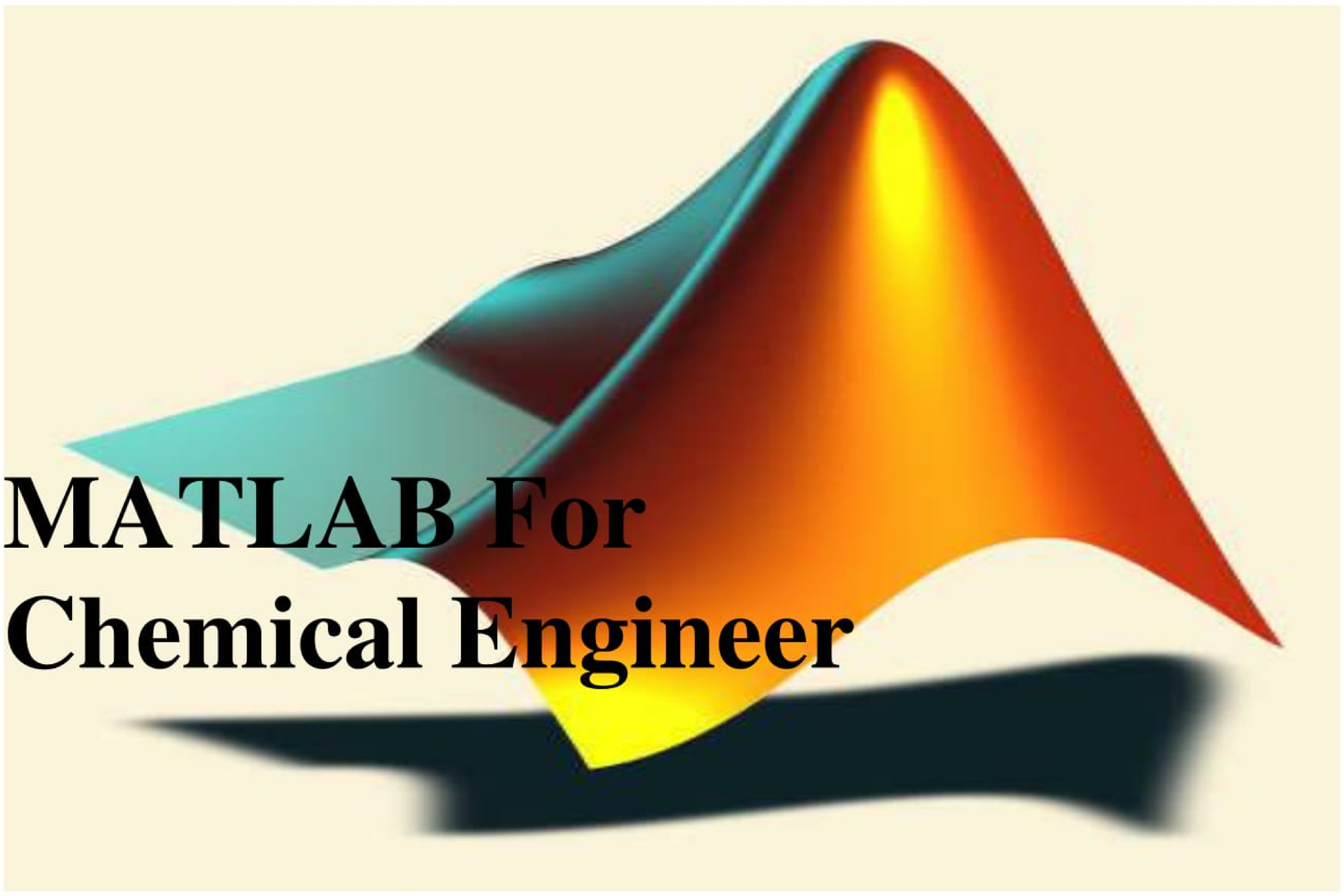


MATLAB For Chemical Engineer

Introduction to MATLAB

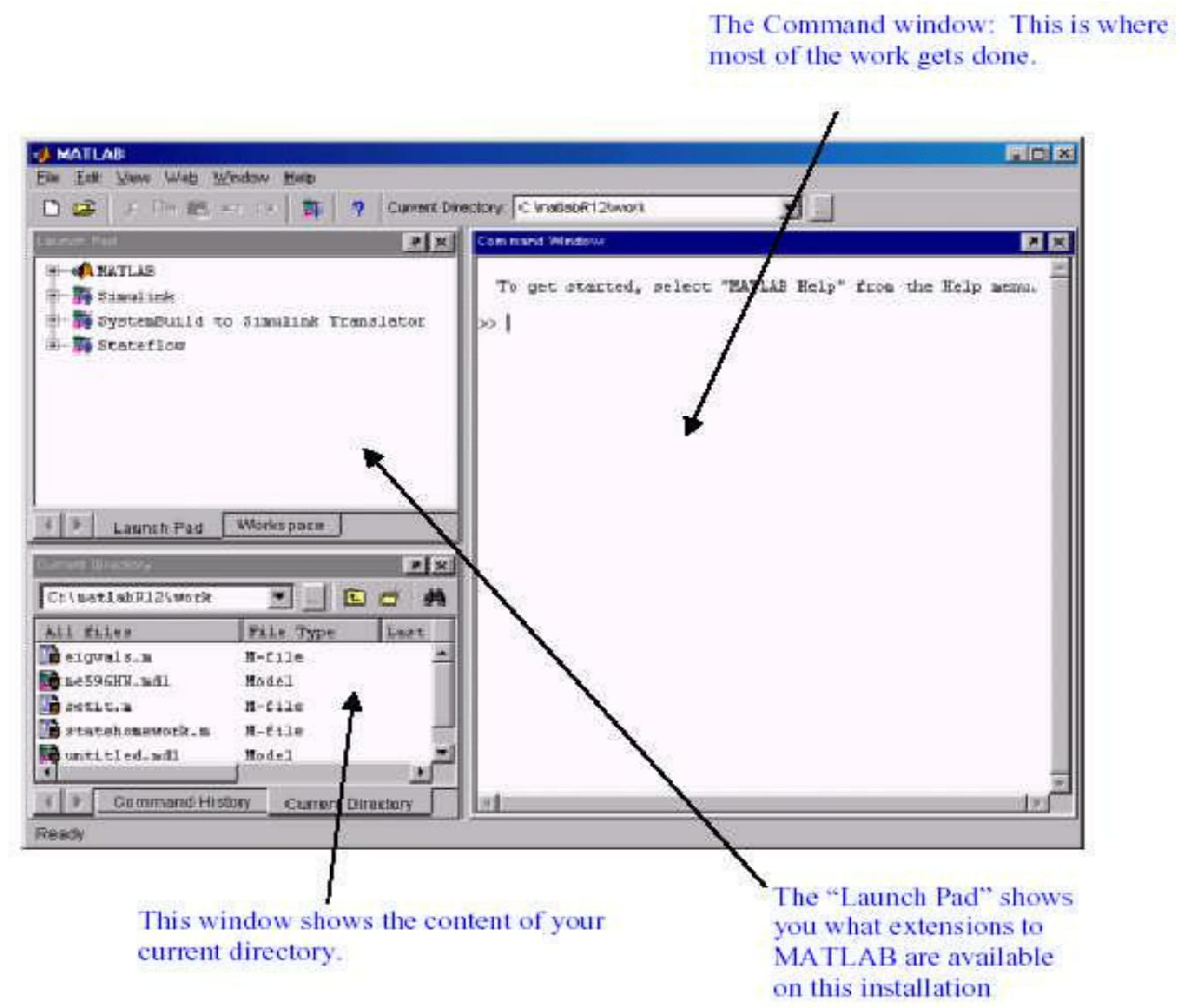
MATLAB (**MAT**rix**LAB**oratory) is an interactive software system for numerical computations and graphics. As the name suggests, MATLAB is especially designed for matrix computations, solving systems of linear equations, computing eigenvalues and eigenvectors, factorization of matrices, and so much.

In addition, it has a variety of graphical capabilities, and can be extended through programs written in its own programming language. Many such programs come with the system; these extend MATLAB's capabilities to a wide range of applications, like solution of nonlinear systems of equations, integration of ordinary and partial differential equations, and many others.

MATLAB widely used in the engineering. It has many features and it could take years to learn all of its capabilities. However, it's basic functionality is quite easy to learn and in the course of the next few weeks we will be learning how to manipulate and graph data in MATLAB as well as writing simple programs to manipulate data. It offers a powerful programming language, excellent graphics, and a wide range of expert knowledge.

1. Getting Started

Start Matlab by double clicking the icon on the desktop, or from the start menu. To UseMatlab you can simply enter commands after the prompt (the >> is the Matlab prompt). Figure 1 below shows the default frame with the three standard Matlab windows.



The Command window: This is where most of the work gets done.

```
To get started, select "MATLAB Help" from the Help menu.
>> |
```

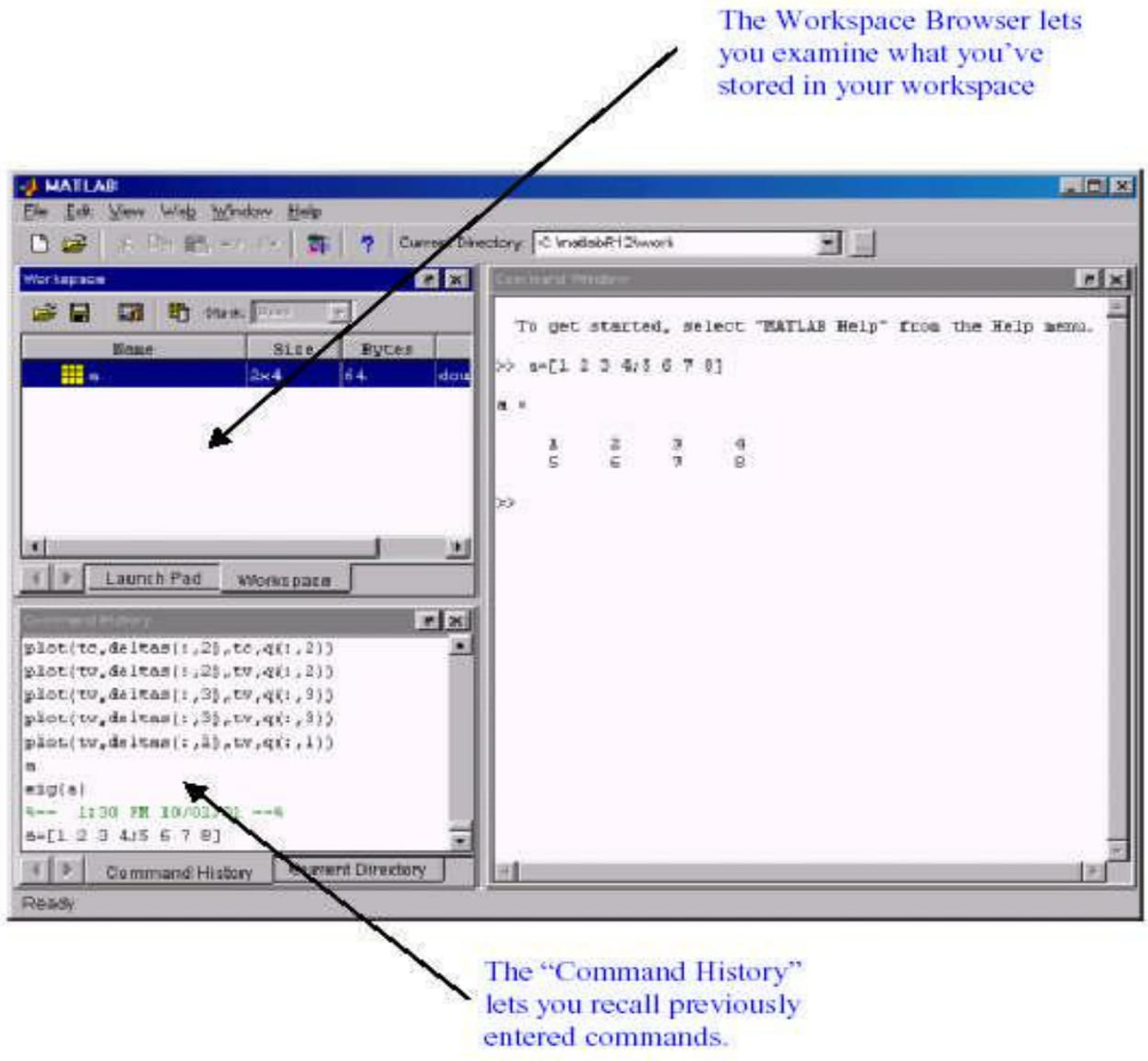
This window shows the content of your current directory.

The "Launch Pad" shows you what extensions to MATLAB are available on this installation

Figure 1: The MATLAB Window

1.1 Alternate windows:

The smaller of the two windows is alternate windows that can be accessed by clicking on the tabs. Figure 2 shows the alternate windows and describes their functions.


The Workspace Browser lets you examine what you've stored in your workspace
MATLAB
File Edit View Web Window Help
Current Directory: C:\matlabR12\work
Workspace
Name Size Bytes
a 2x4 64 dou
Launch Pad
Workspace
Command History
Current Directory
Ready
The "Command History" lets you recall previously entered commands.

```
To get started, select "MATLAB Help" from the Help menu.

>> a=[1 2 3 4;5 6 7 8]

a =

     1     2     3     4
     5     6     7     8

>>
```

```
plot(tc,deltas(:,2),tc,q(:,2))
plot(tv,deltas(:,2),tv,q(:,2))
plot(tv,deltas(:,3),tv,q(:,3))
plot(tv,deltas(:,3),tv,q(:,3))
plot(tv,deltas(:,1),tv,q(:,1))
a
eig(a)
%-- 1:30 PM 10/03/01 --%
a=[1 2 3 4;5 6 7 8]
```

Figure 2: Alternate windows in the default frame

<u>1.2 The command window:</u>

The command window is the active window immediately appears after launching Matlab. One enters Matlab commands after the ">>" prompt and presses enter to execute the command. To recall the last commands entered, simply press the up or down arrows; one can edit the commands before executing them. Multiple commands may be entered on one line separated by commas. Separating commands by a semi-colon suppresses output to the command window.

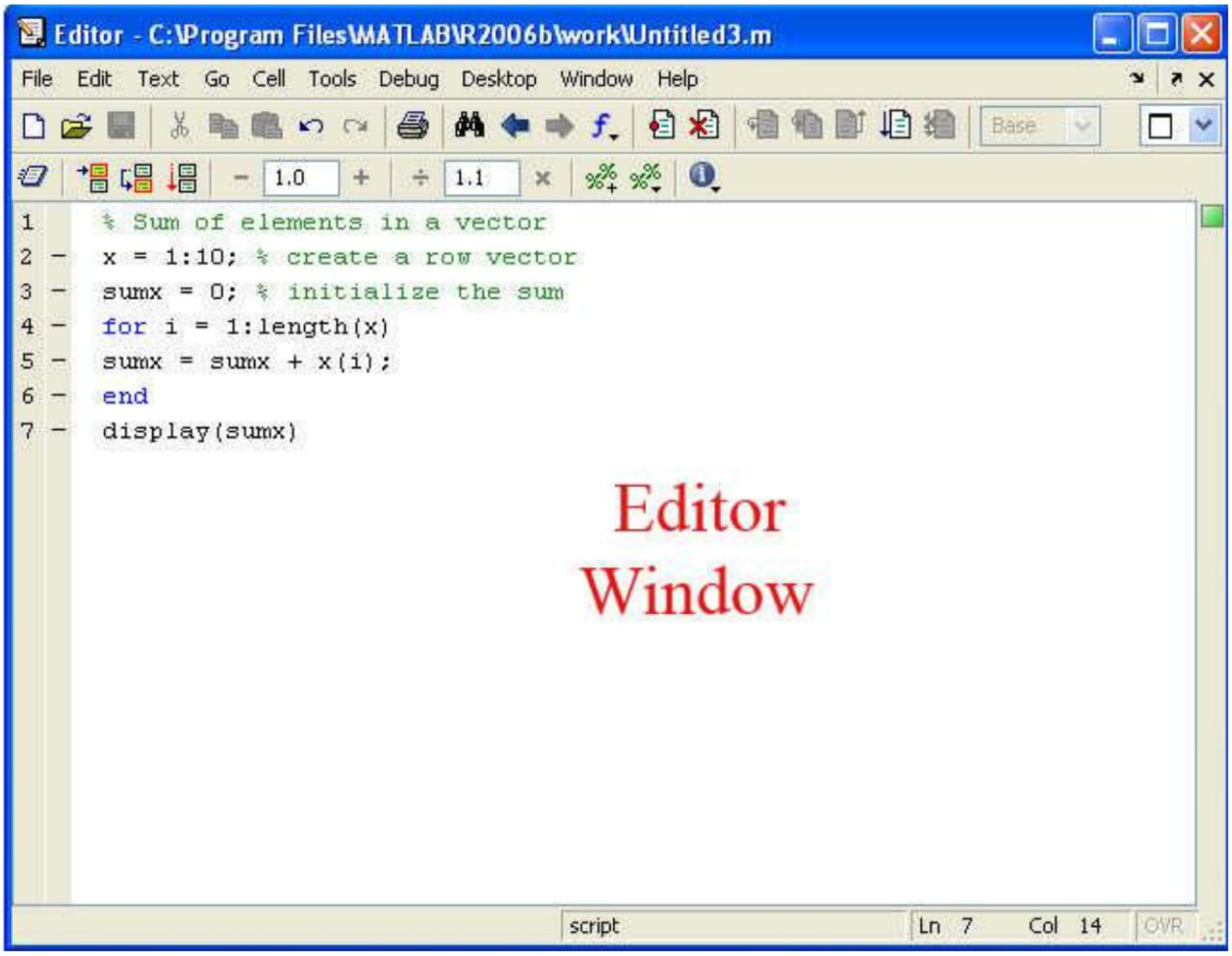
This window allows a user to enter simple commands. To perform a simple computations type a command and next press the Enter or Return key. For instance

```
>> s = 1 + 2
s =
3
```

Note that the results of these computations are saved in variables whose names are chosen by the user. If you need to obtain their values again, type their names and pressingEnter key. If you type again:

```
>>s
s =
3
```

Only for short computations it is useful to execute Matlab straightaway from the command line. The Editor Window is a word processor specifically designed for MATLAB commands. Files that written in this window are called the m-files. Another way to do calculations in MATLAB is to create an m-file with a series of commands and then to run some or all of the commands in that file. To create an m-file, click file, new and then m-files. The same statements that are entered in the command window can also be used in an m file. You can also copy a command you try out in the command window into an m file by using the copy and paste functions on the computer. Save the file under a name that ends in .m by clicking file and using "save as" icon. (See Figure 3)


Editor - C:\Program Files\MATLAB\R2006b\work\Untitled3.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Editor
Window
script
Ln 7 Col 14 OVR

```
% Sum of elements in a vector
x = 1:10; % create a row vector
sumx = 0; % initialize the sum
for i = 1:length(x)
sumx = sumx + x(i);
end
display(sumx)
```

1.3 Function Files

Function files are a special kind of script file (M-file) that allow you to define your Own functions for use during a Matlab session. You might think of them as subroutinesthat can be called from within a script, or even called directly from the command line.Many of the "built-in" functions in Matlab are actually stored as M-files as part of theMatlab package. Function files are created and edited in identically the same manner asthe script files.

2. Numbers, Arithmetic Operations and Special Characters

There are three kinds of numbers used in MATLAB:

• integers

• real numbers

• complex numbers

In addition to these, MATLAB has three variables representing non-numbers: (-Inf ,Inf , Nan) .

The –Inf and Inf are the negative and positive infinity respectively. Infinity isgenerated by overflow or by the operation of dividing by zero. The Nan stands for Not-ANumberand it is obtained as a result of the mathematically undefined operations such as0.0/0.0.

The list of basic arithmetic operations in MATLAB includes six operations:

+ : addition

- : subtraction

* : multiplication

/ : right division

\ : left division

^ : power

One can use Matlab like a calculator without specifying variables. Matlab has all the standard mathematical operations. Try type:

```
>> 2+3
```

Matlab returns the answer:

```
ans=
5
>> 5^3
ans=
125
>>3.89*4.1
ans =
15.9490
>>99.3-25
ans =
74.3000
```

```
>> 3*(23+14.7-4/6)/3.5
ans=
31.7429
```

The result of these operations is assigned to a default variable called ans and displayed. Adding a semicolon to the end of the operation suppresses the output; try it out!

```
>>25*3;
```

Also type:

```
>> 1/0
Warning: Divide by zero.
ans =
Inf
>>Inf/Inf
ans =
Nan
```

3. Relational operations

Relational operators perform element-by-element comparisons between two numbers as following meaning;

A < B Less than

A > B Greater than

A <= B Less than or equal

A >= B Greater than or equal

A == B Equal

A ~= B Not equal

Further, there is a menu of special characters that have specific uses.

Logical AND &

Logical OR |

Logical NOT ~

Colon :

Subscripting ( )

Brackets [ ]

Decimal point .

Continuation ...

Separator ,

Semicolon ; (suppresses the output of a calculation)

Assignment =

Quote ' *statement* '

Transpose '

Comment %

Note: anything after % is a comment and will be ignored by Matlab.

4. Variables

Variable names may be up to 19 characters long. Names must begin with a letter but may be followed by any combination of letters, digits or underscores. Variables are storage locations in the computer that are associated with an alphanumeric name. To assign a value to a variable in MATLAB simply type the name of the variable, followed by the assignment operator, =, followed by the value.

As an example, this is one way to compute 2+2:

```
>> a = 2
a =
2
>> b = 2
b =
2
>> c = a + b
c =
4
```

It is often annoying to have Matlab always print out the value of each variable. To avoid this, put a semicolon after the commands:

```
>> a = 2;
>> b = 2;
>> c = a + b;
>>c
c =
4
```

Only the final line produces output. Semicolons can also be used to string together more than one command on the same line:

```
>> a = 2; b = 2; c = a + b; c
c =
4
```

Of course Matlab will also allow more complicated operations:

```
>> a = 2;
>> b = -12;
>> c = 16;
>> qu1 = (-b + sqrt(b^2 - 4*a*c)) / (2*a)
qu1 =
4
```

Understand that 'Matlab' is "case sensitive", that is, it treats the name 'C' and 'c' as two different variables. Similarly, 'MID' and 'Mid' are treated as two different variables. Assign two different values to the variables and print them out by entering their names separated by a comma.

```
>>var=1.2
var =
1.2000
>>Var=-5.1
Var =
-5.1000
>>var, Var
var =
1.2000
Var =
-5.1000
```

4.1 Predefined variables

There are several predefined variables which can be used at any time, in the same Manner as user defined variables (*ans*, *pi*, *j*, *eps*):

I: sqrt(-1)

j: sqrt(-1)

pi: 3.1416...

For example,

```
>>pi
ans =
3.1416
>>eps
eps =
2.2204e-016
>>j
ans =
0 + 1.0000i
>>y= 2*(1+4*j)
```

yields:

```
y=
2.0000 + 8.0000i
```

5. Reporting format

By default MATLAB returns numerical expressions as decimals with 4 digits. The format function is used to change the format of the output. Type format rat to have MATLAB return rational expressions.

```
>>format rat
>> 5.1-3.3
ans =
9/5
```

To eliminate the extra spacing type format compact.

```
>>format compact
>> 5*7
ans =
35
```

Now type

```
>>format long
>> 3*(23+14.7-4/6)/3.5
ans=
31.74285714285715
>>format short e
>> 3*(23+14.7-4/6)/3.5
ans=
3.1743e+01
```

Note that the answer is accurate to four decimal places. Now type

```
>>format long e
ans=
3.174285714285715e+01
>>format short
ans=
31.7429
```

Note: format short will return the numerical expression to default . Also, the format of reporting does not change the accuracy of the calculations only the appearance of the answer on screen.

6. Mathematical functions

The following functions are defined in MATLAB

6.1. Trigonometric functions

Those known to Matlab are sin, cos, tan and their arguments should be in radians.

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( ) - Inverse tangent.

atanh( ) - Inverse hyperbolic tangent.

sec( ) - Secant.

sech( ) - Hyperbolic secant.

asec( ) - Inverse secant.

asech( ) - Inverse hyperbolic secant.

csc( ) - Cosecant.

csch( ) - Hyperbolic cosecant.

acsc( ) - Inverse cosecant.

acsch( ) - Inverse hyperbolic cosecant.

cot( ) - Cotangent.

coth( ) - Hyperbolic cotangent.

acot( ) - Inverse cotangent.

acoth( ) - Inverse hyperbolic cotangent.

```
>> x =5*cos(pi/6), y = 5*sin(pi/6)
x =
4.3301
y =
2.5000
```

The inverse of trigonometric functions are called asin, acos, atan (as opposed to the Usual arc sin or sin 1 etc.).

The result is in radians.

```
>>acos(x/5), asin(y/5)
ans =
0.5236
ans =
0.5236
```

Note: Matlab uses radian scale to calculate trigonometric functions**.** In other words, sin(90) is not equal to 1, but sin(pi/2) is 1.

6.2. Exponential

These include sqrt, exp, log, log10

exp( ) - Exponential.

log( ) - Natural logarithm.

log10( ) - Common (base 10) logarithm.

sqrt( ) - Square root.

abs( ) - Absolute value.

Note: log( ) is ln in Matlab. To get logarithm in base 10, you must write log10( ).

```
>>exp(log(9)), log(exp(9))
ans =
9
ans =
9
```

Most common functions are available to Matlab

```
>>A=abs(-5), B=cos(3), C=exp(2), D=sqrt(4), E=log(40)
A =
5
B =
-0.9900
C =
7.3891
D =
2
E =
3.6889
```

6.3. Complex Number Functions

conj( ) - Complex conjugate.

Imag( ) - Complex imaginary part.

Real( ) - Complex real part.

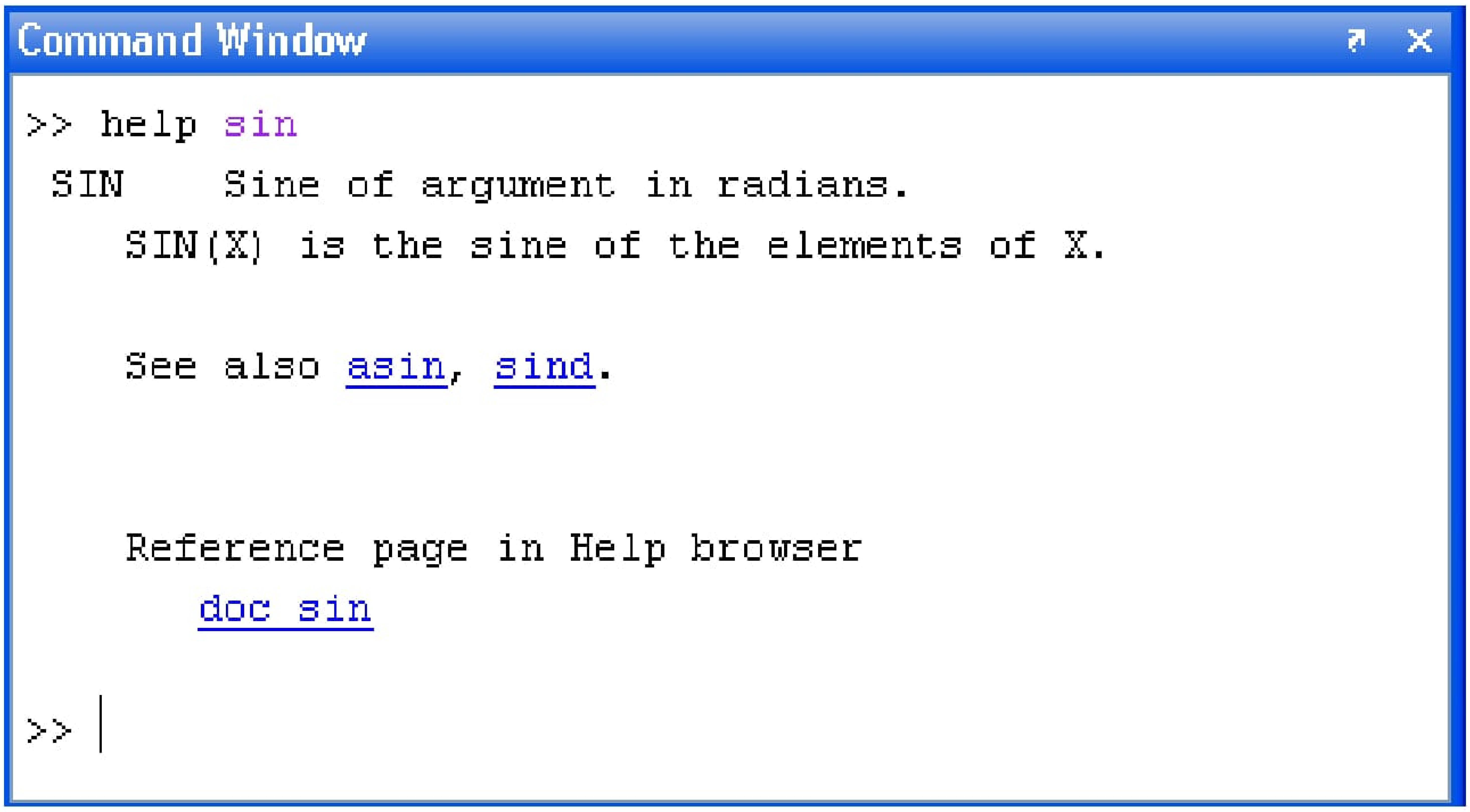
```
>>A=2+4*i, B=conj(A), C=Imag(A), D=real(A)
A =
2.0000 + 4.0000i
B =
2.0000 - 4.0000i

C =
4
D =
2
```

7. Help

MATLAB is a huge package. You can't learn everything about it at once, or always remember how you have done things before. It is essential that you learn how to teachyourself more using the online help.

If you need quick help on the syntax of a command, use help. For example, **help Plot**tells you all the ways in which you can use the plot command. (Of course, you haveto know already the name of the command you want.)

>> help sin



```
>> help sin
 SIN    Sine of argument in radians.
    SIN(X) is the sine of the elements of X.

    See also asin, sind.


    Reference page in Help browser
       doc sin

>> 
```

8. Closing Matlab

To close MATLAB type exit in the command window and next press Enter or Return key. A second way to close your current MATLAB session is to select File in the MATLAB's toolbar and next click on Exit MATLAB option. All unsaved information residing in the MATLAB.

Workspace will be lost. You can also exit by typing:

>> quit

or

>>exit

To terminate a running Matlab command you may use **[Ctrl]+[c]** (Press both the Ctrl button and the c button simultaneously).

9. Common commands

whos: gives a list of Matlab variables stored in the memory

clear : clears the memory

clear A : clears variable named A from the memory

clc: clears the command window

clf: clears the graphical window

The following example show how to assign values to variables, x and y.

```
>> x=sin(pi/4), y=log(2)
x =
0.7071
y =
0.6931
```

You can use who command to list the currently active variables. For the preceding session this results in

```
>>who
Your variables are:
ans x y
```

Use clear command to delete variables from computer memory

```
>>clear x
>>x
??? Undefined function or variable 'x'.
```

Exercise 1:

Write a program to calculate the vapor pressure of water according to Antoine equation: P_o=exp(A-B/(T+C))Where T is any given temperature in Kelvin and A, B, and C are Antoinecoefficients:

A=18.3036 B=3816.44 C= -46.13

Solution: Let temperature equal to 373.15 k, write the following code.

```
T=373.15;
A=18.3036;B=3816.44;C= -46.13;
Pw=exp(A-B/(T+C))
```

The result will be:

```
Pw =
759.9430
```

Note: you can use any variable name in your code.

Exercise 2:

Write a program to calculate the volumetric and mass flow rate of a liquid flowing in a pipe with a velocity equal to 0.5 m/s. Knowing that the diameter of this pipe is 0.1 m and the density of this liquid is 890 kg/m3 ?

Solution:

```
d=0.1;p=890;u=.5;
A=(pi/4)*d^2,Volflow=u*A,Massflow=Volflow*p
```

The result will be:

```
A =
0.0079
Volflow =
0.0039
Massflow =
3.4950
```

Practice Problems

1) Use Matlab as a calculator to calculate the results of each of the following commands:-

Command	Answer
7 + 8/2	
7+8\2	
(7+8)/2	
4 + 5/3 +2	
5^3/2	
5^(10/5)	
27^(1/3) + 32^0.2	
27^1/3 + 32^0.2	

2) Solve the following problems in command window

a) $\dfrac{37+8}{5+2^2}$

b) $\dfrac{6}{2}*3*4+\dfrac{2^5}{3+5}$

c) $(3+1)^2+\dfrac{8^{4/2}}{5+11}$

d) $3^2+2^2+\dfrac{8^4}{2*(5+11)}$

3) Define the variable x as x=6, then evaluate:

a) x^3+2x^2-11

b) $(x-2)^2-8$

c) $\dfrac{(x-2)^2}{4}-2$

4) Compute the reaction rate constant for a first-order reaction given by the Arrhenius law $k=A*e^{-E/RT}$, at a temperature T=500 K. Here the activation energy

is E=20 kcal/mol and the pre-exponential factor is $A=10^{13}s^{-1}$. The ideal gas constant is R=1.987 cal/mol K.

ALGEBRA

1. Symbolic Toolbox

One of the most powerful tools that can be used in Matlab is the “Symbolic Toolbox”. To use Matlab’s facility for doing symbolic mathematics, it is necessary to declare the variables to be “symbolic”. The best way to do that is to use the **syms** declaration statement:

```
>>syms a b x y;
>>c = 5;
>>E = a*x^2 + b*x + c;
```

The **syms**statement makes all the variables listed with it into symbolic variables and gives each of the variables a value that is equal to its own name. Thus, the value of **a** is**a**, the value of **b** is **b**, etc. The variable **E** is now symbolic because it was assigned to be equal to an expression that contained one or more symbolic variables. Its value is **a*x^2 + b*x + 5.**

For example, suppose that you need factor x^2-3x+2. Note that you must type 3*x for 3x. Then you type:

```
>>syms x
```

By syms you are declaring that x is a variable. Then type

```
>>factor(x^2-3*x+2)
ans =
(x-1)*(x-2)
```

To factor x2+2x+1, you write:

```
>>syms x
>>factor(x^2+2*x+1)
```

```
ans =
(x+1)^2
```

To factor the equation x2-y2;

```
>>syms x y
>>factor(x^2-y^2)
ans =
(x-y)*(x+y)
```

Expand command can be used to expand the terms of any power equation. Let's use expand command to expand the following equation (x2-y2)3.

```
>>expand((x^2-y^2)^3)
ans =
x^6-3*x^4*y^2+3*x^2*y^4-y^6
```

The **simplify** command is useful to simplify some equations like.

```
>>simplify((x^3-4*x)/(x^2+2*x))
ans =
x-2
```

2. Solving Equations

2.1 Algebraic Equations

By using Symbolic Toolbox, you can find solutions of algebraic equations with or without using numerical values. If you need to solve equations, you can use the commandsolve. For example, to find the solution of $x_3+x_2+x+1=0$ you write:

```
>>solve('x^3+x^2+x+1=0')
```

And Matlab give you the answer in the form

```
ans =
[ -1]
[ i]
[ -i]
```

That means the three solutions for the equation are 1, j, and –j.

```
>>x=solve('sin(x)+x=0.1')
x =
5.001042187833512e-2
```

In expressions with more than one variable, we can solve for one or more of the variables in terms of the others. Here we find the roots of the quadratic $ax2+bx+c$ in x interms of a, b and c. By default solve sets the given expression equal to zero if an equationis not given.

```
>> x=solve('a*x^2+b*x+c','x')
x =
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

You can solve an equation in two variables for one of them. For example:

```
>> y=solve('y^2+2*x*y+2*x^2+2*x+1=0', 'y')
y =
[-x+i*(x+1)]
[ -x-i*(x+1)]
```

You can solve more than one equation simultaneously. For example to find the value of x and y from the equations: $5x+10y=46$ and $28x+32y=32$, you write:

```
>> [x,y]=solve('5*x+10*y=46', '28*x+32*y=32')
```

And you get the following result:

```
x =
-48/5
y =
47/5
>> [x,y]=solve('log(x)+x*y=0', 'x*y+5*y=1')
x =
.8631121967939437
```

```
y =
.1705578823046945
```

To solve the system $x^2+ x+ y^2 = 2$ and $2x-y = 2$. We can type:

```
>> [x,y] = solve( 'x^2+ x+ y^2 = 2', '2*x-y = 2')
```

And get the solutions

```
x =
[ 2/5]
[ 1]
y =
[ -6/5]
[ 0]
```

This means that there are two points which are (2/5, -6/5) and (1, 0).

Now let's find the points of intersection of the circles $x^2+y^2=4$ and $(x-1)^2+(y-1)^2=1$.

```
>>[x,y]=solve('x^2+y^2=4','(x-1)^2+(y-1)^2=1')
x=
[ 5/4-1/4*7^(1/2)]
[5/4+1/4*7^(1/2)]
y=
[ 5/4+1/4*7^(1/2)]
[5/4-1/4*7^(1/2)]
```

In same way if you have more than two equations you can use the same command to solve them for example:

```
[x,y,z]=solve('x+y+z=1','x+2*y-z=3','2*x-2*z=2')
x =
1/2
y =
1
z =
-1/2
```

2.2 DIFFERENTIAL EQUATIONS

2.2.1 First Order Differential Equations

Matlab can solve linear ordinary differential equations with or withoutinitial/boundary conditions. Do not expect Matlab can solve nonlinear ordinary differentialequations which typically have no analytical solutions. Higher derivatives can be handleads well. The command for finding the symbolic solution of differential equations is **dsolve**.

For that command, the derivative of the function *y* is represented by Dy. For example, suppose that we want to find the solution of the equation $x\,y' - y = 1$. We will have:

```
>>dsolve('x*Dy-y=1', 'x')
ans =
-1+x*C1
```

This means that the solution is any function of the form $y = -1+ cx$, where *c* is any constant. The letter “D” has a special meaning and cannot be used otherwise inside **dsolve**.

It means “first derivative of”. The C1 is a constant of integration.

If we have the initial condition $y(1) = 5$, we can get the particular solution on the following way:

```
>>dsolve('Dy+y=cos(t)')
ans =
1/2*cos(t)+1/2*sin(t)+exp(-t)*C1
>>dsolve('x*Dy-y=1', 'y(1)=5', 'x')
ans =
-1+6*x
```

2.2.2 Second Order Equations

The second order linear equations can be solved similarly as the first order differential equations by using **dsolve**. For the command **dsolve**, the second derivative of *y*

is represented with **D2y**. The letters “**D2**” mean second derivative.

For example, the command for solving ***y''*-3*y*'+2*y* = sin *x*.**

```
>>dsolve('D2y-3*Dy+2*y=sin(x)', 'x')
ans =
3/10*cos(x)+1/10*sin(x)+C1*exp(x)+C2*exp(2*x)
```

If we have the initial conditions y(0) = 1, y'(0)=-1, we would have:

```
>>dsolve('D2y-3*Dy+2*y=sin(x)', 'y(0)=1', 'Dy(0)=-1', 'x')
ans =
3/10*cos(x)+1/10*sin(x)+5/2*exp(x)-9/5*exp(2*x)
```

Example:d^2y/dx^2 **-2dy/dx -3y=**x^2

```
>>dsolve('D2y - 2*Dy - 3*y=x^2', 'x')
ans =
-14/27+4/9*x-1/3*x^2+C1*exp(3*x)+C2*exp(-x)
```

Example:d^2y/dx^2 **-2dy/dx -3y=**x^2**, with y(0)=0,** and **dy/dx =1 at x=1**

```
>>dsolve('D2y - 2*Dy - 3*y=x^2','y(0)=0, Dy(1)=1','x')
ans =
-1/3*x^2+4/9*x-14/27+1/9*(-11+14*exp(3))/(3*exp(3)+exp(-1))*exp(-x)
+1/27*(33+1 4*exp(-1))/(3*exp(3)+exp(-1))*exp(3*x)
```

2.2.3 Higher Order Differential Equations

Similarity you can use the same way to solve the higher order differential equations.

3. Representing Functions

There is a way to define functions in MATLAB that behave in the usual manner. To represent a function in Matlab, we use "**inline**" command. For example to declare $f(x)=x^2+3x+1$ you write:

```
>> f=inline('x^2+3*x+1')
f=
Inline function:
f(x) = x^2+3*x+1
```

Therefore to find f(2), to get the answer you write:

```
>>f(2)
```

```
ans =
11
```

The function *g(x,y)=x2-3xy+2* is defined as follows.

```
>> g=inline('x^2-3*x*y+2')
g =
Inline function:
g(x,y) = x^2-3*x*y+2
```

Now we can evaluate *g*(2,3) in the usual way.

```
>>g(2,3)
ans =
-12
```

In some cases, if we need to define function f as a vector. Then we use:

```
>> f = inline(vectorize('x^2+3*x-2'))
f =
Inline function:
f(x) = x.^2+3.*x-2
```

In this case, we can evaluate a function at more than one point at the same time. For example, to evaluate the above function at 1, 3 and 5 we have:

```
>>f([1 3 5])
ans =
2 16 38
```

4. Differentiation

The Matlab function that performs differentiation is **diff**. These operations show how it works:

```
>>syms x
>>diff(x^2)
ans =
2*x
```

```
>>diff(sin(x)^2)
ans =
2*sin(x)*cos(x)
```

For example, let's find the derivative of *f*(*x*)=sin(*e*x).

```
>>syms x
>>diff(sin(exp(x)))
```

and get the answer as:

```
ans =
cos(exp(x))*exp(x)
```

Note: Instead of using **syms** to declare of variables you can use two Quotes ' ' to declare that the variable x is the interested variable in equation; you can use the same example in otherwise

```
>>diff('sin(exp(x))')
ans =
cos(exp(x))*exp(x)
```

The ***nth***derivative of ***f***is in the written in the form ***diff(f,n)*.** then to find the second derivative we write;

```
>>diff(sin(exp(x)),2)
ans =
-sin(exp(x))*exp(x)^2+cos(exp(x))*exp(x)
```

For example to find the first derivative of *x3+3x2+8x* you simply write:

```
>>syms x
>>diff(x^3+3*x^2+8*x)
ans =
3*x^2+6*x+8
```

Moreover to get the 3rd derivative, write:

```
>>diff(x^3+3*x^2+8*x ,3)
ans =
6
```

Note: To get higher derivatives, you can write the degree in place of 3.To compute the partial derivative of an expression with respect to some variable wespecify that variable as an additional argument in **diff**. For example to find the derivative for x in equation $f(x,y)=x^3y^4+y\sin x$.

```
>>syms x y
>>diff(x^3*y^4+y*sin(x),x)
ans =
3*x^2*y^4+y*cos(x)
```

Next we compute diff for y

```
>>diff(x^3*y^4+y*sin(x),y)
ans =
4*x^3*y^3+sin(x)
```

Finally we compute $d^3 f x^3$.

```
>>diff(x^3*y^4+y*sin(x),x,3)
ans =
6*y^4-y*cos(x)
```

5. Integration

By using the Symbolic Toolbox, you can find both definitive and in-definitive integrals of functions. We can use MATLAB for computing both definite and indefinite integrals using the command int. If **f** is a symbolic expression in **x**, then:

int(f) $\rightarrow \int f(x)dx$

For the indefinite integrals, consider the following example:

```
>>int('x^2')
ans =
1/3*x^3
```

Similarly as for diff command, we do not need the quotes if we declare *x* to be a symbolic variable. Therefore the above command can be re-written in otherwise such as:

```
>>syms x
>>int(x^2)
ans =
1/3*x^3
```

For example to find the in-definitive integral of $x^3+\sin(x)$, you write:

```
>>syms x
>>int(x^3+sin(x))
ans =
1/4*x^4-cos(x)
```

A definite integral can be taken by giving three arguments. The second and third arguments in that case are the first and second limits of integration.

int(f, a, b) $\rightarrow \int_{x=a}^{x=b} f(x)dx$

For the definitive integrals:

```
>>int(x^2, 0, 1)
ans =
1/3
```

Try these examples,

```
int(x,1,2)
int(x*sin(x),-2,7)
```

Moreover to get definitive integral to ln(x)+1/(x+1) from x=1 to x=2 write, you simply write:

```
>>int('ln(x) + 1/(x+1)', 1, 2)
ans =
log(6)-1
```

6. Limits

You can use limit to compute limits. For example, to evaluate the limit when x goes to 2 of the function *$(x^2-4)/(x-2)$*,we have:

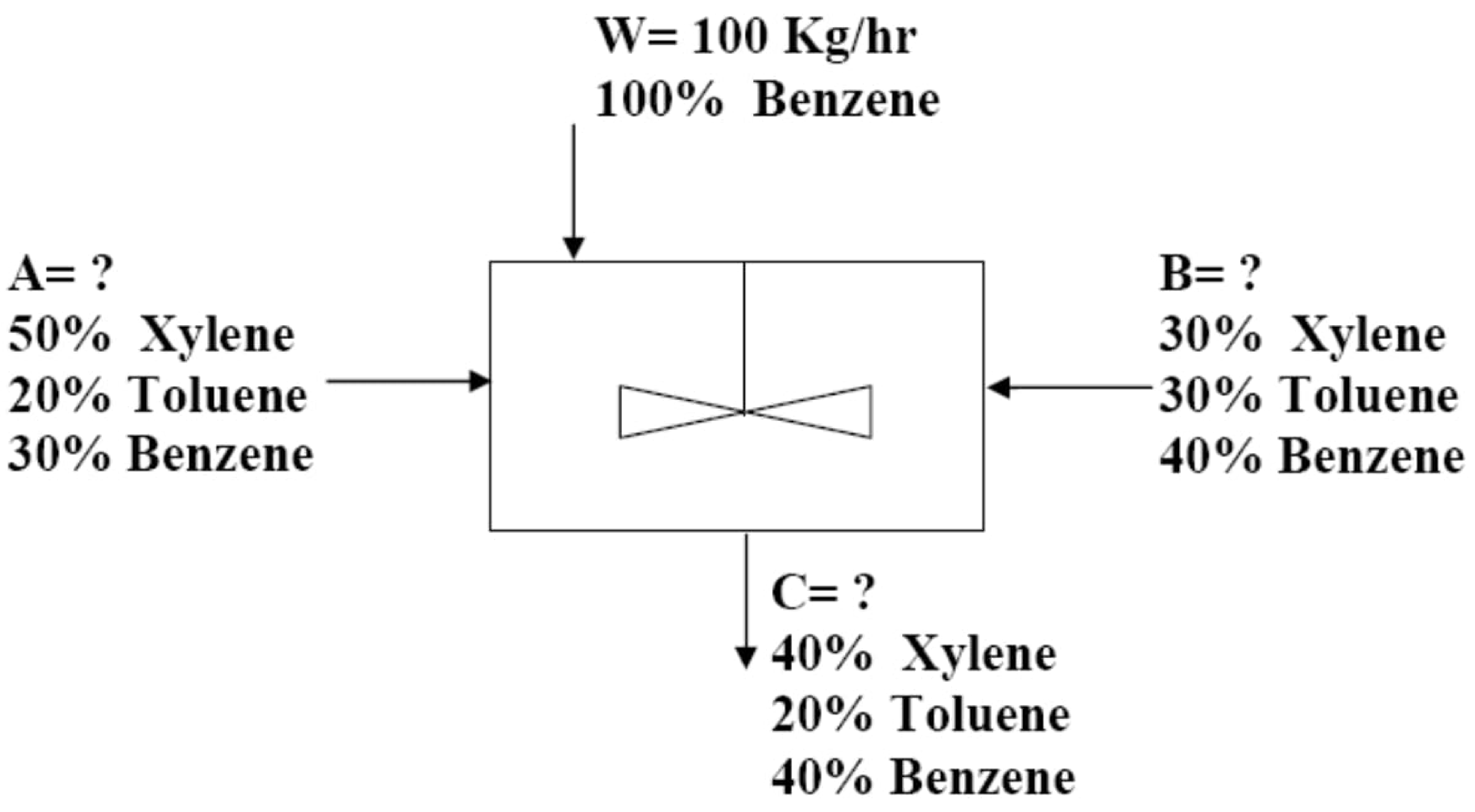
```
>>syms x
>>limit((x^2-4)/(x-2), x, 2)
ans =
4
```

Limits at infinity:

```
>>limit(exp(-x^2-5)+3, x, Inf)
ans =
3
```

Exercise 1:

For the mixer shown below write a code to find the values of streams A, B and C?



Solution: By making component material balance on each component within the mixer you can reach to a system of three equations which can be solve by using the command solve to find the unknowns A, B, C.

Type the following command:

```
[A,B,C]=solve('.5*A+.3*B=.4*C','.2*A+.3*B=.2*C','.3*A+.4*B+100=.4*C')
```

The results will be:

```
A =
600
B =
200
C =
900
```

Exercise 3:

Calculate the heat required to increase the temperature of 1 mol of methane from 533.15 K to 873.15 C at a pressure approximately 1 bar. Where

$$\frac{Cp}{R} = A + BT + CT^{2} + DT^{-2}$$

A=1.702 , B=$9.081*10^{-3}$, C=$-2.164*10^{-6}$, D=0 , R=8.314 and

$$Q = n\int_{Tin}^{Tout} Cpdt$$

Solution:

```
syms T;
T1=533.15; T2=873.15;
A=1.702; B=9.081e-3; C=-2.164e-6; R=8.314;
Cp=(A+B*T+C*T^2);Q=R*int(Cp,T1,T2)
```

The results will be:

```
  Q=
1.9778e+004
```