Using MATLAB's debugger & profiler

presented by Clarissa Garvey
April 7, 2017

github.com/ccgarvey/matlab-debug-profile

Why care?

github.com/ccgarvey/matlab-debug-profile

Why care?

- Debugger
  - find bugs faster, easier, and without modifying code

github.com/ccgarvey/matlab-debug-profile

Why care?

- Debugger
  - find bugs faster, easier, and without modifying code
- Profiler
  - find bottlenecks, test your code

github.com/ccgarvey/matlab-debug-profile

Debugging

Debugging Workflow

1. Add a breakpoint.

github.com/ccgarvey/matlab-debug-profile

Debugging Workflow

1. Add a breakpoint.
2. Run the code.

github.com/ccgarvey/matlab-debug-profile

Debugging Workflow

1. Add a breakpoint.
2. Run the code.
3. Navigate the stack and inspect variables.

github.com/ccgarvey/matlab-debug-profile

Debugging Workflow

1. Add a breakpoint.
2. Run the code.
3. Navigate the stack and inspect variables.
4. Continue running or step through code.

github.com/ccgarvey/matlab-debug-profile

Debugging Workflow

1. Add a breakpoint.
2. Run the code.
3. Navigate the stack and inspect variables.
4. Continue running or step through code.
5. Quit debugging.

github.com/ccgarvey/matlab-debug-profile

Debugging Workflow

1. Add a breakpoint.
2. Run the code.
3. Navigate the stack and inspect variables.
4. Continue running or step through code.
5. Quit debugging.
6. Remove breakpoints if needed.

github.com/ccgarvey/matlab-debug-profile

Debugging Workflow

1. Add a breakpoint.
2. Run the code.
3. Navigate the stack and inspect variables.
4. Continue running or step through code.
5. Quit debugging.
6. Remove breakpoints if needed.
7. Rerun fixed code.

github.com/ccgarvey/matlab-debug-profile

Demo

Commands Recap

- Add breakpoints with **dbstop**.

github.com/ccgarvey/matlab-debug-profile

Commands Recap

- Add breakpoints with **dbstop**.
- Use **dbstack** to see the call stack.
- Use **dbup** and **dbdown** to navigate the stack.

github.com/ccgarvey/matlab-debug-profile

Commands Recap

- Add breakpoints with **dbstop**.
- Use **dbstack** to see the call stack.
- Use **dbup** and **dbdown** to navigate the stack.
- Inspect variables like normal.

github.com/ccgarvey/matlab-debug-profile

Commands Recap

- Add breakpoints with **dbstop**.
- Use **dbstack** to see the call stack.
- Use **dbup** and **dbdown** to navigate the stack.
- Inspect variables like normal.
- Step single lines with **dbstep**.
- Continue execution with **dbcont**.

github.com/ccgarvey/matlab-debug-profile

Commands Recap

- Add breakpoints with **dbstop**.
- Use **dbstack** to see the call stack.
- Use **dbup** and **dbdown** to navigate the stack.
- Inspect variables like normal.
- Step single lines with **dbstep**.
- Continue execution with **dbcont**.
- Quit the debugger with **dbquit**.

github.com/ccgarvey/matlab-debug-profile

Commands Recap

- Add breakpoints with **dbstop**.
- Use **dbstack** to see the call stack.
- Use **dbup** and **dbdown** to navigate the stack.
- Inspect variables like normal.
- Step single lines with **dbstep**.
- Continue execution with **dbcont**.
- Quit the debugger with **dbquit**.
- Remove breakpoints with **dbclear**.

github.com/ccgarvey/matlab-debug-profile

Profiling

Timing Features

github.com/ccgarvey/matlab-debug-profile

Timing Features

- Visualization of run time

github.com/ccgarvey/matlab-debug-profile

Timing Features

- Visualization of run time
- Navigable call stack
  - red lines take the most time

github.com/ccgarvey/matlab-debug-profile

Timing Features

- Visualization of run time
- Navigable call stack
  - red lines take the most time
- Customizable times (CPU, wall, etc.)

github.com/ccgarvey/matlab-debug-profile

Timing Demo

What is coverage?

github.com/ccgarvey/matlab-debug-profile

What is coverage?

- Used in testing

What is coverage?

- Used in testing
- Shows what methods/lines actually ran

github.com/ccgarvey/matlab-debug-profile

Coverage Features

- % coverage for files

github.com/ccgarvey/matlab-debug-profile

Coverage Features

- % coverage for files
- Individual lines of coverage

github.com/ccgarvey/matlab-debug-profile

Coverage Demo

Further Reading

Debugging: https://www.mathworks.com/help/matlab/debugging-code.html

Time Profiling: https://www.mathworks.com/help/matlab/ref/profile.html

Coverage:
https://www.mathworks.com/help/matlab/matlab_prog/determining-profiler-coverage.html

Testing: https://www.mathworks.com/help/matlab/matlab-unit-test-framework.html

Getting the profiler working on Ubuntu with Intel graphics

github.com/ccgarvey/matlab-debug-profile

Debugging Cheat Sheet

Break when error is thrown:	**dbstop if error**
Break on specific line:	**dbstop in file at line_number**
Break on function:	**dbstop in file>function**
List breakpoints:	**dbstatus**
Display call stack:	**dbstack**
Run next line:	**dbstep**
Step *in* to next line:	**dbstep in**
Run to end of function and stop:	**dbstep out**
Step up and down in callstack:	**dbup** and **dbdown**
Remove all breakpoints:	**dbclear all**
Remove one breakpoint:	**dbclear [expression used to create breakpoint]**

github.com/ccgarvey/matlab-debug-profile

Profiling Cheat Sheet

Turn profiler on, clearing history:	**profile on**
Turn profiler on, keeping history:	**profile resume**
Use total CPU time:	**profile --timer 'cpu'**
Use processor time:	**profile --timer 'processor'**
Stop profiler:	**profile off**
Stop profiler & view profile:	**profile viewer**
Check profiler state:	**profile status**

github.com/ccgarvey/matlab-debug-profile

Thank you!