### Debugger Example Windows

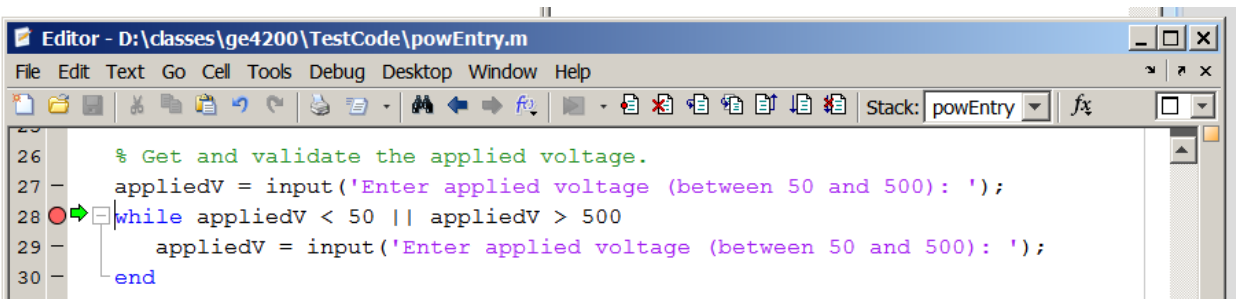


Figure 1: The editor window during a typical debugging session. An unconditional break point was set at line 28 (as indicated by the red circle) and the program was run. Execution automatically paused on line 28 (as indicated by the green arrow).

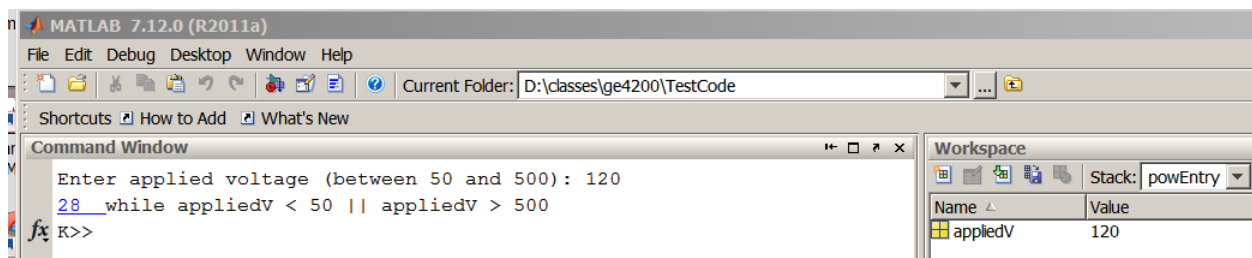


Figure 2: The command window during the debugging session described in Figure 1. The blue 28 indicates the program has stopped on line 28 (and links to it in the editor). The K>> prompt indicates the debugger is waiting for a command. Matlab expressions can also be entered at this prompt.

## Debugger Command Window Commands (bold indicates the most commonly used commands)

**dbstop** – sets a break point at the specified location or for the specified condition. See help for details.

**dbclean** – Remove breakpoint.

**dbcont** – Resume execution.

dbdown – Change local workspace context.

dbstack – List who called whom.

dbstatus – List all breakpoints.

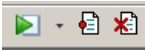
**dbstep** – Execute one or more lines.

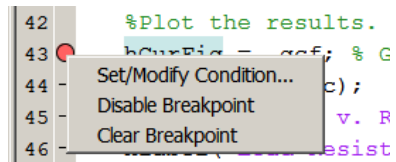
dbtype – List M-file with line numbers.

dbup – Change local workspace context.







**dbquit** – Quit debug mode.

## Setting and Clearing Breakpoints in the Editor

The  buttons are used in the editor to control debugging. The *Set/Clear Breakpoint* (red circle) button sets and clears a breakpoint on the current line. The *Clear Breakpoints* (red X) button clears all currently set breakpoints. The *Run* (green triangle) button saves the file if it has been modified and starts running the file. Execution will pause at the first applicable breakpoint. Breakpoints can also be set by left clicking in the dashes (–) to the right of the line numbers. Breakpoints can be cleared, disabled or modified by right clicking on them (as shown to the right).



## Editor Debugging Buttons

| Toolbar Button  | Debug Menu Item                               | Description   | Function Alternative       |
|---|---|---|----------------------------|
|  | <b>Run file or Run Configuration for file</b> | Commence execution of file and run until completion or until a breakpoint is encountered. The <b>Run Configurations for file</b> menu option provides a submenu. The submenu enables you to select a particular run configuration or to edit the run configurations for the MATLAB file. If you choose <b>Run file</b> , MATLAB uses the default run configuration. | None                       |
| None  | <b>Go Until Cursor</b>                        | Continue execution of file until the line where the cursor is positioned. Also available on the context menu.   | None                       |
|  | <b>Step</b>                                   | Execute the current line of the file.   | <a href="#">dbstep</a>     |
|  | <b>Step In</b>                                | Execute the current line of the file and, if the line is a call to another function, step into that function.   | <a href="#">dbstep</a> in  |
|  | <b>Continue</b>                               | Resume execution of file until completion or until another breakpoint is encountered.   | <a href="#">dbcont</a>     |
|  | <b>Step Out</b>                               | After stepping in, run the rest of the called function or subfunction, leave the called function, and pause.  | <a href="#">dbstep</a> out |
|  | <b>Exit Debug Mode</b>                        | Exit debug mode.  | <a href="#">dbquit</a>     |

## Debugging Sessions

A debugging session can be started from either the editor window (by setting one or more break points and clicking on the green triangle *Run* button) or from the command windows (by setting one or more breakpoints using the *dbstop* command and entering the name of the function or script).