Introduction to Matlab

Emil Petkov
Pandon 229
emil.petkov@northumbria.ac.uk

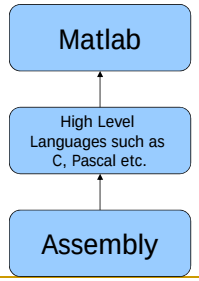
Outline:

- What is Matlab?
- Matlab Screen
- Variables, array, matrix, indexing
- Operators (Arithmetic, relational, logical)
- Display Facilities
- Flow Control
- Using of M-File
- Writing User Defined Functions
- Conclusion

- NU Version MATLAB R2015a

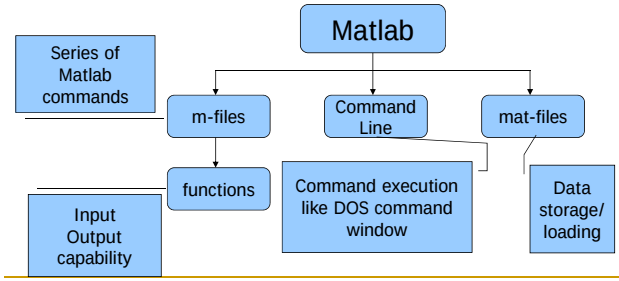
What is Matlab?

- Matlab is basically a high level language which has many specialized toolboxes for making things easier for us
- How high?



What are we interested in?

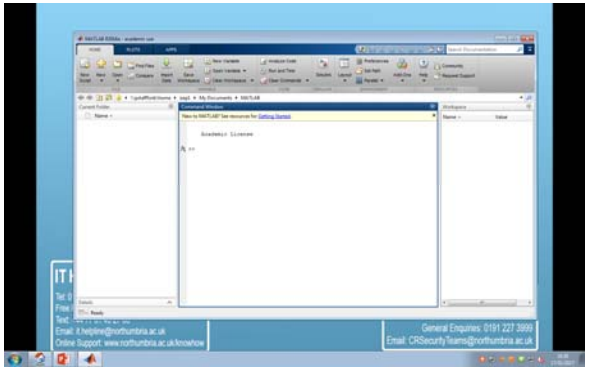
- Matlab is too broad for our purposes in this course.
- The features we are going to require is



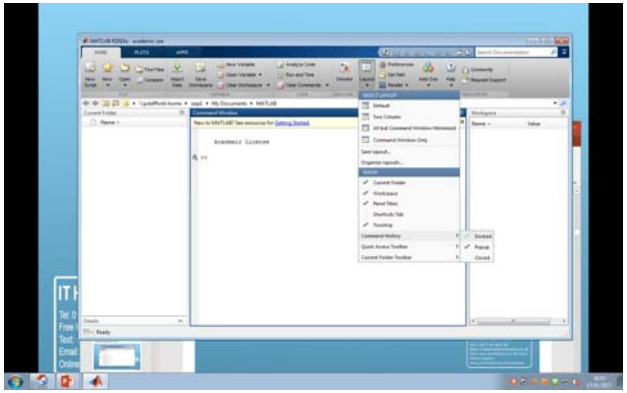
Where is MATLAB

Loading MATLAB….

MATLAB Application Window / Screen

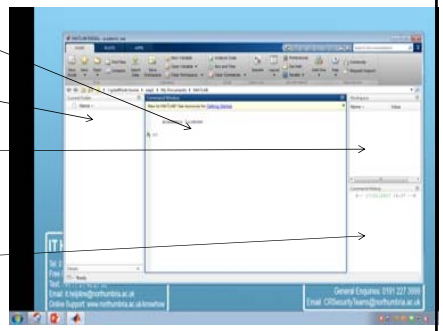


Command History window

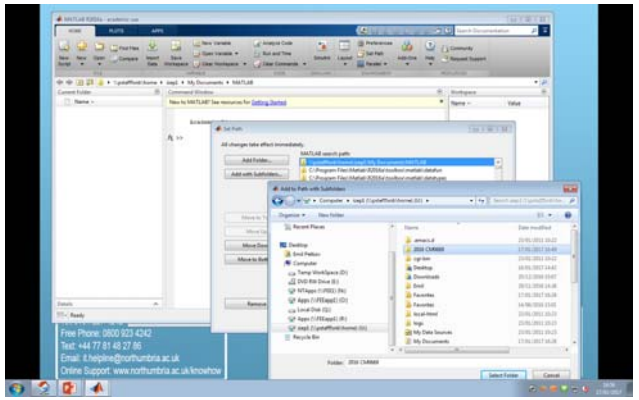


MATLAB screen

- Command Window
  - type commands
- Current Folder
  - View folders and m-files
- Workspace
  - View program variables
  - Double click on a variable to see it in the Array Editor
- Command History
  - view past commands
  - save a whole session using diary

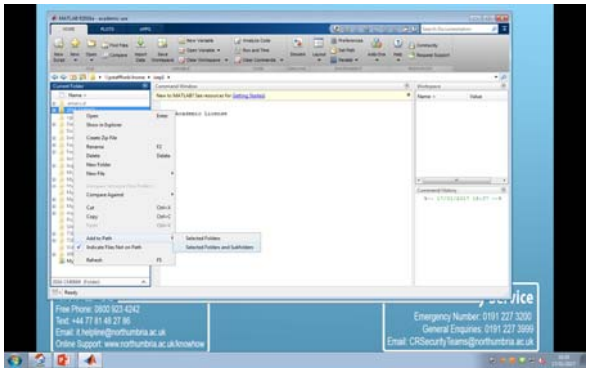


Set path



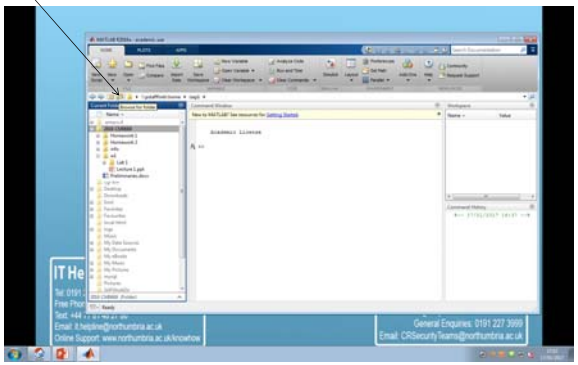
Add a folder to path

Right-click in the “Current Folder” window



Week 1 code directory

Browse for the folder



Variables

- No need for types. i.e.,

```
int a;
double b;
float c;
```

- All variables are created with double precision unless specified and they are matrices.

```
Example:
>>x=5;
>>x1=2;
```

- After these statements, the variables are 1x1 matrices with double precision

Array, Matrix

- a vector `x = [1 2 5 1]`

```
x =
     1   2   5   1
```

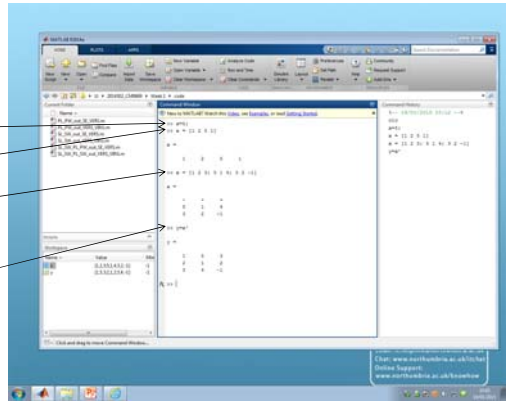
- a matrix `x = [1 2 3; 5 1 4; 3 2 -1]`

```
x =
     1     2     3
     5     1     4
     3     2    -1
```

- transpose `y = x'`

```
y =
     1     5     3
     2     1     2
     3     4    -1
```

MATLAB screen



Long Array, Matrix

- `t =1:10`

```
t =
     1   2   3   4   5   6   7   8   9   10
```

- `k =2:-0.5:-1`

```
k =
    2  1.5  1  0.5  0  -0.5  -1
```

- `B = [1:4; 5:8]`

```
x =
     1     2     3     4
     5     6     7     8
```

Generating Vectors from functions

- zeros(M,N) MxN matrix of zeros

```
x = zeros(1,3)
x =
  0     0     0
```

- ones(M,N) MxN matrix of ones

```
x = ones(1,3)
x =
  1     1     1
```

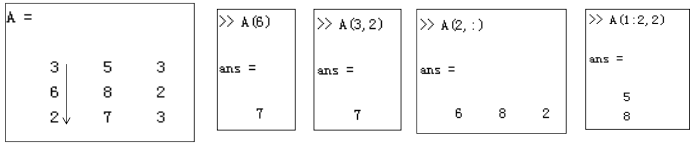
- rand(M,N) MxN matrix of uniformly distributed random numbers on (0,1)

```
x = rand(1,3)
x =
 0.9501  0.2311 0.6068
```

Matrix Index

- The matrix indices begin from 1 (not 0 (as in C))
- The matrix indices must be a positive integer

Given: >>A = [3 5 3; 6 8 2; 2 7 3]



A(0)

Error: ??? Subscript indices must either be real positive integers or logicals.

A(-2), A(4,2)
Error: ??? Index exceeds matrix dimensions.

Concatenation of Matrices

- `x = [1 2];, y = [4 5];, z=[ 0 0];`

```
A = [ x y]% row concatenation

    1   2   4   5

B = [x ; y]% column concatenation

    1 2
    4 5
```

```
C = [x y ;z]
Error:
??? Error using ==> vertcat CAT arguments dimensions are not consistent.
```

Operators (arithmetic)

+ addition
- subtraction
* multiplication
/ division
^ power
‘ complex conjugate transpose

Matrices Operations

Given A and B:
A=[1 2 3;4 5 6;7 8 9]
B=[3 5 2;5 2 8;3 6 9]

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

>> B = [3 5 2; 5 2 8; 3 6 9]
B =
     3     5     2
     5     2     8
     3     6     9
```

Addition

```
>> X = A + B
X =
     4     7     5
     9     7    14
    10    14    18
```

Subtraction

```
>> Y = A - B
Y =
    -2    -3     1
    -1     3    -2
     4     2     0
```

Product

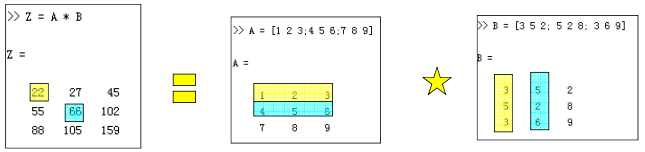
```
>> Z = A * B
Z =
    22    27    45
    55    66   102
    88   105   159
```

Transpose

```
>> T = A'
T =
     1     4     7
     2     5     8
     3     6     9
```

Matrices Operations - Product

Product
Z = A * B
http://en.wikipedia.org/wiki/Matrix_multiplication
http://www.mathsisfun.com/algebra/matrix-multiplying.html



- Z(1,1) = A(1,1) * B(1,1) + A(1,2) * B(2,1) + A(1,3) * B(3,1)
- 22 = 1 * 3 + 2 * 5 + 3 * 3
- Z(2,2) = A(2,1) * B(1,2) + A(2,2) * B(2,2) + A(2,3) * B(3,2)
- 66 = 4 * 5 + 5 * 2 + 6 * 6

Operators (Element by Element)

.* element-by-element multiplication
./ element-by-element division
.^ element-by-element power

The use of "." – "Element" Operation

```
A = [1 2 3; 5 1 4; 3 4 -1]
A =
     1     2     3
     5     1     4
     3     4    -1
```

```
x = A(1,:)
x=
     1     2     3
```

```
y = A(3 ,:)
y=
     3     4    -1
```

```
b = x .* y
b=
     3     8    -3
```

```
c = x ./ y
c=
    0.33   0.5   -3
```

```
d = x .^2
d=
     1     4     9
```

```
K= x^2
Erorr:
??? Error using ==> mpower  Matrix must be square.
Error using  ^
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.
B=x*y
Erorr:
??? Error using ==> mtimes Inner matrix dimensions must agree.
Error using  *
Inner matrix dimensions must agree.
```

Basic Task: Plot the function sin(x) between $0 \le x \le 4\pi$

- Create an x-array of 100 samples between 0 and 4π.

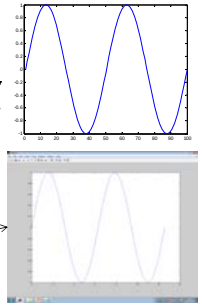
```
>>x=linspace(0,4*pi,100);
```

- Calculate sin(.) of the x-array

```
>>y=sin(x);
```

- Plot the y-array

```
>>plot(y)
>>plot(x,y)
```



Plot the function $e^{-x/3}\sin(x)$ between $0 \le x \le 4\pi$

- Create an x-array of 100 samples between 0 and 4π.

```
>>x=linspace(0,4*pi,100);
```

- Calculate sin(.) of the x-array

```
>>y=sin(x);
```

- Calculate $e^{-x/3}$ of the x-array

```
>>y1=exp(-x/3);
```

- Multiply the arrays y and y1

```
>>y2=y*y1;
```

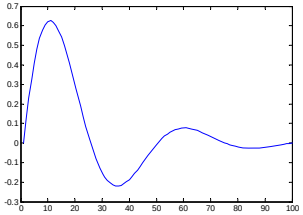
Plot the function $e^{-x/3}\sin(x)$ between $0 \le x \le 4\pi$

- Multiply the arrays y and y1 correctly

```
>>y2=y.*y1;
```

- Plot the y2-array

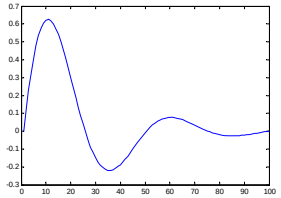
```
>>plot(y2)
```



Display Facilities

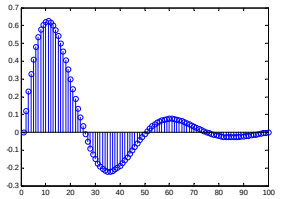
- plot(.)

```
Example:
>>x=linspace(0,4*pi,100);
>>y=sin(x);
>>plot(y)
>>plot(x,y)
```



- stem(.)

```
Example:
>>stem(y)
>>stem(x,y)
```



Display Facilities

- title(.)

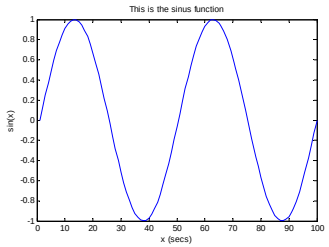
```
>>title('This is the sinus function')
```

- xlabel(.)

```
>>xlabel('x (secs)')
```

- ylabel(.)

```
>>ylabel('sin(x)')
```



Operators (relational, logical)

- == Equal to
- ~= Not equal to
- < Strictly smaller
- > Strictly greater
- <= Smaller than or Equal to
- >= Greater than or Equal to
- & (&& is also correct) AND operator
- | (|| is also correct) OR operator

Flow Control

- if
- for
- while
- break
- ….

Control Structures

- If Statement Syntax

```
if (Condition_1)
        Matlab Commands
elseif (Condition_2)
        Matlab Commands
elseif (Condition_3)
        Matlab Commands
else
        Matlab Commands
end
```

Some Dummy Examples

```
if ((a>3) & (b==5))
    Some Matlab Commands;
end
```

```
if (a<3)
    Some Matlab Commands;
elseif (b~=5)
    Some Matlab Commands;
end
```

```
if (a<3)
    Some Matlab Commands;
else
    Some Matlab Commands;
end
```

Control Structures

- For loop syntax

```
for i=Index_Array
   Matlab Commands
end
```

Some Dummy Examples

```
for i=1:100
    Some Matlab Commands;
end
```

```
for j=1:3:200
    Some Matlab Commands;
end
```

```
for m=13:-0.2:-21
    Some Matlab Commands;
end
```

```
for k=[0.1 0.3 -13 12 7 -9.3]
    Some Matlab Commands;
end
```

Control Structures

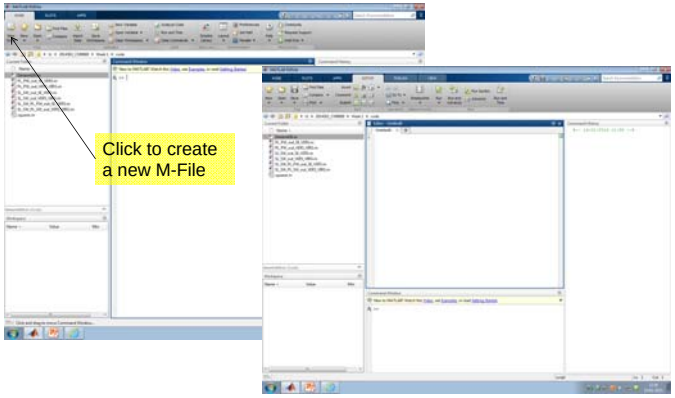
- While Loop Syntax

```
while (condition)
   Matlab Commands
end
```

Dummy Example

```
while  ((a>3) & (b==5))
    Some Matlab Commands;
end
```

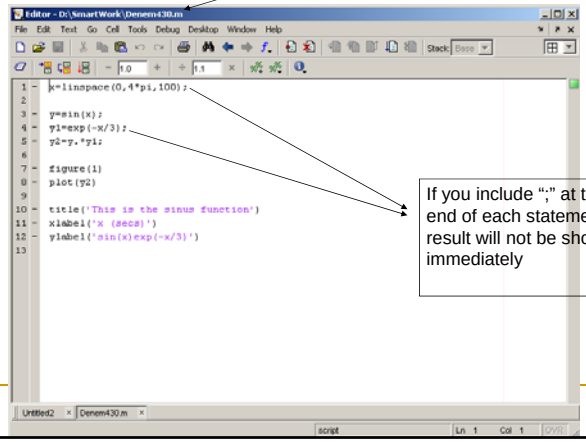
Use of M-File- New script



- Extension ".m"
- A text file containing script or function or program to run

Use of M-File

Save file as *Denem430*.m



If you include ";" at the end of each statement, result will not be shown immediately

Save file as *Denem430*.m

```
x=linspace(0,4*pi,100);

y=sin(x);
y1=exp(-x/3);
y2=y.*y1;

figure(1)
plot(y2)

title('This is the sinus function')
xlabel('x (sec)')
ylabel('sin(x)exp(-x/3)')
```

Writing User Defined Functions

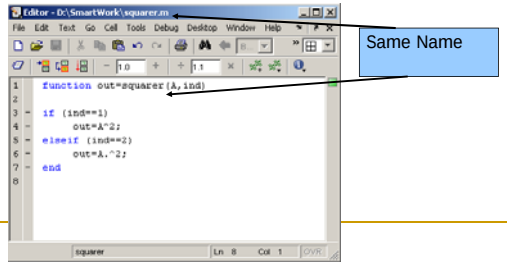
- Functions are m-files which can be executed by specifying some inputs and supply some desired outputs.
- The code telling the Matlab that an m-file is actually a function is

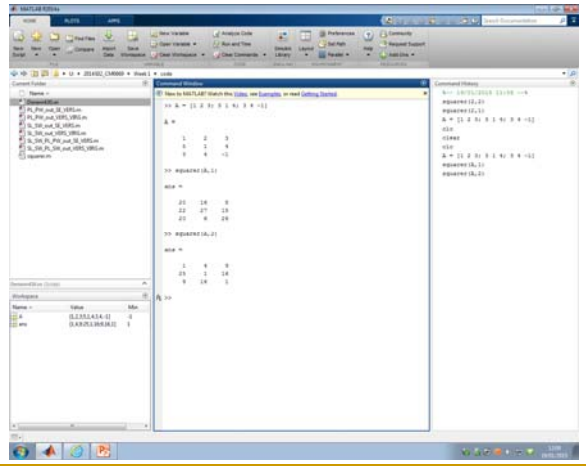
```
function out1=functionname(in1)
function out1=functionname(in1,in2,in3)
function [out1,out2]=functionname(in1,in2)
```

- You should write this command at the beginning of the m-file and you should save the m-file with a file name same as the function name

Writing User Defined Functions

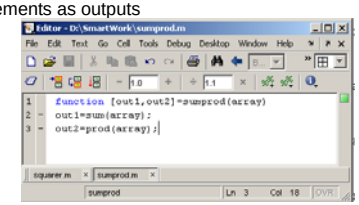
- Examples
  - Write a function : out=squarer (A, ind)
    - Which takes the square of the input matrix if the input indicator is equal to 1
    - And takes the element by element square of the input matrix if the input indicator is equal to 2



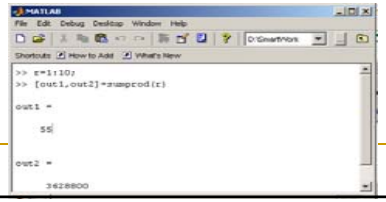


Writing User Defined Functions

- Another function which takes an input array and returns the sum and product of its elements as outputs



- The function sumprod(.) can be called from command window or an m-file as



Save file as *sumprod*.m

```
function [out1,out2]=sumprod(array)
out1=sum(array);
out2=prod(array);
```

Notes:

- "%" is the neglect sign for Matlab (equaivalent of "//" in C). Anything after it on the same line is neglected by Matlab compiler.
- Sometimes slowing down the execution is done deliberately for observation purposes. You can use the command "pause" for this purpose

```
pause %wait until any key
pause(3) %wait 3 seconds
```

Useful Commands

- The two commands used most by Matlab users are

```
>>help functionname
>>lookfor keyword
>>clc
>>clear
```

Getting Started with MATLAB
http://uk.mathworks.com/help/matlab/getting-started-with-matlab.html

Questions

- ?
- ?
- ?
- ?
- ?