

Advanced Topics in MATLAB®

Abishek Gopal

agopal@tamu.edu

Fall 2019 HPRC Short Course

Nov 15, 2019



Disadvantages of Matlab scripts

- No private variables, all variables displayed in the workspace
- Running two different scripts with common variable names could lead to potential problems
- Does not accept input arguments
- Does not return output arguments

Functions

A **function** is a group of statements that together perform a task.

- Encapsulates internal data and procedure
- Accepts input arguments and returns outputs
- Can be called from scripts or the Command Window

```
%Matlab
function [out]=myabs (number)
    if number > 0
        out = number
    else
        out = -number
    end
end
```

Functions

- In MATLAB, functions are defined in separate files.
- The name of the file and of the function should be the same.
- Files can include multiple local functions or nested functions

```
% myabs.m
function[out]=myabs(number)
    if number > 0
        out = number
    else
        out = -number
    end
end
```

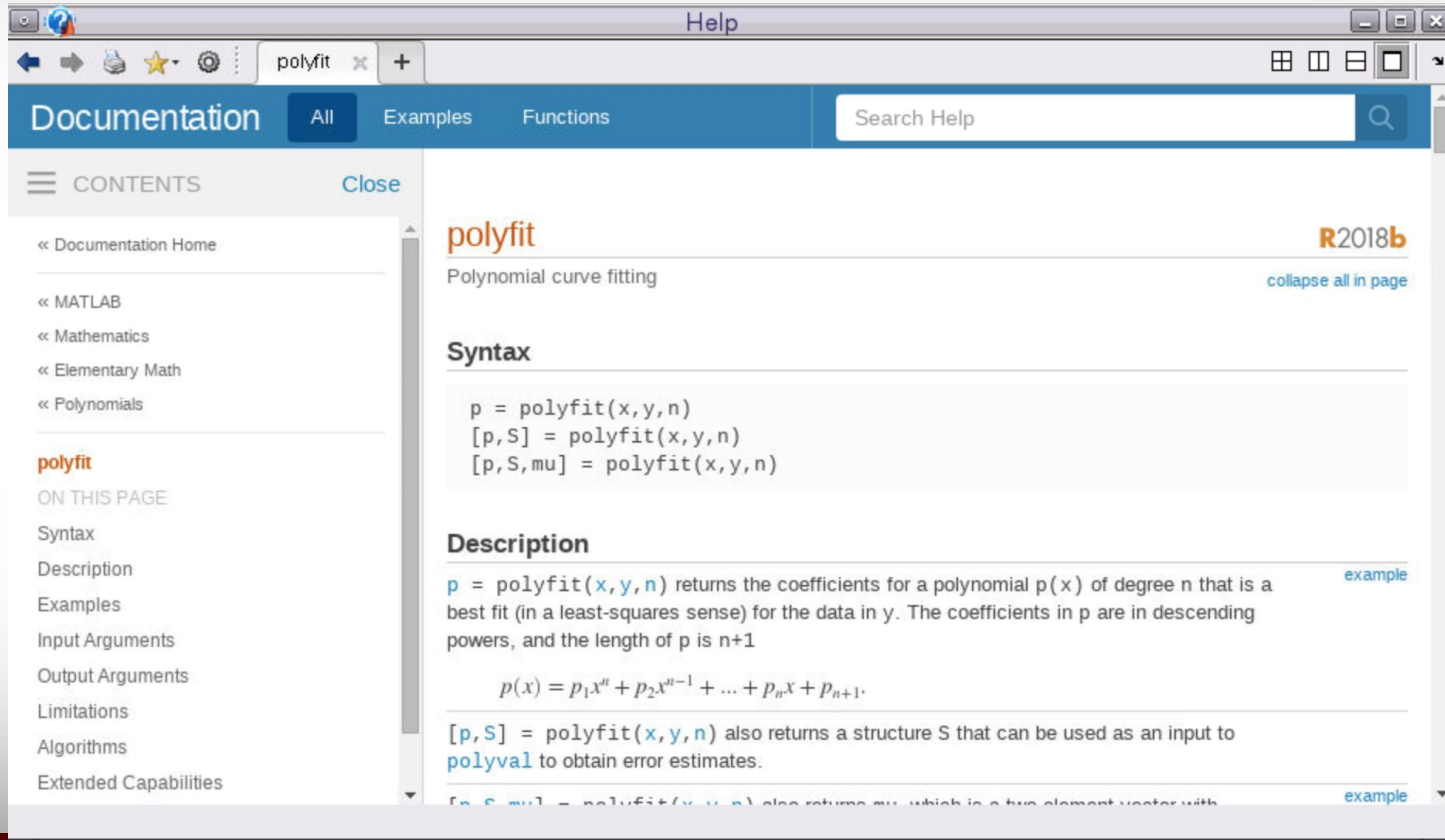
Getting help in MATLAB

The screenshot shows the MATLAB R2018b desktop environment. The Command Window is active, displaying the help text for the `eig` function. The text is as follows:

```
New to MATLAB? See resources for Getting Started.  
... Starting Matlab through the portal, setting up environment  
>> help eig  
eig Eigenvalues and eigenvectors.  
E = eig(A) produces a column vector E containing the eigenvalues of  
a square matrix A.  
  
[V,D] = eig(A) produces a diagonal matrix D of eigenvalues and  
a full matrix V whose columns are the corresponding eigenvectors  
so that A*V = V*D.  
  
[V,D,W] = eig(A) also produces a full matrix W whose columns are the  
corresponding left eigenvectors so that W'*A = D*W'.  
  
[...] = eig(A,'nobalance') performs the computation with balancing  
disabled, which sometimes gives more accurate results for certain  
problems with unusual scaling. If A is symmetric, eig(A,'nobalance')  
is ignored since A is already balanced.  
  
[...] = eig(A,'balance') is the same as eig(A).  
  
E = eig(A,B) produces a column vector E containing the generalized  
eigenvalues of square matrices A and B.  
  
[V,D] = eig(A,B) produces a diagonal matrix D of generalized  
eigenvalues and a full matrix V whose columns are the corresponding  
eigenvectors so that A*V = B*V*D
```

The interface also shows the 'Current Folder' on the left with files like 'make_macros.mk' and 'OpenFOAM-7-foss-2018b.eb'. The 'Workspace' on the right is empty.

Getting help in MATLAB



The screenshot shows the MATLAB Help browser window. The browser address bar shows the URL `polyfit`. The page title is "Help". The main content area displays the documentation for the `polyfit` function, which is categorized under "Polynomials" in the left-hand navigation menu. The documentation includes the following sections:

- polyfit** (R2018b): Polynomial curve fitting. A link to "collapse all in page" is visible.
- Syntax**:

```
p = polyfit(x,y,n)
[p,S] = polyfit(x,y,n)
[p,S,mu] = polyfit(x,y,n)
```
- Description**: `p = polyfit(x,y,n)` returns the coefficients for a polynomial $p(x)$ of degree n that is a best fit (in a least-squares sense) for the data in y . The coefficients in p are in descending powers, and the length of p is $n+1$. An "example" link is provided.
- Equation**:
$$p(x) = p_1x^n + p_2x^{n-1} + \dots + p_nx + p_{n+1}.$$
- Additional Information**: `[p,S] = polyfit(x,y,n)` also returns a structure S that can be used as an input to `polyval` to obtain error estimates. An "example" link is provided.
- Final Line**: `[p,S,mu] = polyfit(x,y,n)` also returns μ , which is a two-element vector with... An "example" link is provided.

MATLAB for Linear Algebra



Row and column vectors

```
>> x = [1 2 3]           y = [1 2 3]'  
  
x = 1    2    3         y = 1  
                          2  
                          3  
  
>> size(x)  
  
ans = 1    3           >> size(y)  
                          ans = 3    1
```


Vector multiplication

```
>> x*y  
ans = 14
```

```
>> x.*y'  
ans = 1 4 9
```

```
>> x.*y  
ans =  
1      2      3  
2      4      6  
3      6      9
```

Matrix-vector multiplication

Try:

```
>> x = [1 2 3]
```

```
>> A = [1 2 3; 4 5 6 ; 7 8 9]
```

```
>> A*x
```

Now try:

```
>> A*x'
```

Matrix-matrix multiplication

```
>> A = [1 2; 3 4; 5 6];    % 3x2 matrix
>> B = [-1 0 1; 1 2 3];    % 2x3 matrix
>> A*B
ans =
     1     4     7
     1     8    15
     1    12    23
```

Solution of linear system

$$A x = b$$

$$x = A^{-1} b$$

```
>> b = [3 5 7]
```

```
>> A = magic(3);
```

```
>> x = inv(A) * b
```

```
>> x = A\b
```

Sparse matrices

```
>> A = sparse([0 2 0 1 0; 1 0 -1 0 0; 0 0  
0 3 1; -2 0 0 0 2; 0 0 2 2 0]);  
>> b = sparse([1; 0; 8; 4; 2]);  
>> x = A\b
```

```
x =
```

(1,1)	-1.5000
(2,1)	-0.7500
(3,1)	-1.5000
(4,1)	2.5000
(5,1)	0.5000

Eigenvalues/eigenvectors

$$A x = d x$$

```
>> A = magic(4)
```

```
>> [V,D]=eig(A)
```

```
V = -0.5000    -0.8236     0.3764    -0.2236  
      -0.5000     0.4236     0.0236    -0.6708  
      -0.5000     0.0236     0.4236     0.6708  
      -0.5000     0.3764    -0.8236     0.2236
```

```
D =34.0000         0         0         0  
         0     8.9443         0         0  
         0         0    -8.9443         0  
         0         0         0    -0.0000
```

Vector/Matrix functions

Matlab Function	Description
dot(a,b)	Dot product of two matrices/vectors
cross(a,b)	Cross product
norm(a)	Vector/Matrix norm
trace	Sum of diagonal elements
det	Determinant of matrix
cond	Matrix condition number
svd	Singular value decomposition

Polynomials in Matlab

Polynomial Functions

Matlab Function	Description
<code>polyval(p,x)</code>	Evaluate polynomial whose coefficients are by array <code>p</code> , at scalar value <code>x</code>
<code>roots(p)</code>	Evaluate roots of polynomial <code>p</code>
<code>polyint(p)</code>	Polynomial integration
<code>polyder(p)</code>	Polynomial derivative
<code>polyfit</code>	Determine least-squares polynomial fit of data
<code>conv()/deconv()</code>	Polynomial multiplication/division

Polynomial evaluation

$$p(x) = x^3 - 6x^2 + 11x - 6 = 0$$

```
>> p = [1 -6 11 -6];  
>> polyval(p, 0)      % evaluate p(x) @ x=0  
ans = -6  
  
>> x=[0:0.01:4];  
>> y=polyval(p, x);  
>> plot(x,y)
```

Polynomial roots

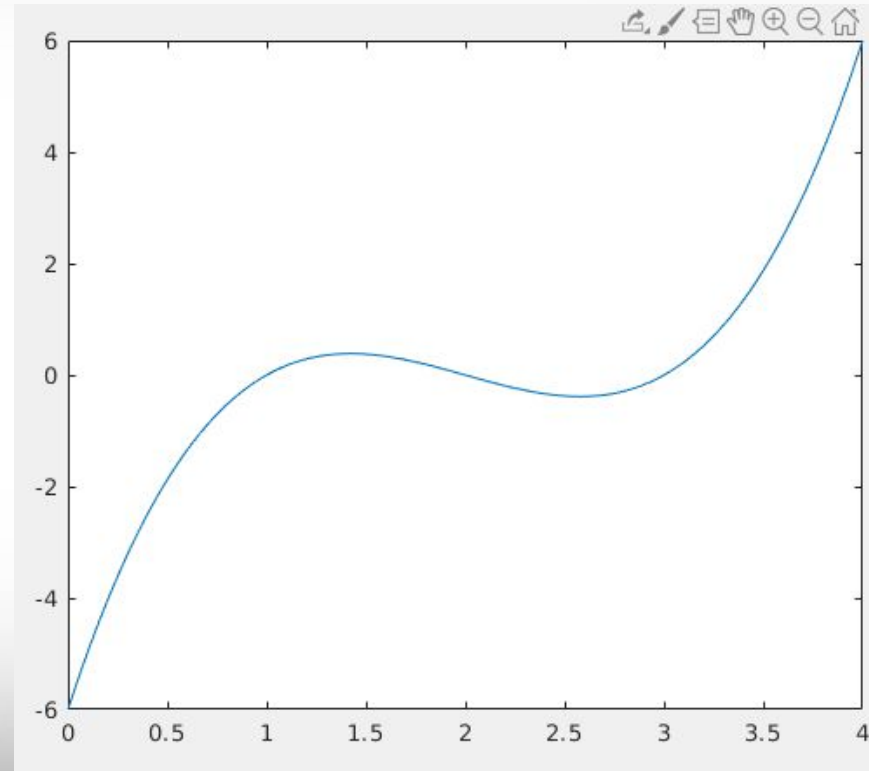
```
>> roots(p)
```

```
ans =
```

```
3.0000
```

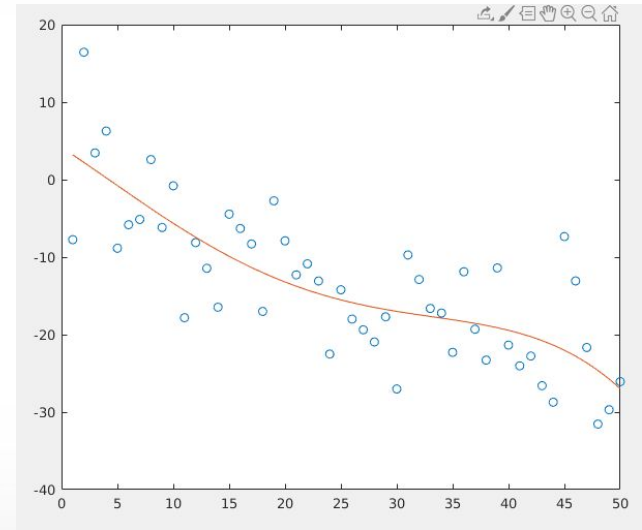
```
2.0000
```

```
1.0000
```



Polynomial fit

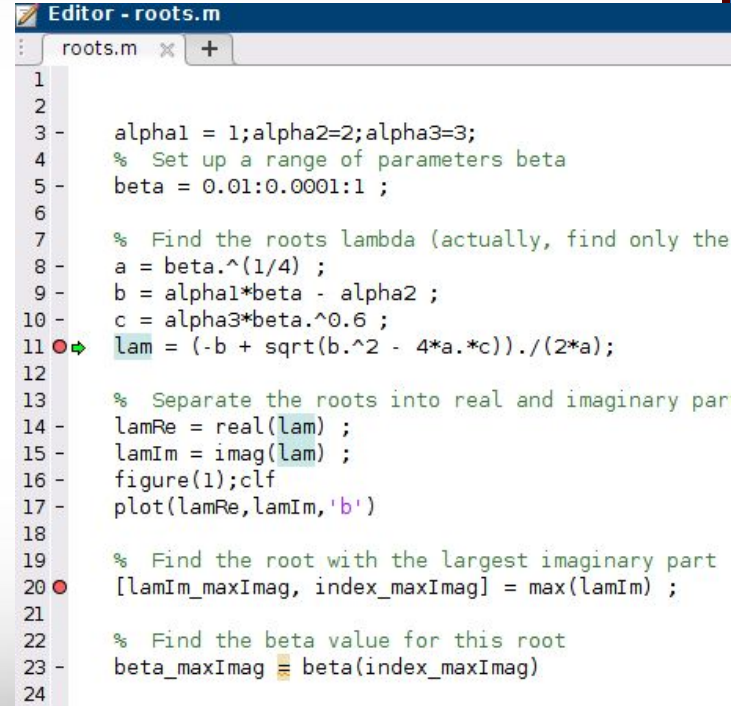
```
>> x = 1:50;  
>> y = -0.5*x + 6*randn(1,50);  
>> p = polyfit(x,y,4);  
>> f = polyval(p,x);  
>> plot(x,y,'o')  
>> hold on;  
>> plot(x,f,'-')
```



Debugging in Matlab








Using breakpoints

1. Click on the dash (-) in the breakpoint alley next to line number or press F12
2. Run script or call function
3. Control (green arrow) pauses at breakpoint
4. Investigate values and continue running script with optionally more breakpoints



```
Editor - roots.m
roots.m x +
1
2
3 - alpha1 = 1;alpha2=2;alpha3=3;
4 % Set up a range of parameters beta
5 - beta = 0.01:0.0001:1 ;
6
7 % Find the roots lambda (actually, find only the
8 - a = beta.^(1/4) ;
9 - b = alpha1*beta - alpha2 ;
10 - c = alpha3*beta.^0.6 ;
11 ● → lam = (-b + sqrt(b.^2 - 4*a.*c))./(2*a);
12
13 % Separate the roots into real and imaginary part
14 - lamRe = real(lam) ;
15 - lamIm = imag(lam) ;
16 - figure(1);clf
17 - plot(lamRe,lamIm,'b')
18
19 % Find the root with the largest imaginary part
20 ● [lamIm_maxImag, index_maxImag] = max(lamIm) ;
21
22 % Find the beta value for this root
23 - beta_maxImag = beta(index_maxImag)
24
```

Debugging toolbar

Toolbar Button	
	Run to Cursor
	Step
	Step In
	Continue
	Step Out
	Pause
	Quit Debugging

Matlab File I/O

Matlab format files .mat

```
>> save('data.mat')
```

```
>> save('data.mat', 'var1', 'var2')
```

```
>> load('data.mat')
```

```
>> whos('-file', 'data.mat')
```

Readmatrix/Writematrix

Supported Formats

- .txt, .dat, or .csv for delimited text files
- .xls, .xlsb, .xslm, .xlsx, .xltm, .xltx, or .ods for spreadsheet files

```
>> M = magic(4)
```

```
M =
```

```
16  2  3 13
 5 11 10  8
 9  7  6 12
 4 14 15  1
```

```
>> writematrix(M, 'mat_M.txt', 'Delimiter',  
'tab')
```

```
>> A = readmatrix('mat_M.txt')
```

Low-level text file I/O

Matlab Function	Description
<code>fopen</code>	Open file, or obtain information about open files
<code>fprintf</code>	Read data from binary file
<code>fscanf</code>	Write data to binary file
<code>fgets/fgetl</code>	Read line from file, keeping/removing newline characters
<code>feof</code>	Check for End of File
<code>fclose</code>	Close file

Text file write - fprintf

```
>> fileID = fopen('test.txt','w')
```

```
>> fprintf(fileID,'Hello world\n')
```

```
>> fprintf(fileID,'%d %f %s\n',a(1),b(1),c{1})
```

```
>> fprintf(fileID,'%d %f %s\n',a(2),b(2),c{2})
```

```
>> fclose(fileID);
```

```
>> a=[1:3]
```

```
>> b = [0.5:0.25:1.0]
```

```
>> c ={'One','Two','Three'}
```

Text file write - fprintf

```
>> fileID = fopen('test.txt','r')
```

```
>> fscanf(fileID,'%d %f %s\n')
```

```
>> fclose(fileID);
```

Binary file I/O

Matlab Function	Description
fopen	Open file, or obtain information about open files
fread	Read data from binary file
fwrite	Write data to binary file
fseek	Move to position in file
feof	Check for End of File
fclose	Close file

Binary file write

```
>> fileID = fopen('myfile.dat', 'w');
```

```
>> fwrite(fileID, a, 'int');
```

```
>> fwrite(fileID, b, 'double');
```

```
>> fclose(fileID)
```

```
>> dir('myfile.dat').bytes
```

```
ans = 36
```

```
>> a = [1 2 3]
```

```
>> b = [0.5 0.75 1.0]
```

Binary file read

```
>> fileID = fopen('myfile.dat','r');
```

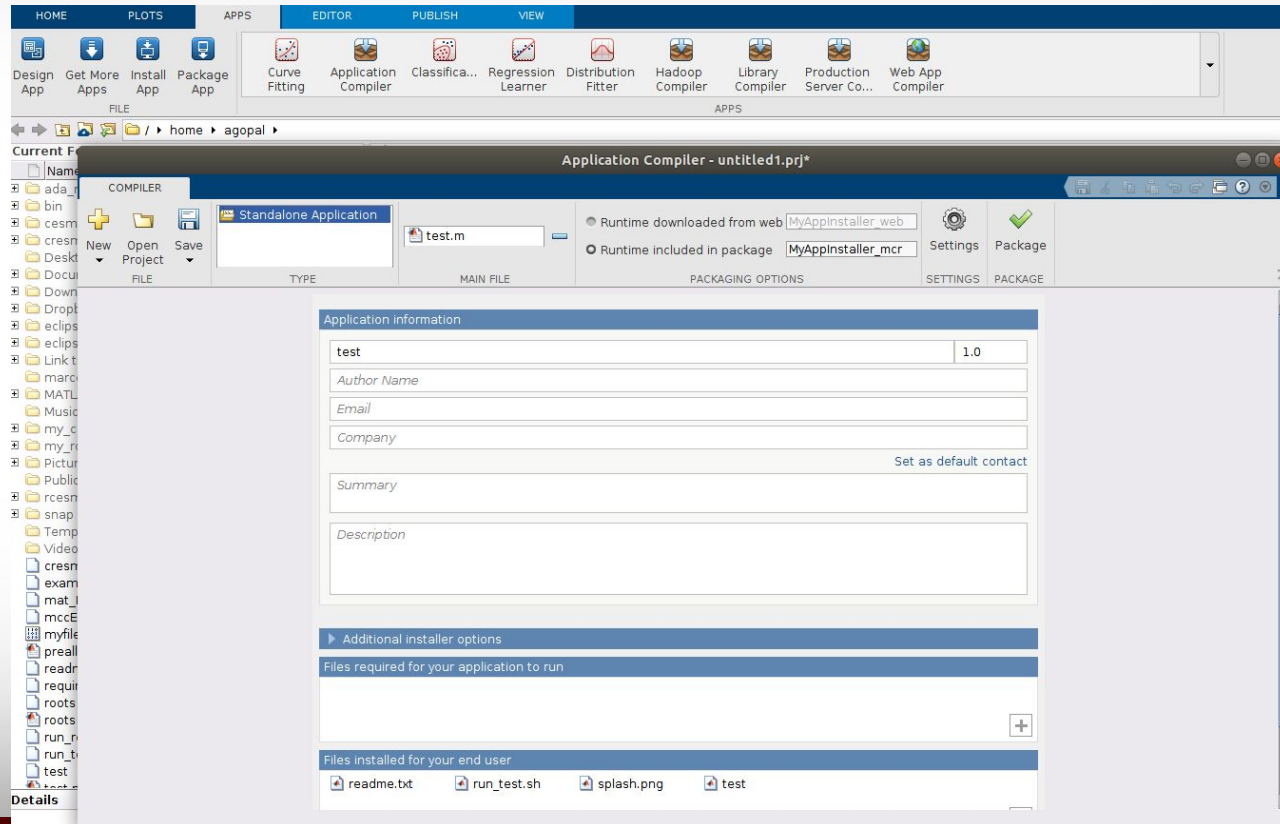
```
>> fread(fileID)      % try it
```

```
>> fread(fileID,3,'int')
```

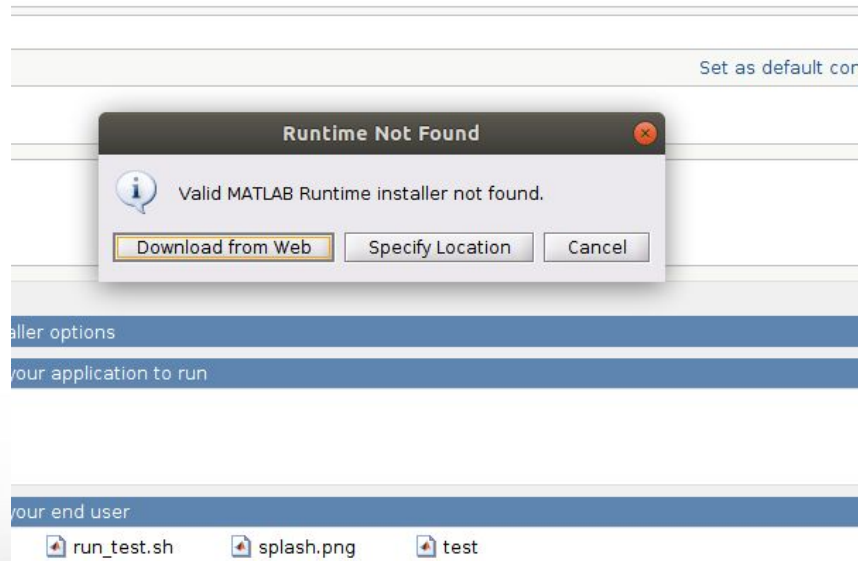
```
>> fread(fileID,3,'double')
```


Matlab compiler

Create standalone matlab programs



Matlab runtime



From the command line - mcc

```
>> mcc -m script.m
```

On linux shell

```
$LD_LIBRARY_PATH=/home/agopal/MATLAB/MATLAB_Runtime/v97/bin/glnxa64:/home/agopal/MATLAB/MATLAB_Runtime/v97/sys/os/glnxa64:LD_LIBRARY_PATH
```

```
$export LD_LIBRARY_PATH
```

```
$./script
```

Appendix