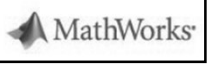
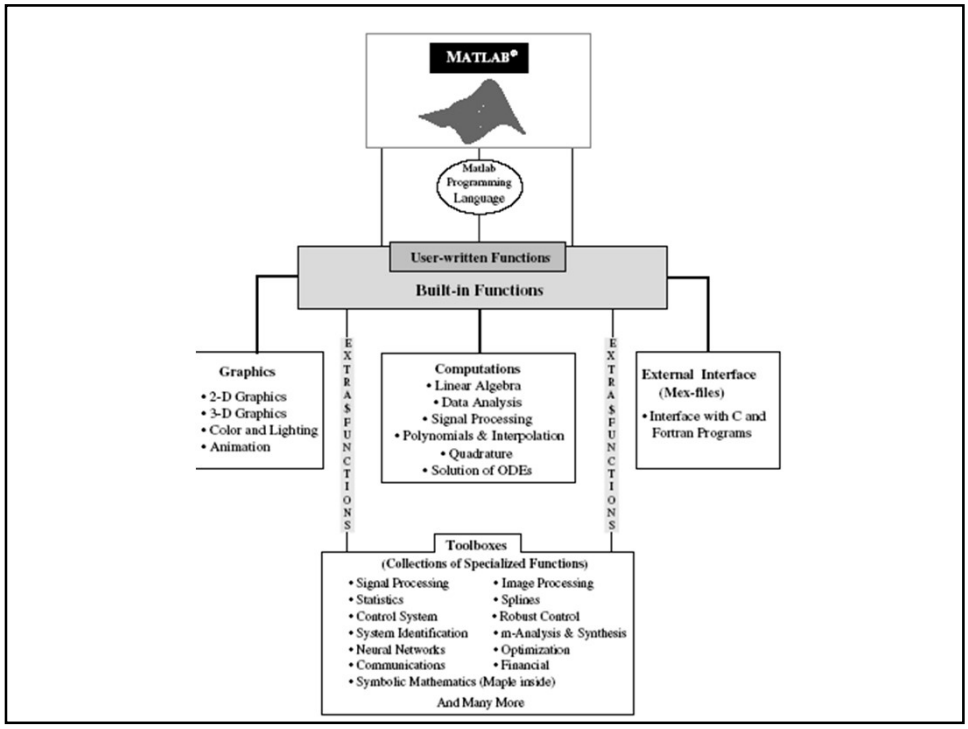
MATLAB

The Language of Technical Computing…

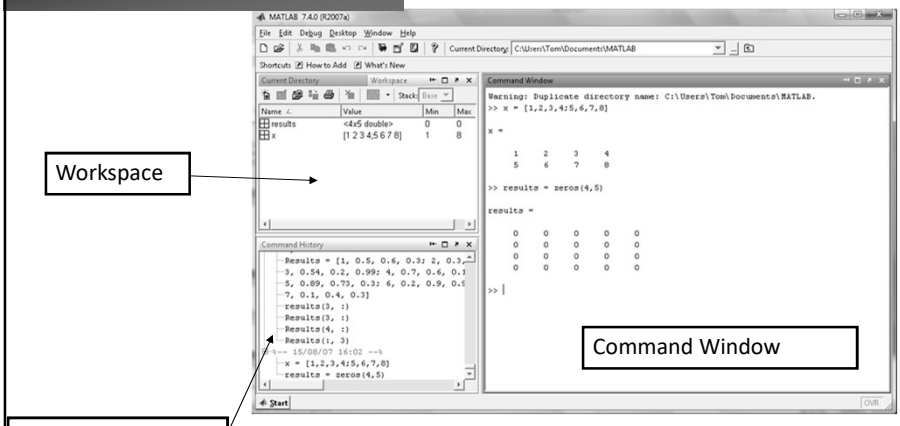


About MATLAB…

- MATLAB is a high-performance language for technical computing.
- The name **MATLAB** stands for **MAT**trix **LAB**oratory.
- It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.
- The basic building block of MATLAB is the Matrix.
- The fundamental data type is Array.



MATLAB Windows



Window	Purpose
Workspace Window	Provides information about the variables that are stored.
Current Folder Window	Shows the files in the current folder.

Table 1-1: MATLAB windows

Window	Purpose
Command Window	Main window, enters variables, runs programs.
Figure Window	Contains output from graphic commands.
Editor Window	Creates and debugs script and function files.
Help Window	Provides help information.
Command History Window	Logs commands entered in the Command Window.

4

Basics of MATLAB

Character set

- Alphabets (Case-sensitive)
- Numerals
- Special characters

Constant or Variables

- Numeric : Integer, Real, Complex
- Character : single, string
- Special constant : pi, i, j,

➢ No need to declare variables

~~int a;~~
~~double b;~~
~~float c;~~

Basics of MATLAB

Operators

- Arithmetic operators : +,-,/,*,^

+	addition
−	subtraction
*	multiplication
/	division
^ (caret)	exponentiation

- Relational operators : <,<=, >, >=, ==

<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
==	equal
~=	not equal.

- Logical operators : &, !, ~

&	logical AND
|	logical OR
~	logical complement (NOT)
xor	exclusive OR

Operator Precedence

1. Inner most ()
2. ^
3. /, *
4. +, -

Precedence	Mathematical Operation
First	Parentheses. For nested parentheses, the innermost are executed first.
Second	Exponentiation.
Third	Multiplication, division (equal precedence).
Fourth	Addition and subtraction.

Some Basic Mathematical Functions

Function	Description	Example

`sqrt(x)`	Square root.	`>> sqrt(81)`
		`ans =`
		`9`

`nthroot(x,n)`	Real *n*th root of a real number *x*. (If *x* is negative *n* must be an odd integer.)	`>> nthroot(80,5)`
		`ans =`
		`2.4022`

`exp(x)`	Exponential (e^x).	`>> exp(5)`
		`ans =`
		`148.4132`

Function	Description	Example

`abs(x)`	Absolute value.	`>> abs(-24)`
		`ans =`
		`24`

`log(x)`	Natural logarithm.
	Base *e* logarithm (ln).
		`>> log(1000)`
		`ans =`
		`6.9078`

`log10(x)`	Base 10 logarithm.	`>> log10(1000)`
		`ans =`
		`3.0000`

`factorial(x)`	The factorial function *x*!
	(*x* must be a positive integer.)
		`>> factorial(5)`
		`ans =`
		`120`

Some Basic Mathematical Functions

Exponential functions

`exp`	Exponential. *Example:* `exp(A)` produces a matrix with elements $e^{(A_{ij})}$. So how do you compute e^A? See the next section.
`log`	Natural logarithm. *Example:* `log(A)` produces a matrix with elements $\ln(A_{ij})$.
`log10`	Base 10 logarithm. *Example:* `log10(A)` produces a matrix with elements $\log_{10}(A_{ij})$.
`sqrt`	Square root. *Example:* `sqrt(A)` produces a matrix with elements $\sqrt{A_{ij}}$.

Some Basic Mathematical Functions

Trigonometric functions

`sin`	Sine.	`sinh`	Hyperbolic sine.
`asin`	Inverse sine.	`asinh`	Inverse hyperbolic sine.
`cos`	Cosine.	`cosh`	Hyperbolic cosine.
`acos`	Inverse cosine.	`acosh`	Inverse hyperbolic cosine.
`tan`	Tangent.	`tanh`	Hyperbolic tangent.
`atan,atan2`	Inverse tangent.	`atanh`	Inverse hyperbolic tangent.
`sec`	Secant.	`sech`	Hyperbolic secant.
`asec`	Inverse secant.	`asech`	Inverse hyperbolic secant.
`csc`	Cosecant.	`csch`	Hyperbolic cosecant.

Function	Description	Example

`sin(x)`
`sind(x)`
	Sine of angle x (x in radians).
	Sine of angle x (x in degrees).
		`>> sin(pi/6)`
		`ans =`
		`0.5000`

`cos(x)`
`cosd(x)`
	Cosine of angle x (x in radians).
	Cosine of angle x (x in degrees).
		`>> cosd(30)`
		`ans =`
		`0.8660`

`tan(x)`
`tand(x)`
	Tangent of angle x (x in radians).
	Tangent of angle x (x in degrees).
		`>> tan(pi/6)`
		`ans =`
		`0.5774`

`cot(x)`
`cotd(x)`
	Cotangent of angle X (X in radians).
	Cotangent of angle X (X in degrees).
		`>> cotd(30)`
		`ans =`
		`1.7321`

Some Basic Mathematical Functions

Round-off functions

`fix`	Round towards 0. *Example:* `fix([-2.33 2.66])` = [−2 2].
`floor`	Round towards $-\infty$. *Example:* `floor([-2.33 2.66])` = [−3 2].
`ceil`	Round towards $+\infty$. *Example:* `ceil([-2.33 2.66])` = [−2 3].
`round`	Round towards the nearest integer. *Example:* `round([-2.33 2.66])` = [−2 3].
`rem`	Remainder after division. `rem(a,b)` is the same as `a - fix(a./b)`. *Example:* If `a=[-1.5 7], b=[2 3]`, then `rem(a,b) = [-1.5 1]`.
`sign`	Signum function. *Example:* `sign([-2.33 2.66])` = [−1 1].

Complex functions

`abs`	Absolute value. *Example:* `abs(A)` produces a matrix of absolute values $\|A_{ij}\|$.
`angle`	Phase angle. *Example:* `angle(A)` gives the phase angles of complex A.
`conj`	Complex conjugate. *Example:* `conj(A)` produces a matrix with elements $\bar{A}_{ij}$.
`imag`	Imaginary part. *Example:* `imag(A)` extracts the imaginary part of A.
`real`	Real part. *Example:* `real(A)` extracts the real part of A.

Examples

Arithmetic operations: Compute the following quantities:

- $\frac{2^5}{2^5-1}$ and compare with $(1-\frac{1}{2^5})^{-1}$.
- $3\frac{\sqrt{5}-1}{(\sqrt{5}+1)^2}-1$. The square root $\sqrt{x}$ can be calculated with the command `sqrt(x)` or `x^0.5`.
- Area $=\pi r^2$ with $r=\pi^{\frac{1}{3}}-1$. (π is `pi` in MATLAB.)

Command

```
2^5/(2^5-1)
3*(sqrt(5)-1)/(sqrt(5)+1)^2 - 1
area=pi*(pi^(1/3)-1)^2
```

Exponential and logarithms: The mathematical quantities e^x, $\ln x$, and $\log x$ are calculated with `exp(x)`, `log(x)`, and `log10(x)`, respectively. Calculate the following quantities:

- e^3, $\ln(e^3)$, $\log_{10}(e^3)$, and $\log_{10}(10^5)$.
- $e^{\pi\sqrt{163}}$.
- Solve $3^x=17$ for x and check the result. (The solution is $x=\frac{\ln 17}{\ln 3}$. You can verify the result by direct substitution.)

Command

```
exp(3)
log(exp(3))
log10(exp(3))
log10(10^5)
exp(pi*sqrt(163))
x=log(17)/log(3)
```

Calculate the following quantities:

- $\sin\frac{\pi}{6}$, $\cos\pi$, and $\tan\frac{\pi}{2}$.
- $\sin^2\frac{\pi}{6}+\cos^2\frac{\pi}{6}$. (Typing `sin^2(x)` for $\sin^2 x$ will produce an error).
- $y=\cosh^2 x-\sinh^2 x$, with $x=32\pi$.

Command

```
sin(pi/6)
cos(pi)
tan(pi/2)
(sin(pi/6))^2+(cos(pi/6))^2
x=32*pi; y=(cosh(x))^2-(sinh(x))^2
```

The semicolon (;):

When a command is typed in the Command Window and the **Enter** key is pressed, the command is executed. Any output that the command generates is displayed in the Command Window. If a semicolon (;) is typed at the end of a command, the output of the command is not displayed. Typing a semicolon is useful when the result is obvious or known, or when the output is very large.

If several commands are typed in the same line, the output from any of the commands will not be displayed if a semicolon instead of a comma is typed between the commands.

Typing %:

When the symbol % (percent) is typed at the beginning of a line, the line is designated as a comment. This means that when the **Enter** key is pressed the line is not executed. The % character followed by text (comment) can also be typed after a command (in the same line). This has no effect on the execution of the command.

The `clc` command:

The `clc` command (type `clc` and press **Enter**) clears the Command Window. After typing in the Command Window for a while, the display may become very long. Once the `clc` command is executed, a clear window is displayed. The command does not change anything that was done before. For example, if some variables were defined previously (see Section 1.6), they still exist and can be used. The up-arrow key can also be used to recall commands that were typed before.

Command	Description	Example

`format short`	Fixed-point with 4 decimal digits for: $0.001 \le number \le 1000$ Otherwise display format `short e`.	`>> 290/7`
		`ans =`
		`41.4286`

`format long`	Fixed-point with 15 decimal digits for: $0.001 \le number \le 100$ Otherwise display format `long e`.	`>> 290/7`
		`ans =`
		`41.428571428571431`

`format short e`	Scientific notation with 4 decimal digits.	`>> 290/7`
		`ans =`
		`4.1429e+001`

`format long e`	Scientific notation with 15 decimal digits.	`>> 290/7`
		`ans =`
		`4.142857142857143e+001`

`format short g`	Best of 5-digit fixed or floating point.	`>> 290/7`
		`ans =`
		`41.429`

`format long g`	Best of 15-digit fixed or floating point.	`>> 290/7`
		`ans =`
		`41.4285714285714`

`format bank`	Two decimal digits.	`>> 290/7`
		`ans =`
		`41.43`

`format compact`	Eliminates blank lines to allow more lines with information displayed on the screen.	
`format loose`	Adds blank lines (opposite of `compact`).	

Vectors & Matrices

Vector

- Row vector: (1xn) matrix
- Column vector: (nx1) matrix

Creation of evenly spaced elements of row vectors

rv = initial value:step size:final value [Ex: r=1:2:15;]
step size>0 when initial value<final value
step size<0 when initial value>final value
Step size may be integer or real no.
rv = initial value:final value [Ex: r=1:15 is same as r=1:1:15;]

rv = linspace(initial value, final value, total no. of elements)
Ex: r=linspace(1,15,8);

The Assignment Operator

Variable_name = A numerical value, or a computable expression

Vectors & Matrices

Functions related to Vectors

```
r1 = r'
r2=sum(r)
r3=mean(r)
r4=length(r)
r5=max(r)
r5=min(r)
r6=prod(r)
r7=sign(r)
```

Vectors & Matrices

Entering elements of a matrix

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

Matrix Indics/subscripts

```
A(3,1) % elements in 3rd row & 1st column
A(2:4, 1:3)
A(:, 2:3)
A(1:3,4)
A(:, end)
[m n]=size(A)
```

Generation of special matrices

```
A=zeros(3,4);
A=ones(3,4);
A=eye(3,4);
A=rand(3,4); A=rand(3);
A=rands(3,4);
```

Vectors & Matrices

Entering elements of a matrix

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

Functions related to Matrices

```
D=det(A);
M=A';
I=inv(A);
R=rank(A);
T=trace(A);
E=eig(A)
```

Array Operations

.*	element-by-element multiplication
./	element-by-element left division
.\	element-by-element right division
.^	element-by-element exponentiation

```
S1=A+3;
S2=A-3;
S3=A.*B;
S4=A./B;
```

```
S5=A.^2;
S6=A.\B;
```

Matrix Operations

```
S1=A+B;
S2=A-B;
S3=A*B;
S4=A/B;
```

```
S5=A^2;
S5=A*A;
S6=A\B;
S6=inv(A)*B
```

Solving Systems of Linear Equations

$5x_1+2x_2+3x_3=2$

$-4x_1+7x_2+5x_3=1$

$9x_1-8x_2+4.5x_3=-3$

```
X=A\B;
X=inv(A)*B
```

Input-Output Statements

Input function

```
n =input(‘  ’);
n =input(‘enter any number’);
c =input(‘enter character string’, ‘s’);
```

Input-Output Statements

Output function

Using the `fprintf` command to display a mix of text and numerical data:

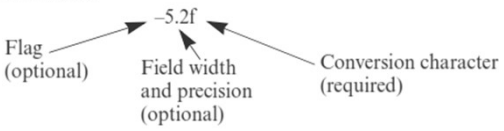
```
fprintf('MESA-NITD');
fprintf('value of n is %g\n', n)
fprintf('a1= %g, a2=%g\n', a1,
```

```
fprintf('text as string %-5.2f additional text',
                                    variable_name)
```

- The `%` sign marks the spot where the number is inserted within the text.
- Formatting elements (define the format of the number).
- The name of the variable whose value is displayed.

```
disp('MESA-NITD');
disp(variable_name);
```

The formatting elements are:



Character used for flag	Description
– (minus sign)	Left-justifies the number within the field.
+ (plus sign)	Prints a sign character (+ or –) in front of the number.
0 (zero)	Adds zeros if the number is shorter than the field.

```
fprintf('An average of %f points was scored in the three games.',ave_points)
```

- Text
- `%` marks the position of the number.
- Additional text.
- The name of the variable whose value is displayed.

e	Exponential notation using lowercase e (e.g., 1.709098e+001).
E	Exponential notation using uppercase E (e.g., 1.709098E+001).
f	Fixed-point notation (e.g., 17.090980).
g	The shorter of e or f notations.
G	The shorter of E or f notations.
i	Integer.

```
fprintf('..text...%g...%g...%f...',variable1,variable2,variable3)
```

```
% This program calculates the distance a projectile flies,
% given its initial velocity and the angle at which it is shot.
% the fprintf command is used to display a mix of text and num-
bers.

v=1584;  % Initial velocity (km/h)
theta=30;  % Angle (degrees)
vms=v*1000/3600;
t=vms*sind(30)/9.81;
d=vms*cosd(30)*2*t/1000;
fprintf('A projectile shot at %3.2f degrees with a velocity
of %4.2f km/h will travel a distance of %g km.\n',theta,v,d)
```

Changing velocity units to m/s.

Calculating the time to highest point.

Calculating max distance.

Command Window is:

```
>> Chapter4Example7
A projectile shot at 30.00 degrees with a velocity of
1584.00 km/h will travel a distance of 17.091 km.
>>
```

For example, the script file below creates a 2×5 matrix `T` in which the first row contains the numbers 1 through 5, and the second row shows the corresponding square roots.

```
x=1:5;
y=sqrt(x);
T=[x; y]
fprintf('If the number is: %i, its square root is: %f\n',T)
```

Create a vector x.

Create a vector y.

Create 2×5 matrix T, first row is x, second row is y.

The `fprintf` command displays two numbers from `T` in every line.

Loop, Branches & Control Flow

Relational operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
~=	Not Equal to

`if` conditional expression consisting of relational and/or logical operators.

Examples:

```
if a < b
if c >= 5
if a == b
if a ~= 0
if (d<h)&(x>7)
if (x~=13)|(y<0)
```

All the variables must have assigned values.

Loop, Branches & Control Flow

Flowchart


```
.....
..... MATLAB program.
.....
if conditional expression
    .....
    .....
    .....
end
.....
..... MATLAB program.
.....
```

A group of MATLAB commands.

If-end

Syntax

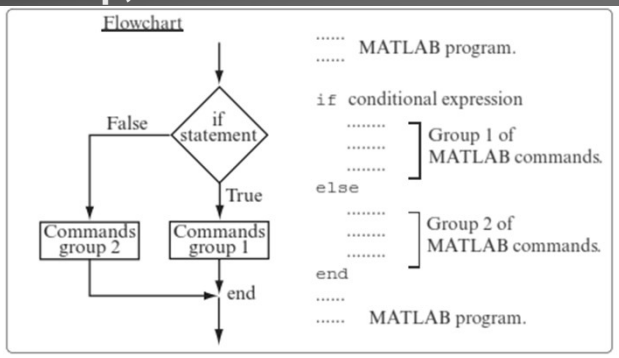
if *condition*

statement 1;

statement 2;

end

Loop, Branches & Control Flow

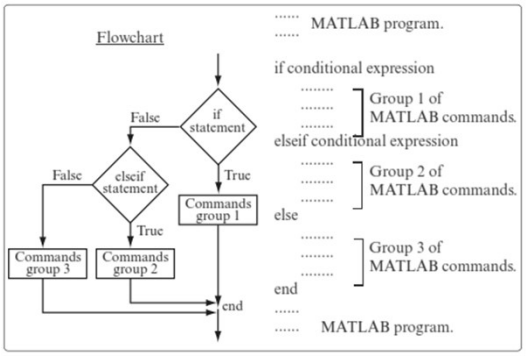


If-else-end

Syntax

if *condition*
  statement 1;
  statement 2;
else
  statement 3;
  statement 4;
end

Loop, Branches & Control Flow



If-else-if-else-end

Syntax

if *condition 1*
  statement 1;
else
  if *condition 2*
    statement 3;
  else
    statement 4;
  end
end

Loop, Branches & Control Flow

- if-else branching
- *for* loop
- *while* loop

If-end

Syntax

```
if condition
    statement 1;
    statement 2;
end
```

If-else-end

Syntax

```
if condition
    statement 1;
    statement 2;
else
    statement 3;
    statement 4;
end
```

If-else-if-else-end

Syntax

```
if condition 1
    statement 1;
else
    if condition 2
        statement 3;
    else
        statement 4;
    end
end
```

Loop, Branches & Control Flow

for *loop*

Syntax

```
for variable=expression
    statement 1;
    statement 2;
end
```

```
for m=1:100
    num = 1/(m+1)
end

for n=100:-2:0, k = 1/(exp(n)), end
```

while *loop*

Syntax

```
while condition
    statement 1;
    statement 2;
end
```

```
v = 1;   num = 1; i=1;
while num < 10000
      num = 2^i;
      v = [v; num];
      i = i + 1;
end
```

for loop	while loop
is used to repeat statements for a fixed number of times	is used to execute statements for an indefinite number of times until condition is met

Loop, Branches & Control Flow

for *loop*

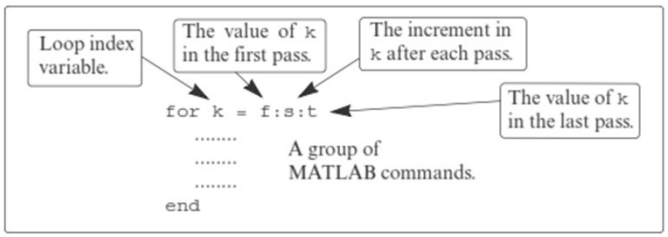
Syntax

for *variable=expression*

statement 1;

statement 2;

end



for *loop*

Use a `for-end` loop in a script file to calculate the sum of the first n terms of the series: $\sum_{k=1}^{n} \frac{(-1)^k k}{2^k}$. Execute the script file for $n = 4$ and $n = 20$.

```
n=input('Enter the number of terms ' );
S=0;
for k=1:n
    S=S+(-1)^k*k/2^k;
end
fprintf('The sum of the series is: %f',S)
```

Setting the sum to zero.

`for-end` loop.

In each pass one element of the series is calculated and is added to the sum of the elements from the previous passes.

Plot

```
r = input('Enter the radius of the circle: ')
theta = linspace(0,2*pi,100);      % create vector theta
x = r*cos(theta);                  % generate x-coordinates
y = r*sin(theta);                  % generate y-coordinates
plot(x,y);                         % plot the circle
axis('equal');                     % set equal scale on axes
title('Circle of given radius r') % put a title
```

A simple sine plot: Plot $y = \sin x$, $0 \le x \le 2\pi$, taking 100 linearly spaced points in the given interval. Label the axes and put 'Plot created

```
x=linspace(0,2*pi,100);
plot(x,sin(x))
xlabel('x'), ylabel('sin(x)')
```

On-line help

`help`	lists topics on which help is available
`helpwin`	opens the interactive help window
`helpdesk`	opens the web browser based help facility
`help` *topic*	provides help on *topic*
`lookfor` *string*	lists help topics containing *string*
`demo`	runs the demo program

Workspace information

`who`	lists variables currently in the workspace
`whos`	lists variables currently in the workspace with their size
`what`	lists `m-`, `mat-`, and `mex-`files on the disk
`clear`	clears the workspace, all variables are removed
`clear x y z`	clears only variables x, y and z
`clear all`	clears all variables and functions from workspace
`mlock` *fun*	locks function *fun* so that `clear` cannot remove it
`munlock` *fun*	unlocks function *fun* so that `clear` can remove it
`clc`	clears command window, command history is lost
`home`	same as `clc`
`clf`	clears figure window

Termination

`^c` (Control-c)	local abort, kills the current command execution
`quit`	quits MATLAB
`exit`	same as `quit`

Output format: Though computations inside MATLAB are performed using double precision, the appearance of floating point numbers on the screen is controlled by the output `format` in use. There are several different screen output formats. The following table shows the printed value of 10π in 7 different formats.

```
format short        31.4159
format short e      3.1416e+001
format long         31.41592653589793
format long e       3.141592653589793e+001
format short g      31.416
format long g       31.4159265358979
format hex          403f6a7a2955385e
format rat          3550/113
format bank         31.42
```

35