



Matlab for Engineers: Debugging - warnings and errors

Violeta Monasterio
Mauricio Villarroel

May 31st, 2012

Centre for Doctoral Training in Healthcare Innovation
Institute of Biomedical Engineering
Department of Engineering Science
University of Oxford

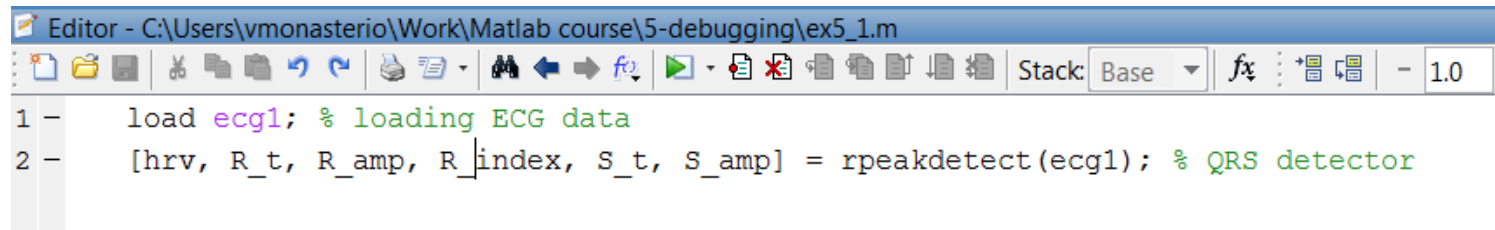
Supported by the RCUK Digital Economy Programme grant number EP/G036861/1



There are two types of errors

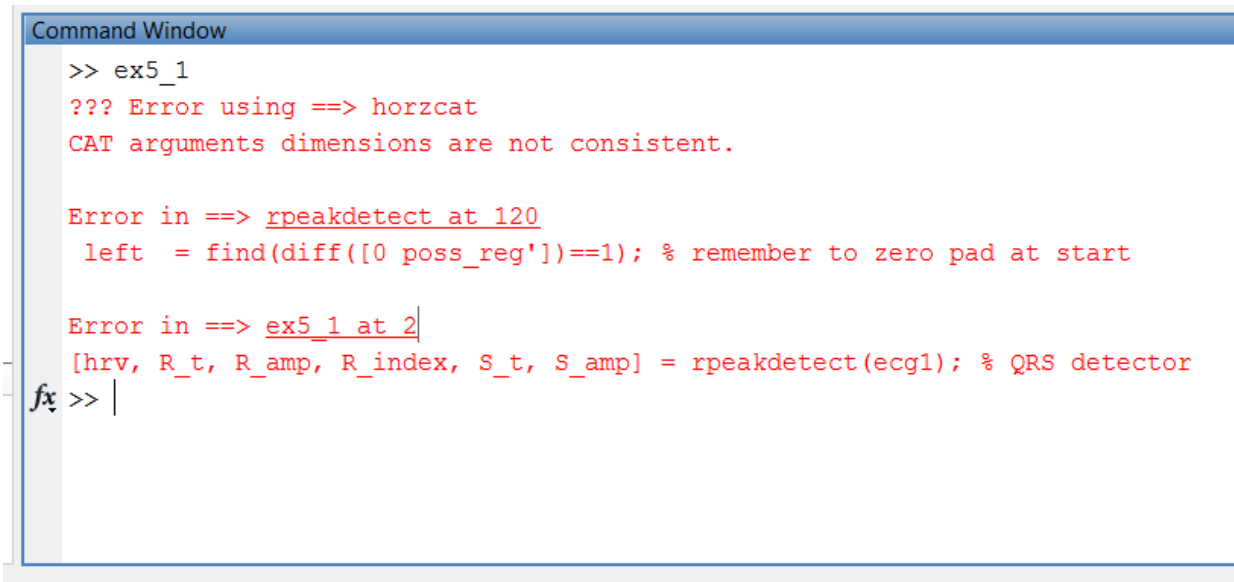
- Syntax errors: detected by matlab compiler
- Runtime errors: due to wrong logic used by the programmer:
 - Usually become apparent when one obtains erroneous or unexpected results
 - It is necessary to find the erroneous statements that caused the error: **debugging**

- Example:



The image shows a MATLAB Editor window titled "Editor - C:\Users\vmasterio\Work\Matlab course\5-debugging\ex5_1.m". The window contains two lines of MATLAB code:

```
1 - load ecg1; % loading ECG data
2 - [hrv, R_t, R_amp, R_index, S_t, S_amp] = rpeakdetect(ecg1); % QRS detector
```



The image shows the MATLAB Command Window with the following text:

```
>> ex5_1
??? Error using ==> horzcat
CAT arguments dimensions are not consistent.

Error in ==> rpeakdetect at 120
    left = find(diff([0 poss_reg'])==1); % remember to zero pad at start

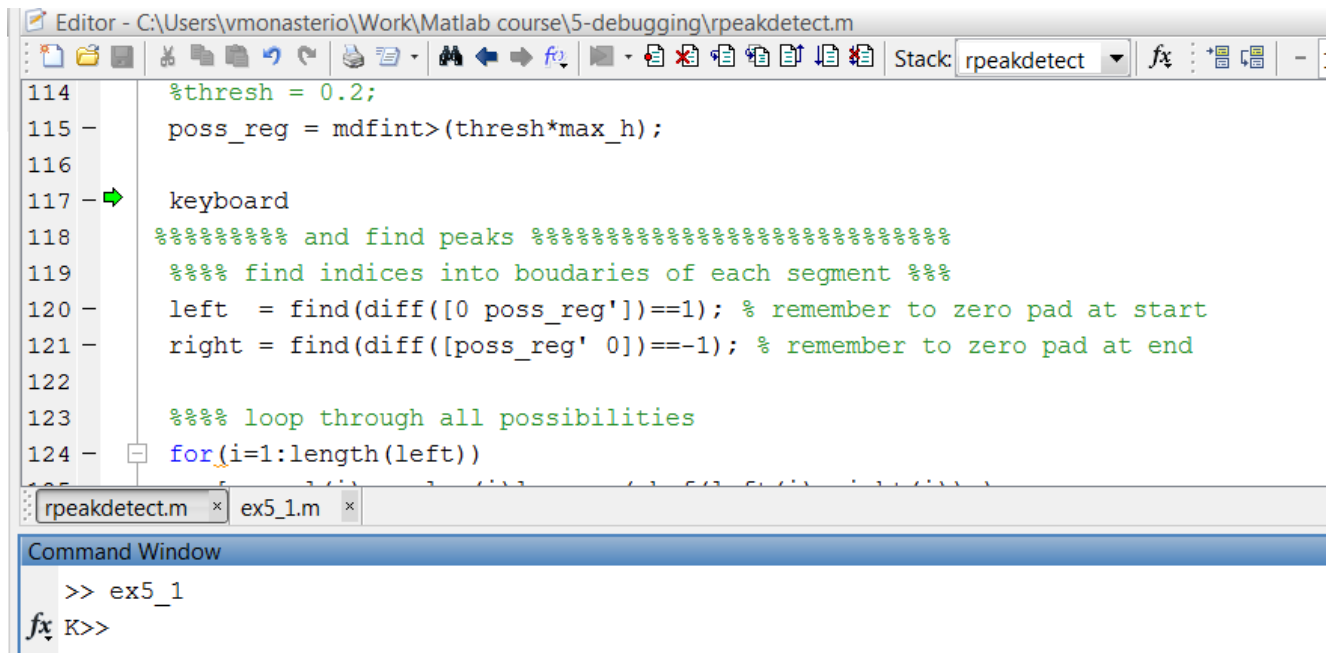
Error in ==> ex5_1 at 2
[hrv, R_t, R_amp, R_index, S_t, S_amp] = rpeakdetect(ecg1); % QRS detector
fx >> |
```

Techniques to track down errors

- Use “Code Analyzer” (mlint)
- Removing / deleting semicolons
- Executing function as a script
 - The inputs can be fixed for which the results are known
- `Keyboard` statement
- Matlab debugger

Using the `Keyboard` statement

- `Keyboard` stops the execution
- Allows the programmer to examine the local workspace and execute statements from the command prompt (`whos`, `size`, ...)



The screenshot shows the MATLAB Editor window with the file `rpeakdetect.m` open. The script contains the following code:

```
114 %thresh = 0.2;
115 poss_reg = mdfint>(thresh*max_h);
116
117 keyboard
118 %%%%%%%%% and find peaks %%%%%%%%%%%%%
119 %%% find indices into boundaries of each segment %%%
120 left = find(diff([0 poss_reg'])==1); % remember to zero pad at start
121 right = find(diff([poss_reg' 0])==1); % remember to zero pad at end
122
123 %%% loop through all possibilities
124 for(i=1:length(left))
```

The `keyboard` statement is highlighted on line 117. Below the editor, the Command Window shows the prompt `>> ex5_1` and the `K>>` prompt, indicating that the execution has stopped at the `keyboard` statement.

Using the debugger

execution control

breakpoints

workspace selection

The screenshot displays the MATLAB debugger environment. The main editor window shows a script named `rpeakdetect.m` with several lines of code. A red circle highlights the 'execution control' toolbar, which includes buttons for stepping through code (step, step in, step out, run to cursor, etc.). A red arrow points to the 'workspace selection' dropdown menu, which is currently set to 'rpeakdetect'. The workspace window on the right shows a list of variables and their values. The command window at the bottom shows the execution of the `ex5_1` script, with the current line of execution highlighted at line 120.

```
114 %thresh = 0.2;
115 poss_reg = md fint>(thresh*max_h);
116
117 % and find peaks
118 % find indices into boudaries of each segment
119 left = find(diff([0 poss_reg])==1); % remember to zero pad at start
120 right = find(diff([poss_reg' 0])==1); % remember to zero pad at end
121
122 % loop through all possibilities
123 for(i=1:length(left))
124     [maxval(i) maxloc(i)] = max( bpf(left(i):right(i)) );
125     [minval(i) minloc(i)] = min( bpf(left(i):right(i)) );
126     maxloc(i) = maxloc(i)-1+left(i); % add offset of present location
127     minloc(i) = minloc(i)-1+left(i); % add offset of present location
128 end
129
130 R_index = maxloc;
131 R_t = t(maxloc);
132 R_amp = maxval;
133 S_amp = minval; % Assuming the S-wave is the lowest
134 % amp in the given window
```

Workspace

Name	Value
a	1
b	86250
bpf	<1x86250 doub...
d	[1,1,1,1,1,1]
data	<1x86250 doub...
delay	4
dff	<1x86249 doub...
len	86249
max_h	3.9883
mdfint	<1x86246 doub...
poss_reg	<1x86246 logic...
samp_freq	256
sqr	<1x86249 doub...
t	<1x86250 doub...
testmode	0
thresh	0.2000
tt	<1x86272 doub...
x	<1x86250 doub...

Command Window

```
>> ex5_1
120 left = find(diff([0 poss_reg])==1); % remember to zero pad at start
K>>
```

Debugging from the command line

Command	Description
<code>dbstop</code>	set breakpoint
<code>dbclear</code>	clear breakpoint
<code>dbclear all</code>	clear all breakpoints
<code>dbstop if</code>	stop on warning, error or NaN/Inf
<code>error</code>	NaN/Inf generation
<code>dbstep</code>	single step execution
<code>dbstep in</code>	step into a function
<code>dbstep nlines</code>	execute one or more lines
<code>dbcont</code>	continue execution
<code>dbquit</code>	quit debugging
<code>dbstack</code>	list function call stack
<code>dbstatus</code>	list all breakpoints
<code>dbtype</code>	list M-file with line numbers
<code>dbdown / dbup</code>	change local workspace down / up

Preventing common errors

- Avoid dividing by zero: $1/x \rightarrow 1/(x + \text{eps})$
- Default `else` for `if-elseif`,
Default `otherwise` for `switch-case`

```
if condition1,  
    statement1;  
elseif condition2  
    statement2;  
...  
elseif conditionN,  
    statementN;  
else default_statement  
end
```


Preventing common errors

- Check inputs: number, type, size
 - assume default values where possible
 - if a required input is missing: throw error and exit (assert)

```
function write2file(varargin)
min_inputs = 2;
assert(nargin >= min_inputs, 'You must call function...
    %s with at least %d inputs', mfilename, min_inputs)

infile = varargin{1};
assert(ischar(infile), 'First argument must be a filename.')

fid = fopen(infile, 'w');
assert(fid > 0, 'Cannot open file %s for writing', infile)

fwrite(fid, varargin{2});
```

Handling errors

- Try / catch block

```
try
    [hrv, R_t] = rpeakdetect(ecg1); % QRS detector
catch err
    if(strcmp(err.identifier,...
        'MATLAB:catenate:dimensionMismatch'))
        try % try again with transposed input
            [hrv, R_t] = rpeakdetect(ecg1);
        catch
            rethrow(err) % rethrow original error
        end
    end
end
end
```

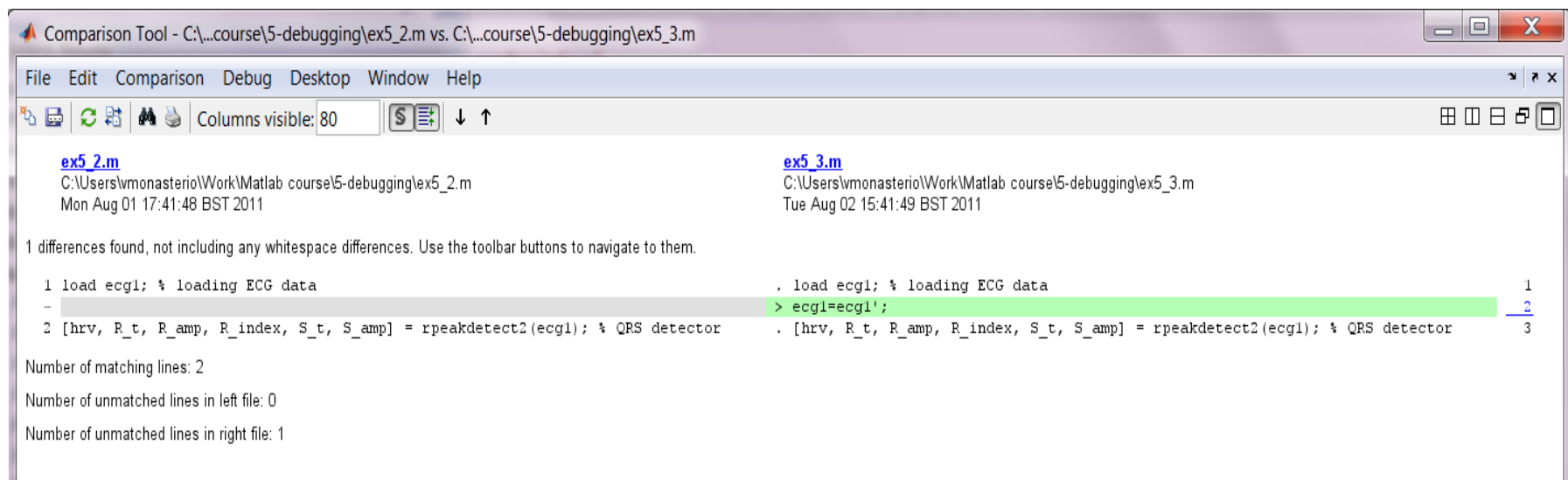
Keeping things tidy (onCleanup)

- Leave your program environment in a clean state:
 - close any open files
 - restore the MATLAB path
 - set the working folder back to its default
 - make sure global variables are in the correct state

```
function openFileSafely(fileName)
fid = fopen(fileName, 'r');
c = onCleanup(@() fclose(fid));
s = fread(fid);
    .
    .
    .
end
```

Other tools

- In the editor -> Tools -> Compare against
– compares M-files, MAT-files and directories



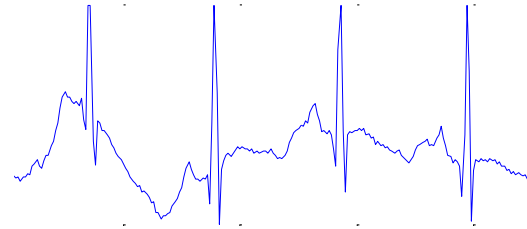
The screenshot shows the MATLAB Comparison Tool window. The title bar reads "Comparison Tool - C:\...course\5-debugging\ex5_2.m vs. C:\...course\5-debugging\ex5_3.m". The menu bar includes File, Edit, Comparison, Debug, Desktop, Window, and Help. The toolbar shows icons for file operations and a "Columns visible: 80" indicator. The main area displays two files side-by-side: **ex5_2.m** (left) and **ex5_3.m** (right). Both files are located at C:\Users\lmonasterio\Work\Matlab course\5-debugging\ and were last modified on August 1, 2011. A message states "1 differences found, not including any whitespace differences. Use the toolbar buttons to navigate to them." The code for both files is shown, with line numbers on the right. In the right file, line 2 is highlighted in green. Summary statistics at the bottom indicate 2 matching lines, 0 unmatched lines in the left file, and 1 unmatched line in the right file.

```
Comparison Tool - C:\...course\5-debugging\ex5_2.m vs. C:\...course\5-debugging\ex5_3.m
File Edit Comparison Debug Desktop Window Help
Columns visible: 80
ex5_2.m C:\Users\lmonasterio\Work\Matlab course\5-debugging\ex5_2.m Mon Aug 01 17:41:48 BST 2011
ex5_3.m C:\Users\lmonasterio\Work\Matlab course\5-debugging\ex5_3.m Tue Aug 02 15:41:49 BST 2011
1 differences found, not including any whitespace differences. Use the toolbar buttons to navigate to them.
1 load ecg1; % loading ECG data
-
2 [hrv, R_t, R_amp, R_index, S_t, S_amp] = rpeakdetect2(ecg1); % QRS detector
. load ecg1; % loading ECG data 1
> ecg1=ecg1'; 2
. [hrv, R_t, R_amp, R_index, S_t, S_amp] = rpeakdetect2(ecg1); % QRS detector 3
Number of matching lines: 2
Number of unmatched lines in left file: 0
Number of unmatched lines in right file: 1
```

Practice: QRS detector

(practice_5.m)

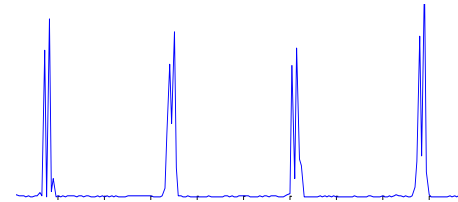
1. Low-pass filter



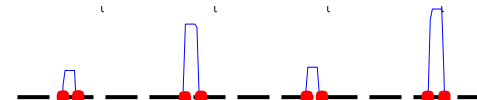
2. Derivation



3. Squaring



4. Integration



5. Thresholding

