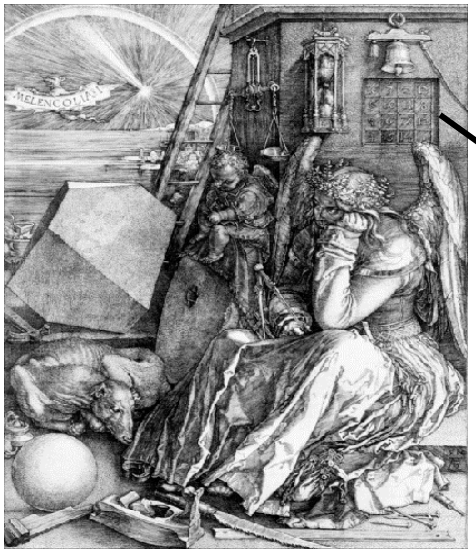
Matlab Lecture 1 - Introduction to MATLAB

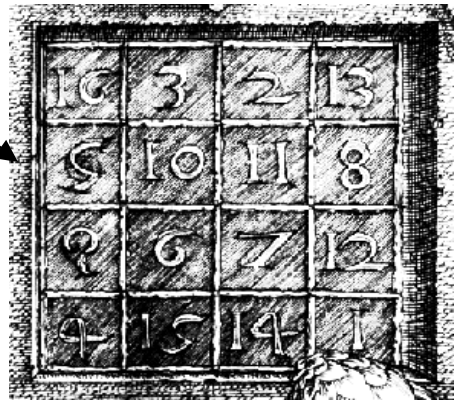
- **MATLAB** is a high-performance language for ***technical computing***. It integrates computation, visualization, and programming in an easy-to-use environment. Typical uses include:
  - Math and computation
  - Algorithm development
  - Modeling, simulation, and prototyping
  - Data analysis, exploration, and visualization
  - Scientific and engineering graphics
- MATLAB is an ***interactive*** system whose basic data element is an ***array*** that does not require dimensioning. This allows you to solve many technical computing problems, especially those with ***matrix*** and ***vector*** formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or Fortran.

Five Parts of Matlab

- **The MATLAB language**
  - High-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features
- **The MATLAB working environment**
  - Facilities for managing the variables and importing and exporting data
  - Tools for developing, managing, debugging, and profiling M-files
- **Handle Graphics**
  - Two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics
  - Graphical User Interface functions
- **The MATLAB mathematical function library**
- **The MATLAB Application Program Interface (API)**
  - Allows you to write C and Fortran programs that interact with MATLAB

Entering Matrices (1) - Magic Square





- Engraving by Albrecht Dűrer, German artist and mathematician in 1514.

Entering Matrices (2) - Method 1:Direct entry

- **4 ways of entering matrices in MATLAB**:
  - Enter an explicit list of elements
  - Load matrices from external data files
  - Generate matrices using built-in functions
  - Create matrices with your own functions in M-files
- **Rules of entering matrices**:
  - Separate the elements of a row with ***blanks*** or commas
  - Use a ***semicolon*** ";" to indicate the end of each row
  - Surround the entire list of elements with ***square brackets***, [ ]
- To enter Dürer's matrix, simply type:

```
» A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

- MATLAB displays the matrix you just entered,

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

No need to define or declare size of A

Entering Matrices (3) - as lists

- Why is this a magic square? Try this in Matlab :-

```
» sum(A)
ans =
    34    34    34    34
» A'
ans =
    16     5     9     4
     3    10     6    15
     2    11     7    14
    13     8    12     1
» sum(A')'
ans =    34
         34
         34
         34
```

Result in row vector variable `ans`

Compute the sum of each column in A

Transpose matrix A

Compute the sum of each row in A

Result in cloumn vector variable `ans`

Entering Matrices (4) - subscripts

- `A(i,j)` refers to element in row `i` and column `j` of `A` :-

row

col

```
» A(4,2)
ans = 15
» A(1,4) + A(2,4) + A(3,4) + A(4,4)
ans = 34
» X = A;
» X(4,5) = 17
X =
    16     3     2    13     0
     5    10    11     8     0
     9     6     7    12     0
     4    15    14     1    17
```

Slow way of finding sum of column 4

Make another copy of A in X ';' suppress output

Add one element in column 5, auto increase size of matrix

Entering Matrices (5) - colon ; Operator

- ';' used to specify range of numbers

start

end

```
» 1:10
ans = 1  2  3  4  5  6  7  8  9  10
» 100:-7:50
ans = 100  93  86  79  72  65  58  51
» 0:pi/4:pi
ans = 0  0.7854  1.5708  2.3562  3.1416
» A(1:k,j);
» sum(A(1:4,4))
ans = 34
» sum(A(:,end))
ans = 34
```

incr

First k elements of the j^{th} column in A

'0' to 'pi' with incr. of 'pi/4'

Short-cut for "all rows"

last col

Expressions & built-in functions

Elementary functions

```
» rho = (1+sqrt(5))/2
rho =  1.6180
```

Complex number

```
» a = abs(3+4i)
a =  5
```

Special functions

```
» z = sqrt(besselk(4/3,rho-i))
z = 0.3730+ 0.3214i
```

Built-in constants (function)

```
» huge = exp(log(realmax))
huge =  1.7977e+308
```

```
» toobig = pi*huge
toobig = Inf
```

pi	3.14159265
I or j	Imaginary unit, -1
eps	FP relative precision, 2^{-52}
realmin	Smallest FP number, 2^{-1022}
realmax	Largest FP number, (2-)2^{1023}
Inf	Infinity
NaN	Not-a-number

Entering Matrices (6) - Method 2: Generation

```
» Z = zeros(2,4)
Z =   0     0     0     0
      0     0     0     0

» F = 5*ones(3,3)
F =   5     5     5
      5     5     5
      5     5     5

» N = fix(10*rand(1,10))
N =   4     9     4     4

» R = randn(4,4)
R =   1.0668   0.2944   0.6918  -1.4410
      0.0593  -1.3362   0.8580   0.5711
     -0.0956   0.7143   1.2540  -0.3999
     -0.8323   1.6236  -1.5937   0.6900
```

Useful Generation Functions

- `Zeros` — All zeros
- `ones` — All ones
- `rand` — Uniformly distributed random elements between (0.0, 1.0)
- `randn` — Normally distributed random elements, mean = 0.0, var = 1.0

Entering Matrices (7) - Method 3 & 4: Load & M-File

magik.dat

```
16.0  3.0  2.0 13.0
 5.0 10.0 11.0  8.0
 9.0  6.0  7.0 12.0
 4.0 15.0 14.0  1.0
```

```
» load magik.dat
```

Read data from file into variable `magik`

```
» magik
```

.m files can be run by just typing its name in Matlab

Three dots (…) means continuation to next line

magik.m

```
A = [ ...
16.0  3.0  2.0 13.0
 5.0 10.0 11.0  8.0
 9.0  6.0  7.0 12.0
 4.0 15.0 14.0  1.0};
```

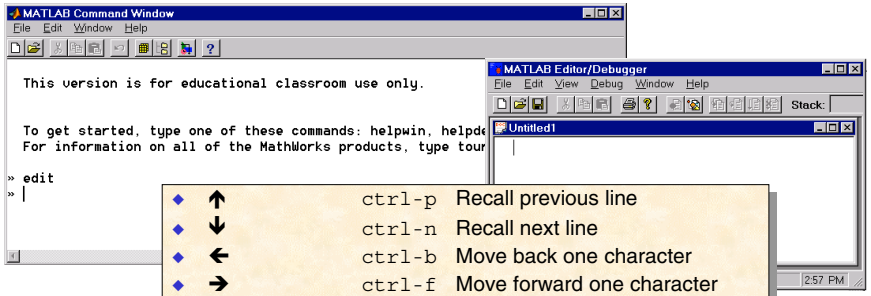

Entering Matrices (8) - Concatenate & delete

```
» B = [A  A+32; A+48  A+16]
B =
    16     3     2    13    48    35    34    45
     5    10    11     8    37    42    43    40
     9     6     7    12    41    38    39    44
     4    15    14     1    36    47    46    33
    64    51    50    61    32    19    18    29
    53    58    59    56    21    26    27    24
    57    54    55    60    25    22    23    28
    52    63    62    49    20    31    30    17
```

2nd column deleted

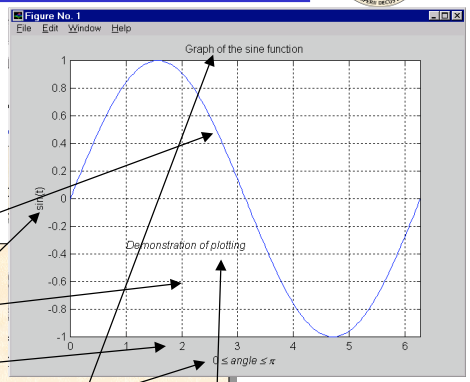
```
» X = A;
» X(:,2) = []
X =
    16     2    13
     5    11     8
     9     7    12
     4    14     1
```


Command Window



↑	`ctrl-p`	Recall previous line
↓	`ctrl-n`	Recall next line
←	`ctrl-b`	Move back one character
→	`ctrl-f`	Move forward one character
ctrl - →	`ctrl-r`	Move right one word
ctrl - ←	`ctrl-l`	Move left one word
home	`ctrl-a`	Move to beginning of line
end	`ctrl-e`	Move to end of line
esc	`ctrl-u`	Clear line
del	`ctrl-d`	Delete character at cursor
backspace	`ctrl-h`	Delete character before cursor
	`ctrl-k`	Delete to end of line

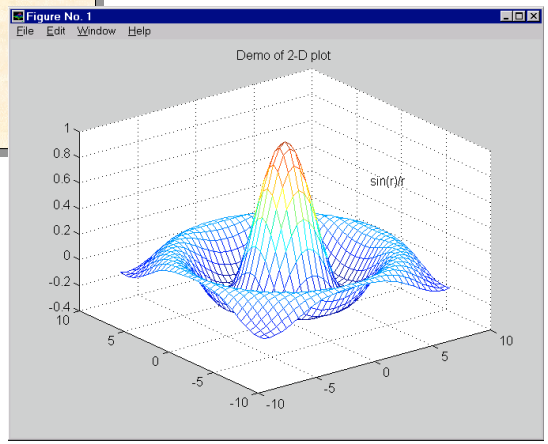
MATLAB Graphics(1) - Creating a Plot




```
» t = 0:pi/100:2*pi;
» y = sin(t);
» plot(t,y)
» grid
» axis([0 2*pi -1 1])
» xlabel('0 \leq \itangle \leq \pi')
» ylabel('sin(t)')
» title('Graph of the sine function')
» text(1,-1/3,'\it{Demonstration of plotting}')
```

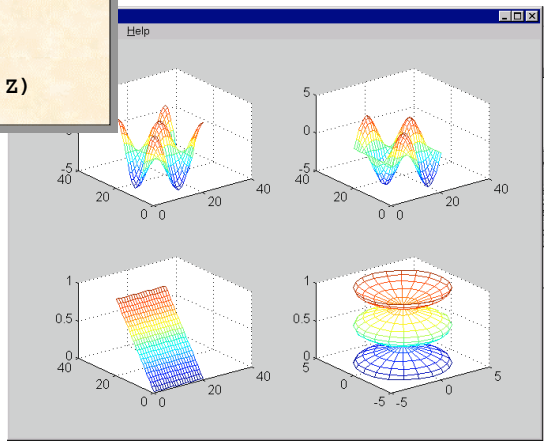

MATLAB Graphics(2) - Mesh & surface plots

```
» [X,Y] = meshgrid(-8:.5:8);
» R = sqrt(X.^2 + Y.^2) + eps;
» Z = sin(R)./R;
» mesh(X,Y,Z)
» text(15,10,'sin(r)/r')
» title('Demo of 2-D plot');
```



MATLAB Graphics(3) - Subplots

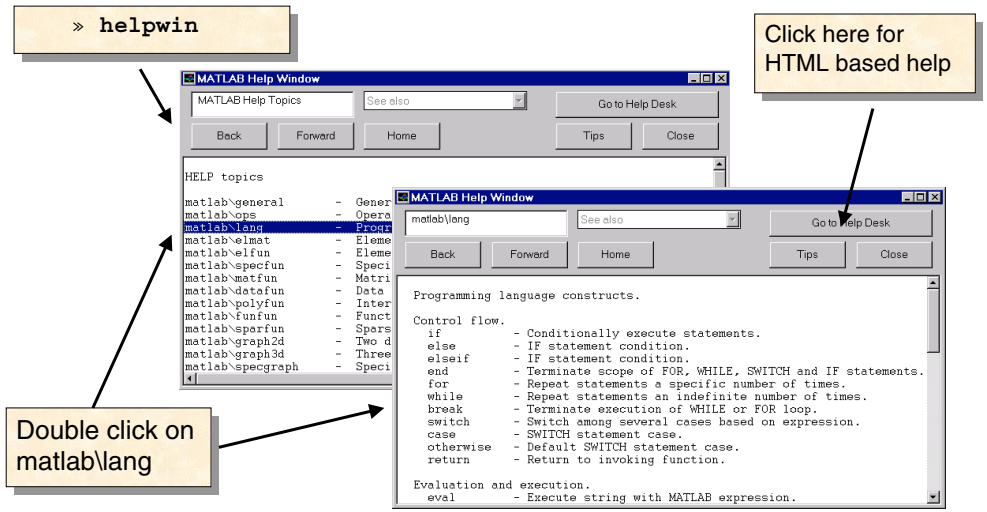
```
» t = 0:pi/10:2*pi;
» [X,Y,Z] = cylinder(4*cos(t));
» subplot(2,2,1); mesh(X)
» subplot(2,2,2); mesh(Y)
» subplot(2,2,3); mesh(Z)
» subplot(2,2,4); mesh(X,Y,Z)
```



MATLAB Graphics(3) - Subplots

- Matlab official method: generate encapsulated postscript files -
  ```
  » print -depsc2 mesh.eps
  ```
- My method:-
  - Use <***PrintScreen***> key (top right corner) to capture the plot on screen
  - Use MS Photo Editor or similar bit-map editing program to cut out the the plot that I want
  - Paste it into MS Word or MS PowerPoint or save it as .BMP/.GIF file
  - Resize as necessary
  - Fit as many as required on page
  - Type written description (or report) if needed
  - Print document to any printer (not necessarily postscript printer)

MATLAB Help and Online Tutorial



Web-based MATLAB Help & Documentation

MATLAB Environment (1)

- **Managing Commands and Functions**
  - **addpath** Add directories to MATLAB's search path
  - **help** Online help for MATLAB functions and M-files
  - **path** Control MATLAB's directory search path
- **Managing Variables and the Workspace**
  - **clear** Remove items from memory
  - **length** Length of vector
  - **load** Retrieve variables from disk
  - **save** Save workspace variables on disk
  - **size** Array dimensions
  - **who, whos** List directory of variables in memory

MATLAB Environment (2)

- **Working with Files and the Operating Environment**
  - **cd** Change working directory
  - **delete** Delete files and graphics objects
  - **diary** Save session in a disk file
  - **dir** Directory listing
  - **edit** Edit an M-file
  - **!** Execute operating system command