

Matlab

- Stands for MATrix LABoratory
- Interpreted language
- Scientific programming environment
- Very good tool for the manipulation of matrices
- Great visualisation capabilities
- Loads of built-in functions
- Easy to learn and simple to use

Matlab Desktop

The screenshot displays the MATLAB Desktop environment. The top menu bar includes File, Edit, View, Web, Window, and Help. The current directory is set to 'w:\work'. The Workspace window shows a list of variables with their names, sizes, byte counts, and classes. The Command Window shows the execution of several MATLAB commands and their outputs. The Command History window shows a list of previously executed commands.

Name	Size	Bytes	Class
C1_4096	1x4	2359536	cell array
C2_4096	1x4	6029552	cell array
Dist1_4096	4096x8x2	3580696	cell array
Dist2_4096	4096x22x2	10731648	cell array
Dist4_4096	4096x70x2	35755056	cell array
Ind4096	4096x7	229376	double array
M4096	1x4096	11264400	cell array
Mout4096	2x4096	22528800	cell array
Neigh4096	4096x8	262144	double array
NeighN2_4096	2x4096	1080464	cell array
NeighN4_4096	4x4096	2951648	cell array
S1_4096	1x4	2359536	cell array
S2_4096	1x4	6029552	cell array
W4096	4096x2	65536	double array
ans	1x2	16	double array
tri4096	8106x3	194544	double array

```
>> size(Ind4096,2)
ans =
     7
>> size(Ind4096)
ans =
    4096     7
>> whos
Name           Size           Bytes  Class
C1_4096         1x4             2359536  cell array
C2_4096         1x4             6029552  cell array
Dist1_4096     4096x8x2       3580696  cell array
Dist2_4096     4096x22x2     10731648  cell array
Dist4_4096     4096x70x2     35755056  cell array
Ind4096        4096x7         229376   double array
M4096          1x4096         11264400  cell array
Mout4096       2x4096         22528800  cell array
Neigh4096     4096x8         262144   double array
NeighN2_4096  2x4096         1080464  cell array
NeighN4_4096  4x4096         2951648  cell array
S1_4096        1x4             2359536  cell array
S2_4096        1x4             6029552  cell array
W4096         4096x2         65536    double array
ans            1x2             16       double array
tri4096       8106x3         194544   double array

Grand total is 8186522 elements using 105422504 bytes
>>
```

Workspace /
Current Directory

Command
Window

Command
History



Explore the Matlab Desktop

Variables

- Don't have to declare type
- Don't even have to initialise
- Just assign in command window

>>

>> a=12; % variable a is assigned 12

Matlab
prompt

assign
operator

suppress
command
output

comment
operator



Try the same line without the semicolon and comments

Variables (continued ...)

- View variable contents by simply typing the variable name at the command prompt

```
>> a
```

```
a =
```

```
    12
```

```
>>
```

```
>> a*2
```

```
a =
```

```
    24
```

```
>>
```

Workspace

- The workspace is Matlab's memory
- Can manipulate variables stored in the workspace

```
>> b=10;
```

```
>> c=a+b
```

```
c =
```

```
    22
```

```
>>
```

Workspace (continued ...)

- Display contents of workspace

```
>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array
b	1x1	8	double array
c	1x1	8	double array

```
Grand total is 3 elements using 24 bytes
```

```
>>
```

- Delete variable(s) from workspace

```
>> clear a b; % delete a and b from workspace
```

```
>> whos
```

```
>> clear all; % delete all variables from workspace
```

```
>> whos
```

Matlab help commands

- help
 - >> help whos % displays documentation for the function whos
 - >> lookfor convert % displays functions with convert in the first help line
- Start Matlab help documentation
 - >> helpdesk

Matrices

- Don't need to initialise type, or dimensions

```
>>A = [3 2 1; 5 1 0; 2 1 7]
```

```
A =
```

```
3 2 1
```

```
5 1 0
```

```
2 1 7
```

```
>>
```

square brackets to define matrices

semicolon for next row in matrix

Manipulating Matrices

```
A =  
    3    2    1  
    5    1    0  
    2    1    7
```

- Access elements of a matrix

```
>>A(1,2)
```

```
ans =
```

```
2
```

indices of matrix element(s)



- Remember Matrix(row,column)
- Naming convention Matrix variables start with a capital letter while vectors or scalar variables start with a simple letter

The : operator

- VERY important operator in Matlab
- Means 'to'

```
>> 1:10
```

```
ans =
```

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

```
>> 1:2:10
```

```
ans =
```

```
1 3 5 7 9
```



Try the following
>> x=0:pi/12:2*pi;
>> y=sin(x)

The : operator and matrices

```
>>A(3,2:3)
```

```
ans =
```

```
1 7
```

```
>>A(:,2)
```

```
ans =
```

```
2
```

```
1
```

```
1
```

```
A =
```

```
3 2 1
```

```
5 1 0
```

```
2 1 7
```



What'll happen if you type A(:,,:) ?

Manipulating Matrices

A =
3 2 1
5 1 0
2 1 7

```
>> A'           % transpose  
>> B*A         % matrix multiplication  
>> B.*A        % element by element multiplication  
>> B/A         % matrix division  
>> B./A        % element by element division  
>> [B A]       % Join matrices (horizontally)  
>> [B; A]      % Join matrices (vertically)
```

B =
1 3 1
4 9 5
2 7 2



Enter matrix B
into the Matlab
workspace

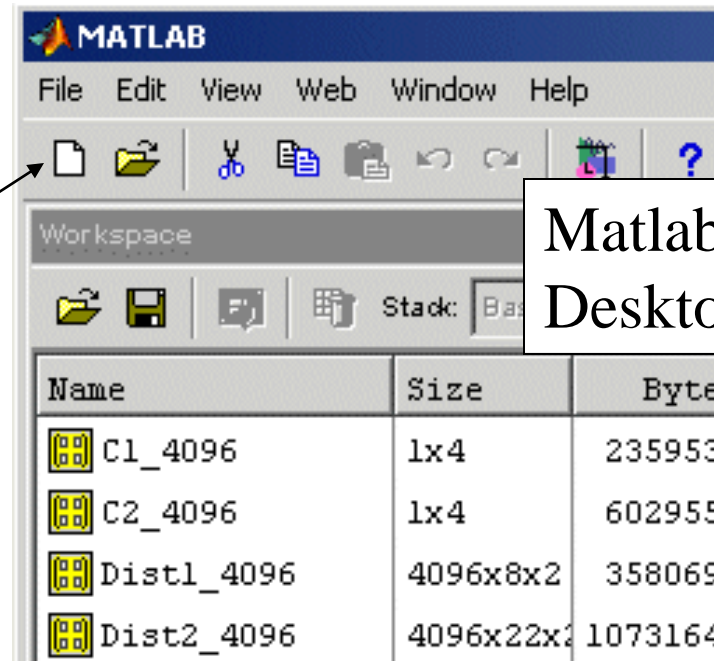


Create matrices A and B and try out the the matrix operators in this slide

Scripts

- Matlab editor
- Use scripts to execute a series of Matlab commands

Press to create new m-file in the matlab editor



Matlab
Desktop

Scripts (continued)

- Scripts will manipulate and store variables and matrices in the Matlab Workspace (memory).

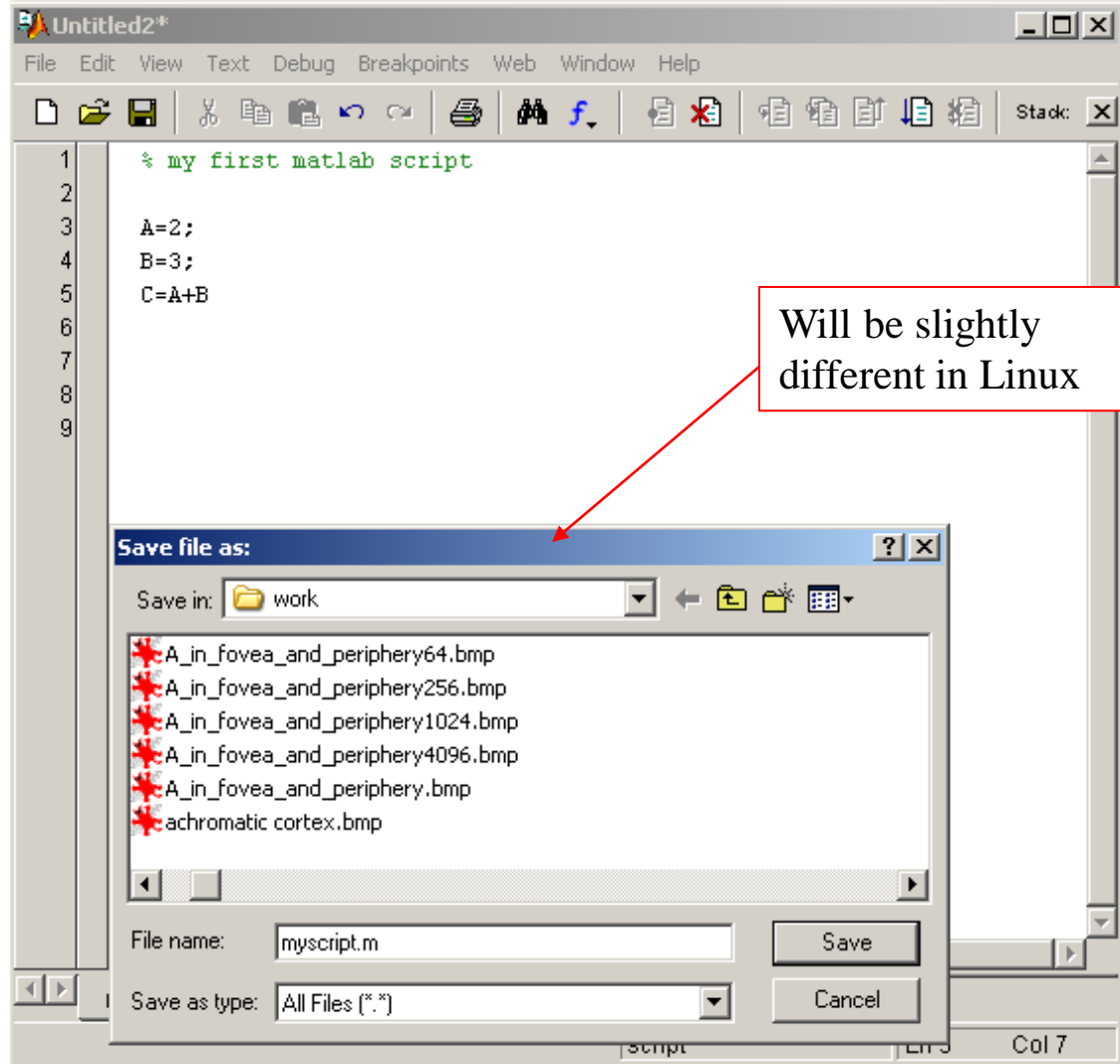
- They can be called from the Matlab command line by typing the (case sensitive!) filename of the script file.

>> myscript

- Scripts can be opened in the editor by the following

>> open myscript

Highlight a few lines of your script by left- clicking and dragging the mouse over the lines. Right-click the highlighted lines and select Evaluate Selection.



Functions

- Programming in Matlab.
- Users can write functions which can be called from the command line.
- Functions can accept input variable(s)/matrice(s) and will output variable(s)/matrice(s).
- Functions will **not** manipulate variable(s)/matrice(s) in the Matlab Workspace.
- In Matlab functions closely resemble scripts and can be written in the Matlab editor. Matlab functions have the **function** keyword.
- Remember that the filename of a function will be its calling function name.
- Don't overload any built-in functions by using the same filename for your functions or scripts!
- Functions can be opened for editing using the **open** command. Many built-in Matlab functions can also be viewed using this command.

Functions (continued)

```
>> I=iterate(5)
```

```
I =
```

```
1    4    9   16   25
```

output

function name

input

function keyword

help lines for function

for statement block

```
w:\work\iterate.m
File Edit View Text Debug Breakpoints Web Window Help
function O=iterate(n)
% iterate(n) outputs the square of the
% integers upto integer n
for i=1:n
O(i)=i*i;
end
```

Make sure you save changes to the m-file before you call the function!



Access the comments of your Matlab functions

```
>> help iterate
```


Functions (continued)

```
>> [i j]=sort2(2,4)
```

```
i =
```

```
4
```

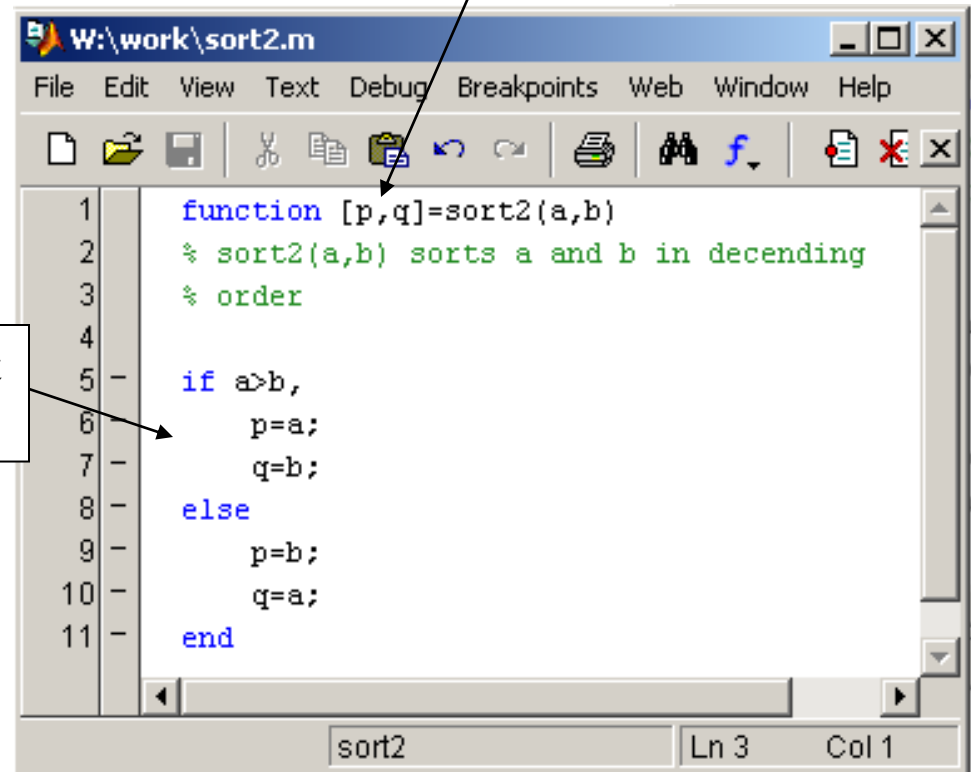
```
j =
```

```
2
```

```
>>
```

if statement
block

Functions can have many
outputs contained in a matrix



```
function [p,q]=sort2(a,b)
% sort2(a,b) sorts a and b in decending
% order
if a>b,
    p=a;
    q=b;
else
    p=b;
    q=a;
end
```

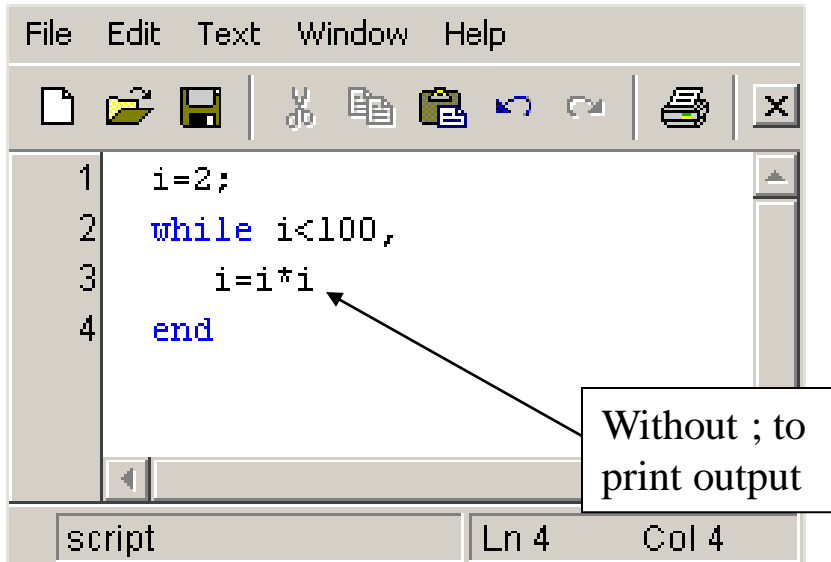


Remember to use the
Matlab help command for
syntax

```
>> help if
```

More flow control

While statement block



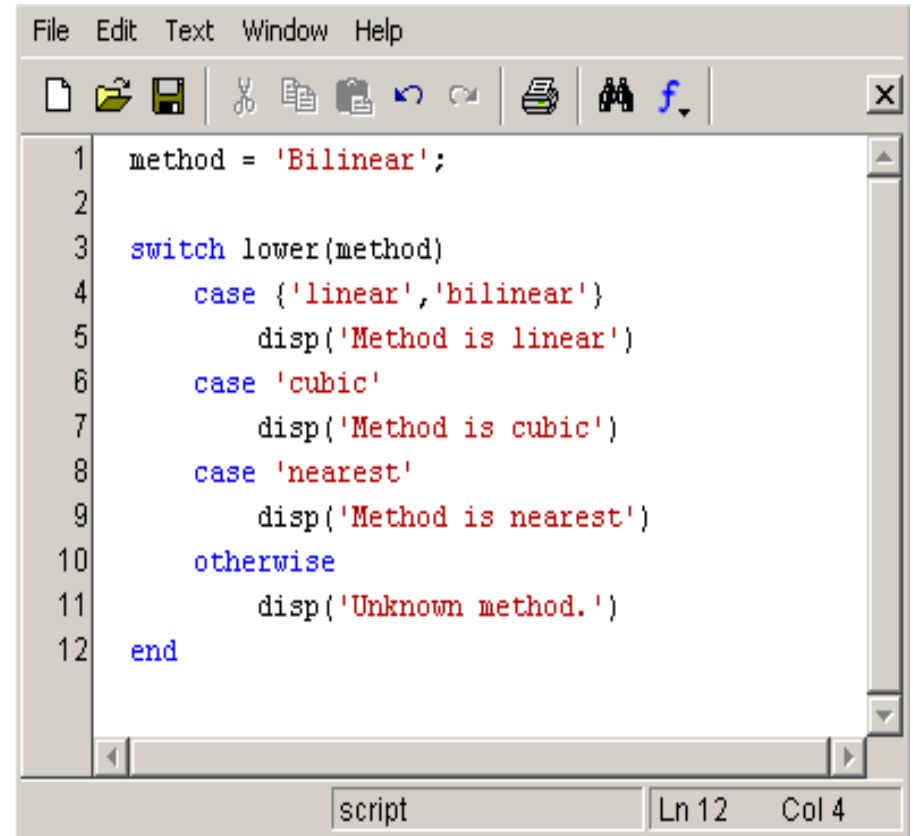
```
File Edit Text Window Help
[Icons]
1 i=2;
2 while i<100,
3     i=i*i
4 end
```

Without ; to print output

script Ln 4 Col 4

```
i =
     4
i =
    16
i =
   256
```

Switch statement block



```
File Edit Text Window Help
[Icons]
1 method = 'Bilinear';
2
3 switch lower(method)
4     case {'linear','bilinear'}
5         disp('Method is linear')
6     case 'cubic'
7         disp('Method is cubic')
8     case 'nearest'
9         disp('Method is nearest')
10    otherwise
11        disp('Unknown method.')
12 end
```

script Ln 12 Col 4

```
Method is linear
>>
```

Debugging

- Set breakpoints to stop the execution of code

```
>> [i j]=sort2(2,4)
```

```
K>>
```

```
K>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array
b	1x1	8	double array

Grand total is 2 elements using 16 bytes

```
K>> a
```

```
a =  
    2
```

```
K>> return
```

```
i =
```

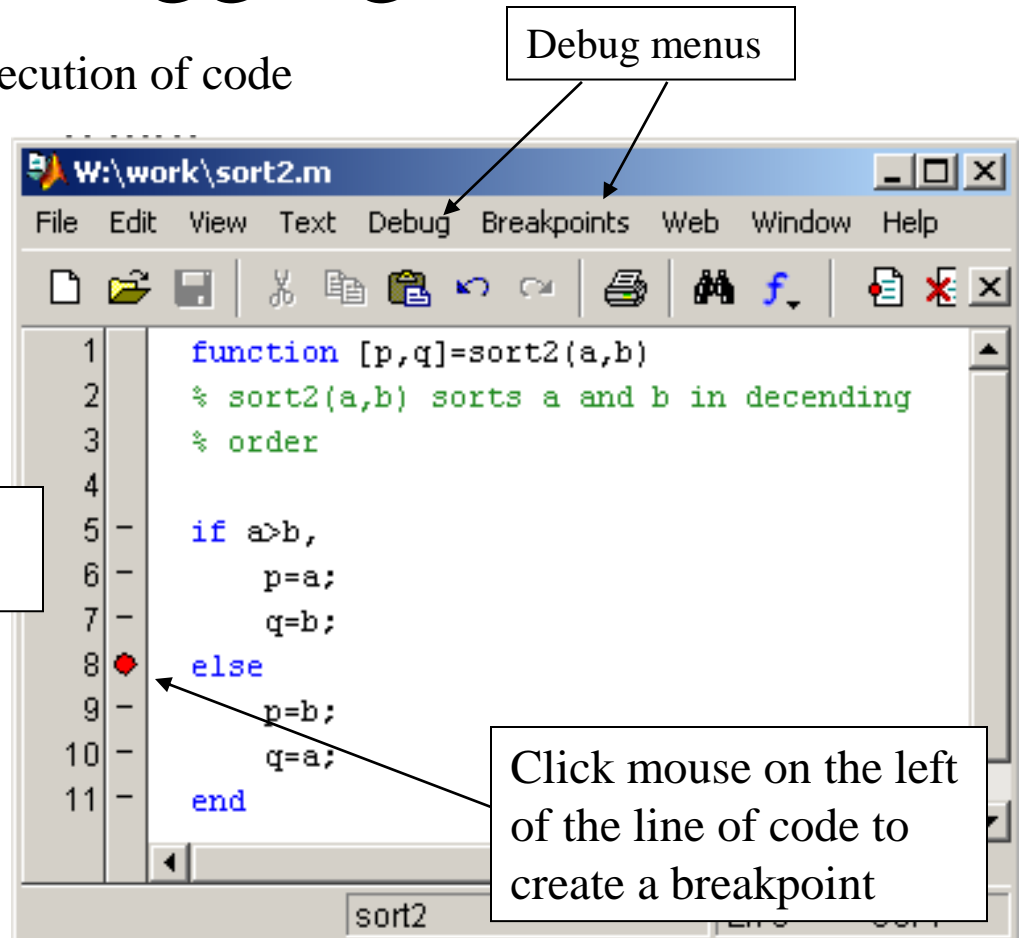
```
    4
```

```
j =
```

```
    2
```

local function workspace

exit debug mode



Visualisation - plotting data

```
>> figure % create new figure
```

```
>> t=0:pi/12:8*pi;
```

```
>> y=cos(t);
```

```
>> plot(t,y,'b.-')
```

Plot style

Investigate the function

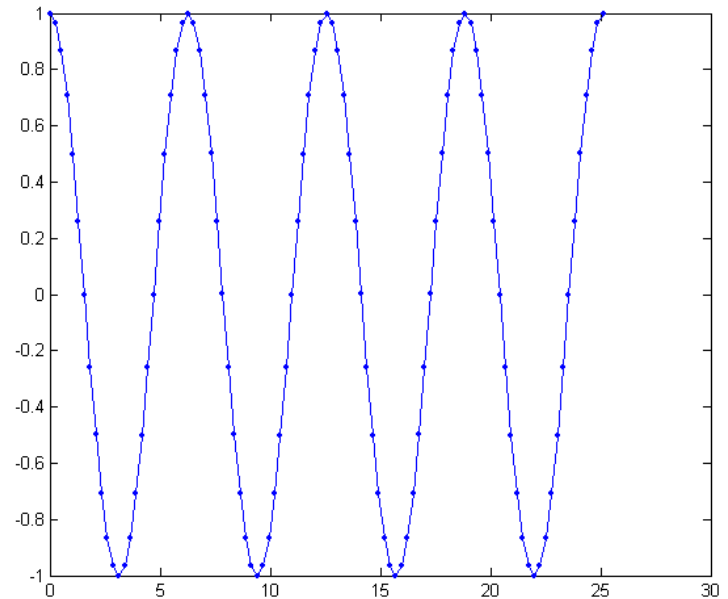
```
>> y=A*cos(w*t+phi);
```

for different values of phi (eg: 0, pi/4, pi/3, pi/2), w (eg: 1, 2, 3, 4) and A (eg: 1, 0.5, 2). Use



the **hold on** Matlab command to display your plots in the same figure. Remember to type **hold off** to go back to normal plotting mode.

Try using different plot styles (**help plot**)



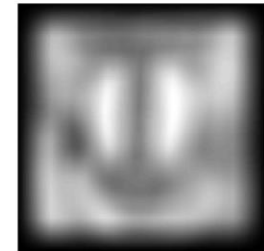
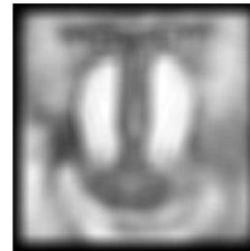
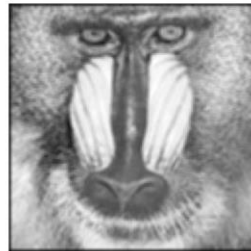
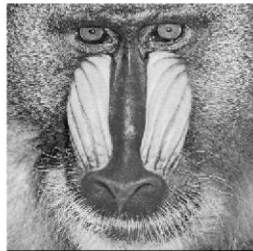
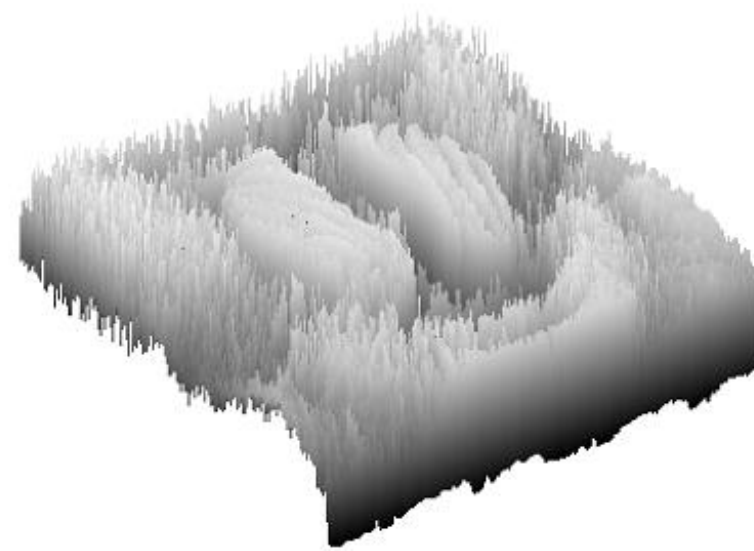
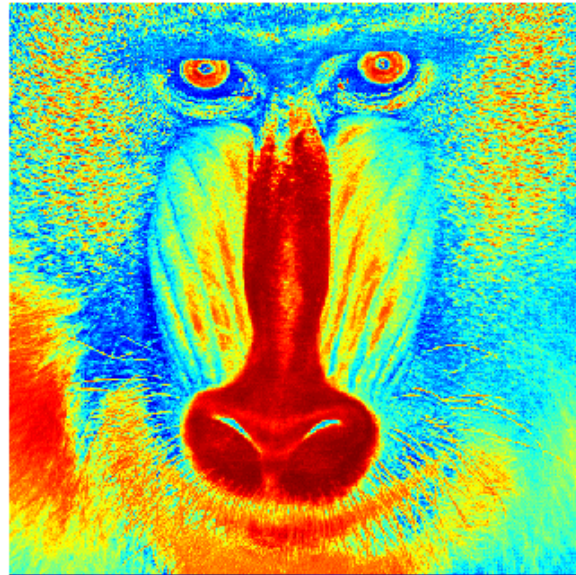
A = amplitude

phi = phase

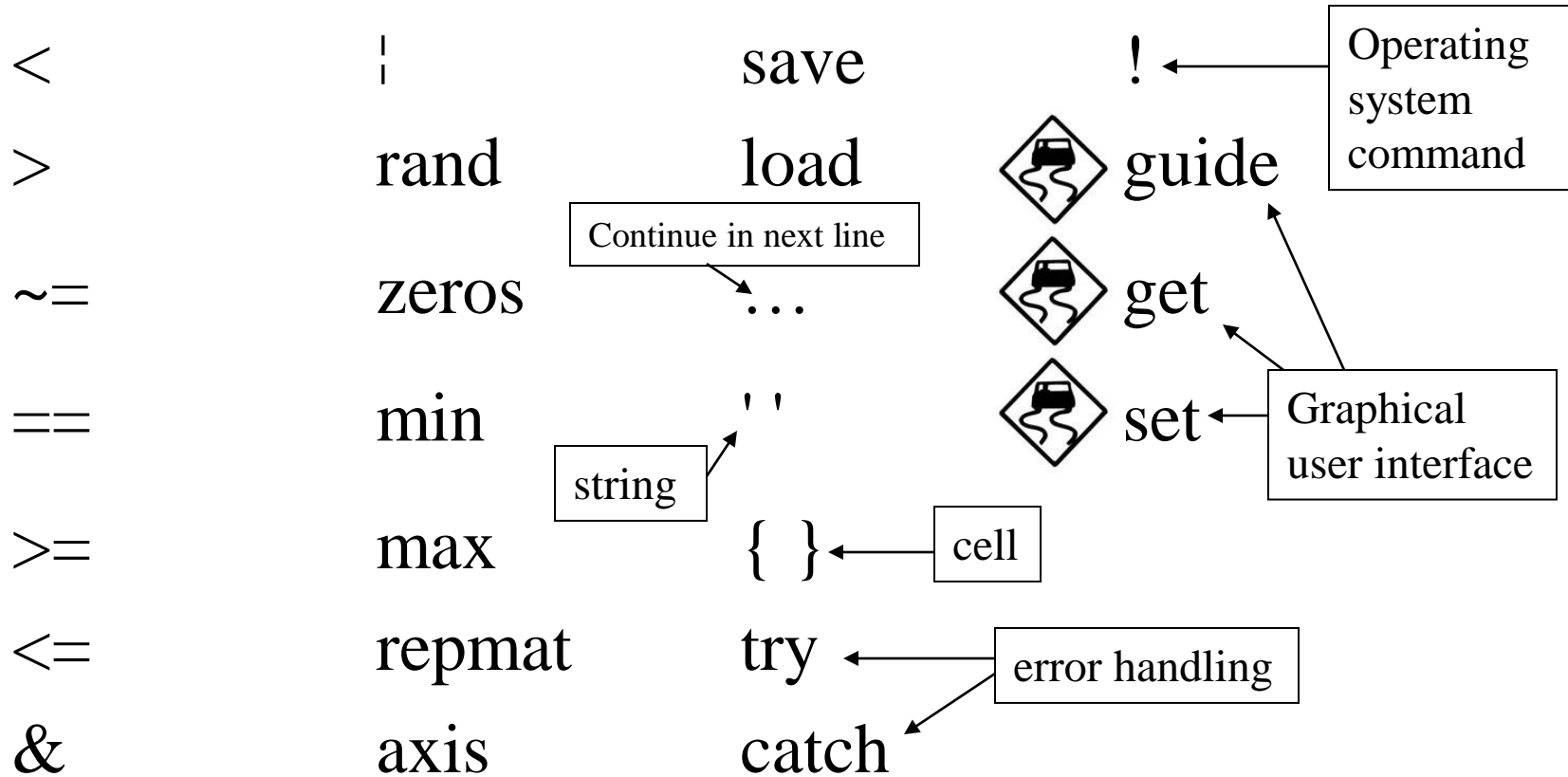
w = angular frequency = $2*\pi*$ frequency

Next week ...

Image Processing using Matlab



Useful operators and built-in functions



Remember to use the Matlab help command if you get stuck

Tutorial 1

- Login to your workstation, start Matlab and create a working directory
 - 1) Login to Linux using your username/password
 - 2) Open a terminal session by right clicking the mouse on the screen and selecting New Terminal
 - 3) Type the following in the terminal session (do not type the prompt sign >)

```
> matlab  
> mkdir work
```
 - 4) Type the following in Matlab (do not type the prompt sign >>)

```
>> cd work
```
- Explore Matlab! Use the **help** matlab command to understand the built-in Matlab functions
- Type the code in this handout in Matlab and investigate the results.
- Write a Matlab function fibonacci.m to generate the Fibonacci series. This is generated by starting with zero and one and adding the last two numbers of the sequence to generate the next number in the series. Fibonacci series:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, ...
- Create an graph of the Fibonacci series using the built-in **plot** Matlab function. Your graph should resemble figure 1 which contains a plot of the first 20 numbers in the sequence.
- Plot the Fibonacci series in polar coordinates using the built-in Matlab **polar** function. Eccentricity (rho) should be the Fibonacci number and angle (theta) should vary with the Fibonacci number's order in the sequence. Your plot should resemble figure 2 which is a polar plot of the first 10 numbers of the series.
- Exit Matlab by typing quit and logout of Linux.

```
>> quit
```

Figure 1

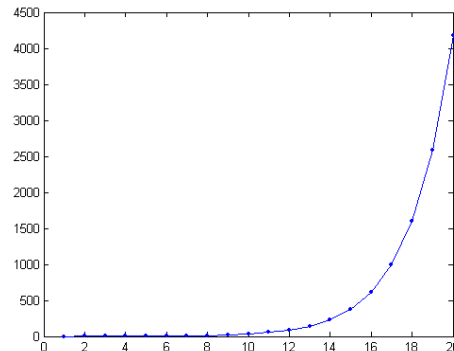


Figure 2

