Getting started with Matlab

Dr. Guanyi Yang

St. Lawrence University

January 20, 2019

First thing first

Organize your work

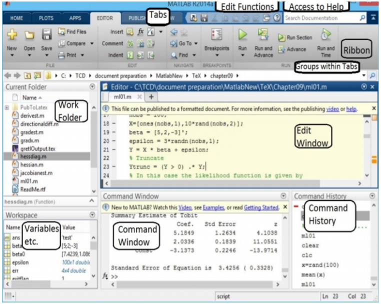
- set up a new directory for each project
- name each .m file in a recognizable and organized way
- write clear comments for every line of the work for each .m file
- keep a journal within the directory and document everything you have done, and how to operate each .m file

Matlab desktop

- **Command window**: enter and execute Matlab commands and display any output
- **Editor window**: edit Matlab files, which contain scripts of codes for later execution
- **Workspace or variables window**: all the objects created during the current session are listed. Double clicking on an item in this window opens an Editor where you may examine it or edit it.
- **Current or work folder window**: this is your project directory

Matlab desktop

- **Command window**: enter and execute Matlab commands and display any output
- **Editor window**: edit Matlab files, which contain scripts of codes for later execution
- **Workspace or variables window**: all the objects created during the current session are listed. Double clicking on an item in this window opens an Editor where you may examine it or edit it.
- **Current or work folder window**: this is your project directory



This file can be published to a formatted document. For more information, see the publishing video or help.

```
    nobs = 100;
    X=[ones(nobs,1),10*rand(nobs,2)];
    beta = [5,2,-3]';
    epsilon = 3*randn(nobs,1);
    Y = X * beta + epsilon:
    % Truncate
    Ytrunc = (Y > 0) .* Y;
    % In this case the likelihood function is given by
```

Name	Value
ans	'test'
beta	[5;2;-3]
beta0	[7.4239,1.08
epsilon	100x1 double
err	4x4 double
exitflag	1

New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
Summary Estimate of Tobit
                  Coef.    Std Error          z
                 5.1849       1.2634     4.1038
                 2.0336       0.1839    11.0551
       const    -3.1373       0.2246   -13.9714

Standard Error of Equation is  3.4256 ( 0.3328)
>>
```

```
ml01
clear
clc
x=rand(100)
mean(x)
ml01
```

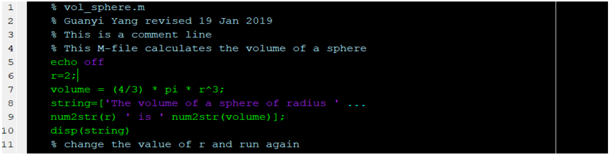
Matlab operations

The simplest use of the command window is a calculator.

- $+$,-,*,$/$,$\wedge$ all have their usual meanings
- ; at the end of a command suppresses output, but any command is going to be executed
- If a statement will not fit on one line, you may type ... at the end of the line to be continued.

Matlab Editor window

Editor saves all your scripts into an *.m* file.



```
% vol_sphere.m
% Guanyi Yang revised 19 Jan 2019
% This is a comment line
% This M-file calculates the volume of a sphere
echo off
r=2;
volume = (4/3) * pi * r^3;
string=['The volume of a sphere of radius ' ...
num2str(r) ' is ' num2str(volume)];
disp(string)
% change the value of r and run again
```

Always put **clear; close all; clc;** at the beginning of a program (unless you need to use previous memory).

% in a line means it is a comment that matlab will skip this line while executing.

You can select **save** and **save as** and name the file as **vol_sphere**.

It will be saved in your default directory.

In the Command Window, enter **vol_sphere**, matlab will execute it as if it were a matlab instruction.

Assign matrices

Assign values to a (1 by 4) matrix (row vector):

$$A = [1\ 2\ 3\ 4]$$

Assign values to a (4 by 1) (column) vector:

$$B = [1;\ 2;\ 3;\ 4]$$

Assign value to a (2 by 3) matrix:

$$Re\_1988 = [1,\ 2,\ 3;\ 4,\ 5,\ 6]$$

Assign empty array:

$$Ca = []$$

Matrix operations

$$X = [1\ 2;\ 3\ 4];$$

$$Y = [3\ 7;\ 5\ 4];$$

Addition and subtraction of two matrices - same dimensions:$X + Y$, and $X - Y$.
For multiplication, the matrix dimension must agree i.e. (m by n)*(n by l)
You can try $X * A$, and check the error message.
Then try $X * Re\_1988$.

Matrix operations

Matrix Multiplication

$$A = \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \end{bmatrix}$$

$$B = \begin{bmatrix} b11 & b12 \\ b21 & b22 \\ b31 & b32 \end{bmatrix}$$

$A * B$ refers to:

$$= \begin{bmatrix} a11 * b11 + a12 * b21 + a13 * b31 & a11 * b12 + a12 * b22 + a13 * b32 \\ a21 * b11 + a22 * b21 + a23 * b31 & a21 * b12 + a22 * b22 + a23 * b32 \end{bmatrix}$$

Matrix operations

There is no divide in matrix-to-matrix operations. However, there is a $\backslash$ sign.
It is used to solve equations $Ax = B$, where:

$$A = [1\ 2;\ 3\ 4]$$

$$x = [x1;\ x2]$$

$$B = [8;\ 10]$$

Try: $x = A \backslash B$.
It is the same thing as solving two equations with two variables:

$$1 * x1 + 2 * x2 = 8$$

$$3 * x1 + 4 * x2 = 10$$

Matrix operations

The $+, -$ operators between two *same dimension* matrices do standard operations element by element. For multiplication and division, however, you need **dot** operation:

Element by element multiplication and division

$$A = \begin{bmatrix} a11 & a12 \\ a21 & a22 \end{bmatrix}$$

$$B = \begin{bmatrix} b11 & b12 \\ b21 & b22 \end{bmatrix}$$

And dot operation as: $A. * B$ gives you:

$$= \begin{bmatrix} a11 * b11 & a12 * b12 \\ a21 * b21 & a22 * b22 \end{bmatrix}$$

And $A./B$ (notice, it is $/$ sign rather than the $\backslash$ in the inverse matrix operation before) gives you:

$$= \begin{bmatrix} a11/b11 & a12/b12 \\ a21/b21 & a22/b22 \end{bmatrix}$$

Miscellaneous operations

Mixed scalar and matrix

Adding a scalar to or multiplying a scalar by a matrix does that operation on each element of the matrix. Let $a = 2$;

Try:

$$A + a$$

$$A - a$$

$$A * a$$

$$A/a$$

Miscellaneous operations

Exponents

There are three types of exponents:

- Raise the matrix to some power: A^{power}.This only applies to square matrices, same as $A * A * A * \dots$ **Try**:

$$A \wedge a$$

- Raise each element of a matrix to a power: $A. \wedge power$ *(dot operation)* **Try**:

$$A. \wedge a$$

- Raise the elements to specific powers: **Try**:

$$Z = [1\ 2;\ 2\ 1]$$

$$A. \wedge Z$$

Sequences

`[first:increment:last]` is a row vector whose elements are a sequence with first element `first`, second element `first+ increment` and continues while the new entry is less than `last`.
Try:

$$[1:2:9]$$

$$[2:2:9]$$

Or if only two numbers are specified, the default increment is 1:

$$[1:4]$$

Another way of creating an equal spaced sequence, without specifying increment, but specifying number of elements (length), we use command `linspace`:

$$z = linspace(1, 10, 150)$$

We usually declare sequences as some *grid* base for us to find complicated numerical values.

Special matrices

- Create an *n x m* matrix with all elements equal to one:

$$x = ones(4, 2)$$

- Create an *n x m* matrix with all elements equal to zero:

$$y = zeros(3, 5)$$

- If you want to know the size of a multi-dimension matrix:

$$size(x)$$

- If you want to know the length of a one dimension array: **Try**:

$$z = linspace(1, 10, 150);$$

$$length(z)$$

Many more command and operations

Google is always a great way to help you find the easiest command to work on certain operations. Learn as you go.

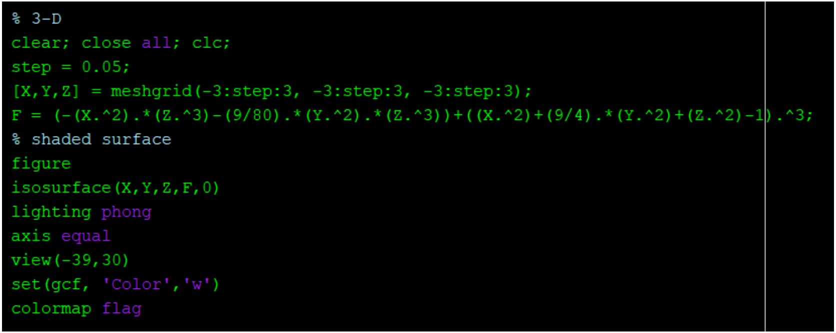
Try this

Open a new script and save it in your directory under file name *try1_2D.m*

```
clear; close all; clc;
n = linspace(-pi,pi, 350);
x = n .* sin( pi * sin(n)./n);
y = -abs(n) .* cos( pi * sin(n)./n);
figure
plot(x,y,’.r’);
fill(x, y, ’r’);
set(gcf, ’Position’, get(0,’Screensize’));
title(’Happy ........  day’, ’FontSize’, 26);
```

Try this

Open a new script and save it in your directory under file name *try2_3D.m*



```
% 3-D
clear; close all; clc;
step = 0.05;
[X,Y,Z] = meshgrid(-3:step:3, -3:step:3, -3:step:3);
F = (-(X.^2).*(Z.^3)-(9/80).*(Y.^2).*(Z.^3))+((X.^2)+(9/4).*(Y.^2)+(Z.^2)-1).^3;
% shaded surface
figure
isosurface(X,Y,Z,F,0)
lighting phong
axis equal
view(-39,30)
set(gcf, 'Color','w')
colormap flag
```

If logic

Version 1:

```
if (logical-expression)
  statements
end
```

If the program satisfies the logic, execute statement, otherwise skip the loop.

Version 2:

```
if (logical-expression)
  statements-1
else
  statements-2
end
```

If the program satisfies the logic, execute statement 1, else execute statement 2.

Version 3:

```
if (logical-expression-1)
  statements-1
elseif (logical-expression-2)
  statements-2
end
```

If the program satisfies the logic-1, execute statement 1, else if it satisfies logic-2, execute statement 2, and if it satisfies none of logic-1 and logic-2, it skips the if loop and moves on.

For loop

Given an array grid space, you want to do some statement for each position of the grid space:

```
for (index looping over the grid space)
  statements
end
```

`"index looping over the grid space"` is the same as when we were learning sequence. An example of common way of doing it

```
i = starting position:increment:ending position
```

```
i = 1:1:20
```

I name the index i, and execute the loop starting from $i = 1$ with increment 1 until it reaches $i = 20$.

For loop - example

Write the following program in a new script.

```
 x = linspace(1,2,10); % create a sequence starting from 1
and end at 2, with 10 equally spaced elements (length of
10).
for i = 1:2:10 % starting from the first point to the last
point, with increment of 2
  y = x(i)+i; % execute this statement, which says adding
i to the value of array x at element position i, and name
the result y
  todsiplay = [’ result is ’,num2str(y),’ when i is ’,
num2str(i)];
  disp(todsiplay) % these two lines are a way of asking
matlab to display your result while executing it
end
```

While loop

While loop is like a combination of for and if loop. Given a logic requirement (or a stopping rule), the program keeps looping within the while loop until the stopping rule is fulfilled.

```
while (stopping rule)
  statements
end
```

Note that this is where you may explode your computer if the stopping rule will never be satisfied. Use `ctrl+c` to break the execution.

While loop - example

Type the following program in a script:

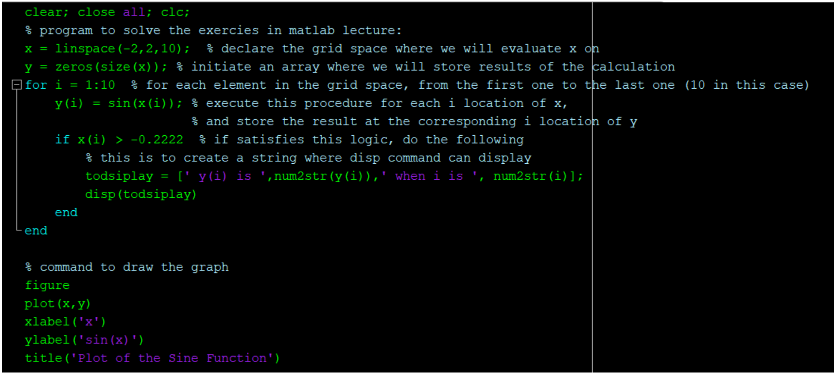
```
iteration = 0; % Initialize a counter with name
"iteration"
x = 1.15; % initialize the starting value of x
while x < 10 % stopping rule:  keeps on going the
program as long as x < 10, and stop immediately if
x >= 10
  x = x+0.1; % statement:  add 0.1 to the existing
value of x, and let it replace the existing x
  iteration = iteration+1;
end
zz = [’total iterations:  ’  num2str(iteration)]; %
display the result as how many iterations we needed
to get x>=10
disp(zz)
```

Exercise

I have the solution at the end of the notes, but try work on it without seeing the solution. Write a program that finds value of $sin(x)$ for x in the grid space $x = linspace(-2, 2, 10)$

1. initialize an array y to store the results of $sin(x)$ by specifying an array y with the same size of the x grid:
   $y = zeros(size(x))$;
2. use for loop to find value of $sin(x)$ for each grid point, and save the value for each grid point in the corresponding element in y array. You may use the following statement in the if-logic:
3. display the result if $x > -0.2222$.
   ```
   todsiplay = [’ y(i) is ’,num2str(y(i)),’ when i is ’, num2str(i)];
   disp(todsiplay)
   ```
4. plot the value of $sin(x)$ in the x grid space using command $plot(sin(x))$

Sample solution script

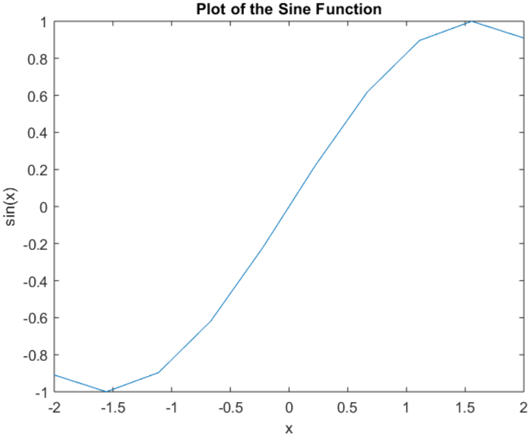


```
clear; close all; clc;
% program to solve the exercies in matlab lecture:
x = linspace(-2,2,10);  % declare the grid space where we will evaluate x on
y = zeros(size(x)); % initiate an array where we will store results of the calculation
for i = 1:10  % for each element in the grid space, from the first one to the last one (10 in this case)
    y(i) = sin(x(i)); % execute this procedure for each i location of x,
                      % and store the result at the corresponding i location of y
    if x(i) > -0.2222  % if satisfies this logic, do the following
        % this is to create a string where disp command can display
        todsiplay = [' y(i) is ',num2str(y(i)),' when i is ', num2str(i)];
        disp(todsiplay)
    end
end

% command to draw the graph
figure
plot(x,y)
xlabel('x')
ylabel('sin(x)')
title('Plot of the Sine Function')
```

Sample solution script

You should also get this graph:


Plot of the Sine Function
sin(x)
x