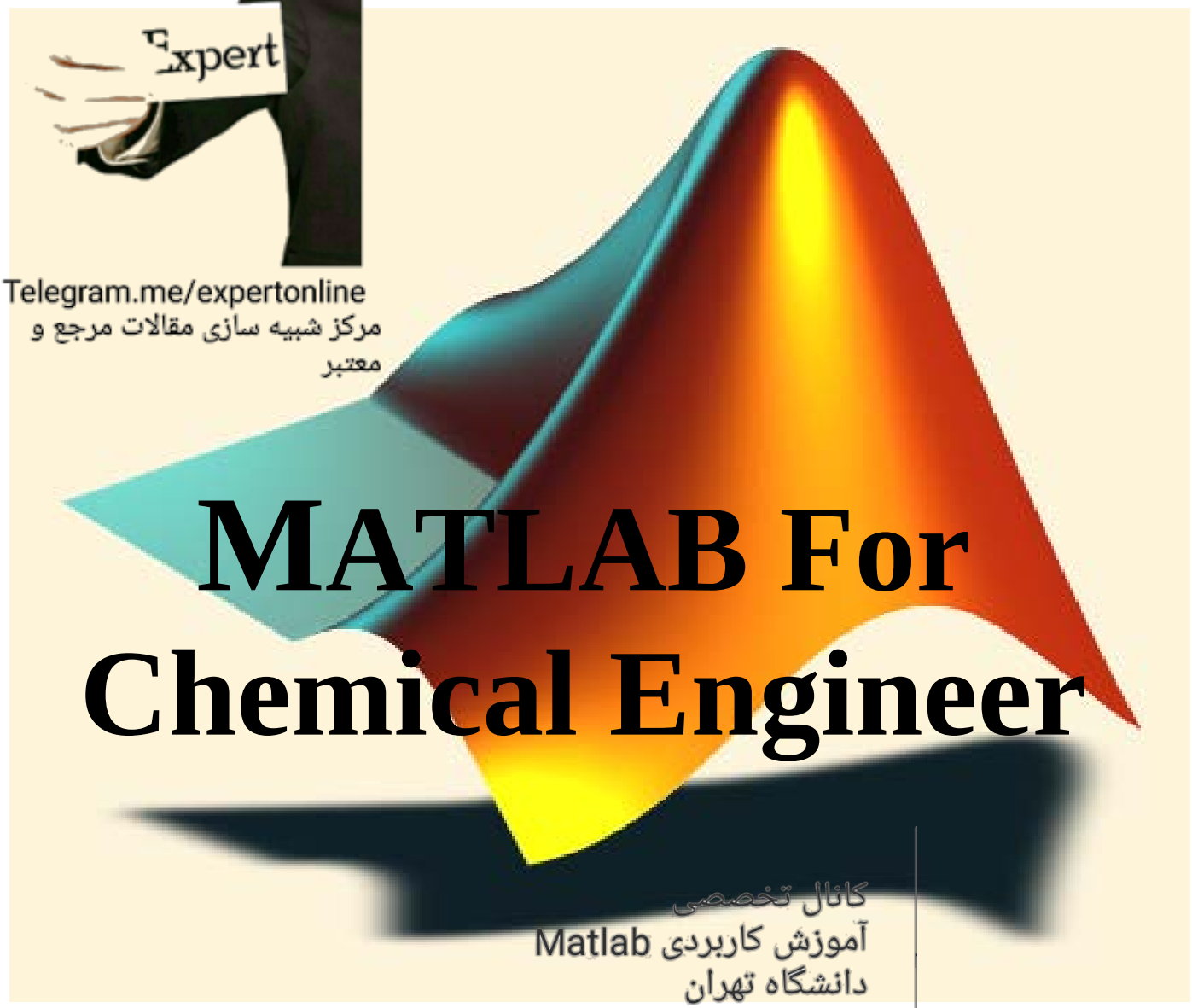


Introduction to MATLAB

MATLAB (Matrix Laboratory) 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 the solution of nonlinear systems of equations, the 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 use Matlab 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.

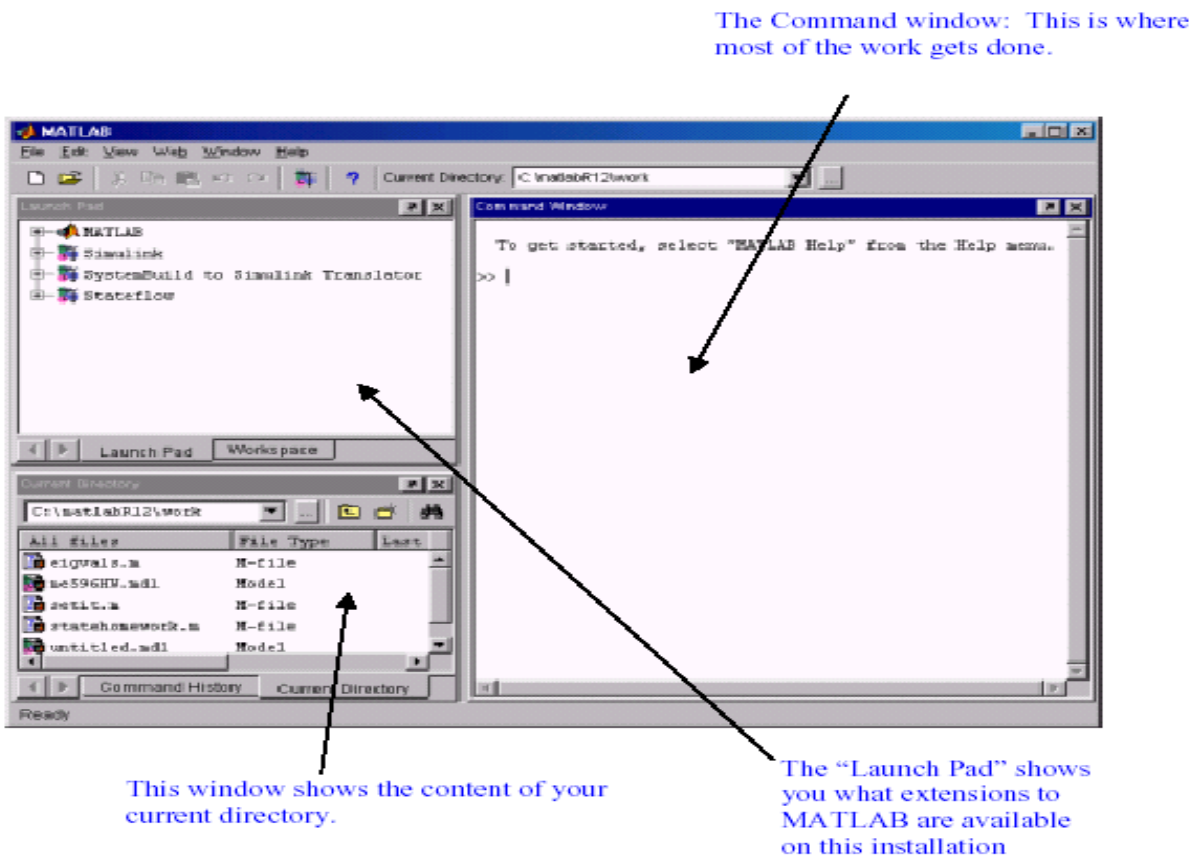


Figure 1: The MATLAB Window

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.

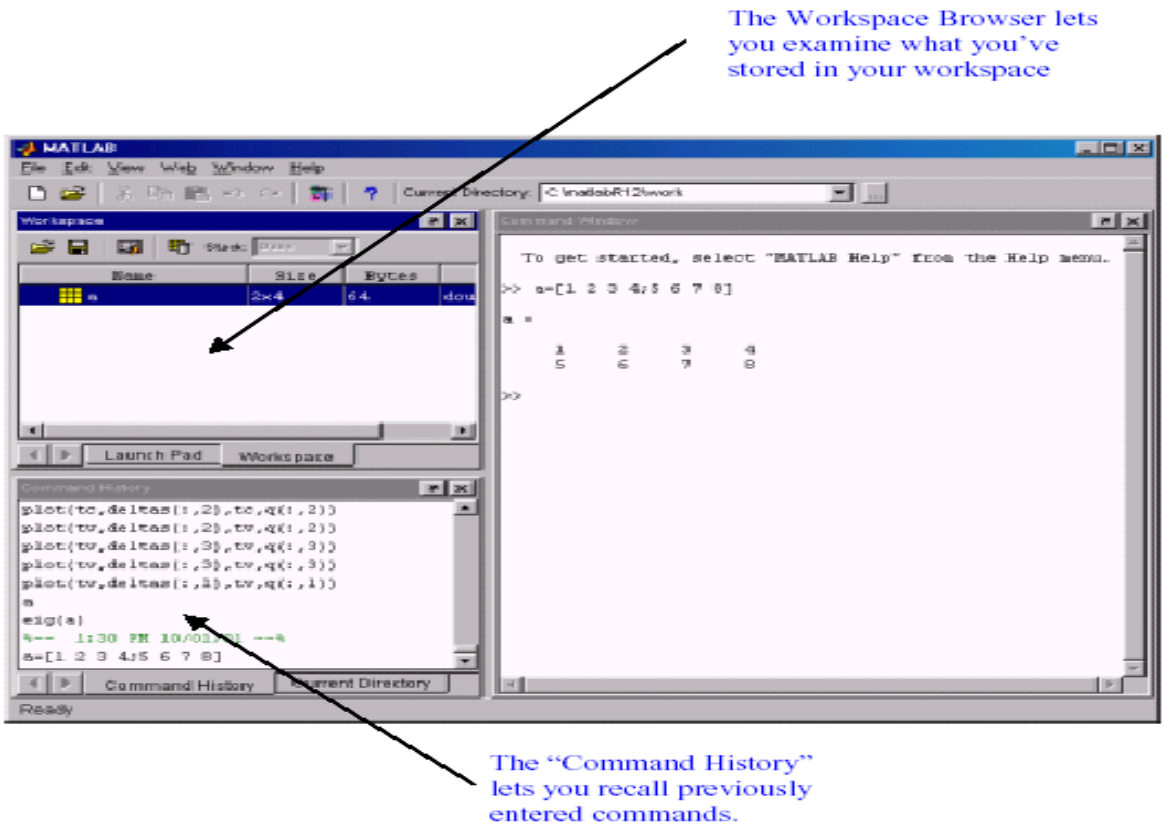


Figure 2: Alternate windows in the default frame

1.2 The command window:

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.

One may enter a sequence of commands into a M-file. Entering the M-file name in the command window executes the commands. Once a M-file is created, one can easily alter the commands for specific problems without having to reenter the entire sequence from 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 pressing Enter key. If you type again:

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

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 subroutines that 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 the Matlab package. Function files are created and edited in identically the same manner as the 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 is generated by overflow or by the operation of dividing by zero. The NaN stands for Not-A-Number and it is obtained as a result of the mathematically undefined operations such as 0.0/0.0.

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

+ : addition

- : substraction

* : 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
```

Similarly:

```
>> 5^3
ans=
   125
>>2.5^2
ans =
   6.2500
>>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
```

Another special value is NaN, which stands for not a number. NaN is used to express an undefined value. For example,

```
>> 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. The main ones are:

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: that 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*, *eps*, *j*):

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 arcsin or sin 1 etc.).

The result is in radians.

```
>> acos(x/5), asin(y/5)
ans =
   0.5236
ans =
   0.5236
>> pi/6
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.

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( ).

```
>> x =9;
>> sqrt(x),exp(x),log(sqrt(x)),log10(x^2+6)
ans =
   3
ans =
   8.1031e+03
ans =
   1.0986
ans =
   1.9395
>> 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

conj( ) - Complex conjugate.

Imag( ) - Complex imaginary part.

Real( ) - Complex real part.

7. 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

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 Antoine coefficients:

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
```

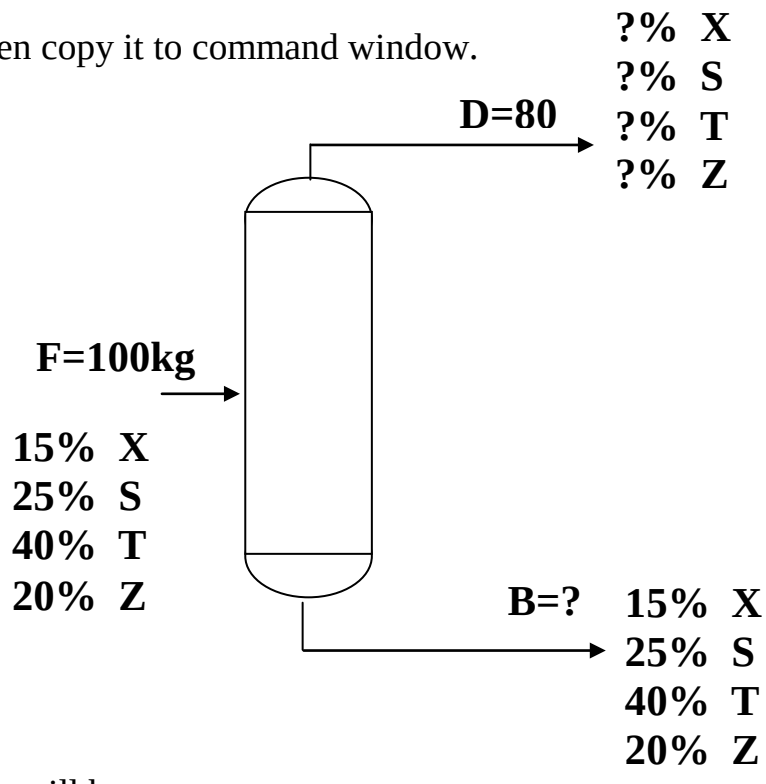
Exercise 3:

For the following distillation column write a code to find the value of stream B and the compositions of stream D?

Solution:

Type the commands as m-file. Then copy it to command window.

```
F=100; D=80; B=F-D
XF=0.15;
SF=0.25;
TF=0.4;
ZF=0.2;
XB=0.15;
SB=0.25;
TB=0.4;
ZB=0.2;
XD=(F*XF-B*XB)/D*100
SD=(F*SF-B*SB)/D*100
TD=(F*TF-B*TB)/D*100
ZD=(F*ZF-B*ZB)/D*100
```



Then after pressing enter the result will be:

```
B =
    20
XD =
    15
SD =
    25
TD =
    40
ZD =
    20
```

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 x^2-y^2;

```
>>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 ((x^2-y^2)^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 solving equations, you can use the command solve. 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.

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

For example, let us solve the equation $3x^2-8x+2=0$.

```
>> solve('3*x^2-8*x+2=0','x')
ans =
[ 4/3+1/3*10^(1/2)]
[ 4/3-1/3*10^(1/2)]
```

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 ax^2+bx+c in x in terms of a, b and c. By default solve sets the given expression equal to zero if an equation is not given.

```
>> solve('a*x^2+b*x+c','x')
ans =
[ 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 more than one equation simultaneously. For example suppose that we need 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).

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 screen:

```
x =
 -48/5
 y =
 47/5
```

```
[x,y]=solve('log(x)+x*y=0', 'x*y+5*y=1')
x =
.8631121967939437
y =
.1705578823046945
```

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)]
```

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

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

In same way if you have more then 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 without initial/boundary conditions. Do not expect Matlab can solve nonlinear ordinary differential equations which typically have no analytical solutions. Higher derivatives can be handled as 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('x*Dy-y=1', 'y(1)=5', 'x')
ans =
-1+6*x
```

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

>> dsolve('Dy+2*y=12*t^2')
ans =
 6*t^2-6*t+3+exp(-2*t)*C1
```

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 $y''-3y'+2y = \sin x$.

```
>> 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: $\mathbf{d^2y/dx^2 - 2dy/dx - 3y = x^2}$

```
>> 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: $\mathbf{d^2y/dx^2 - 2dy/dx - 3y = x^2}$**, 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)=x2+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 $g(x,y)=x^2-3xy+2$ 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(x^n)
ans =
x^n*n/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 n^{th} 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 we specify 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 $d^3 fx^3$.

```
>> 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) → $\int f(x)dx$

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)
```

Try these examples:

int(x)
int(a^x)
int(a^x^2)
int(x*sin(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) → $\int_{x=a}^{x=b} f(x)dx$

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)
int(a^x,a,3,4)

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 =
-1+log(2)+log(3)
```

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:

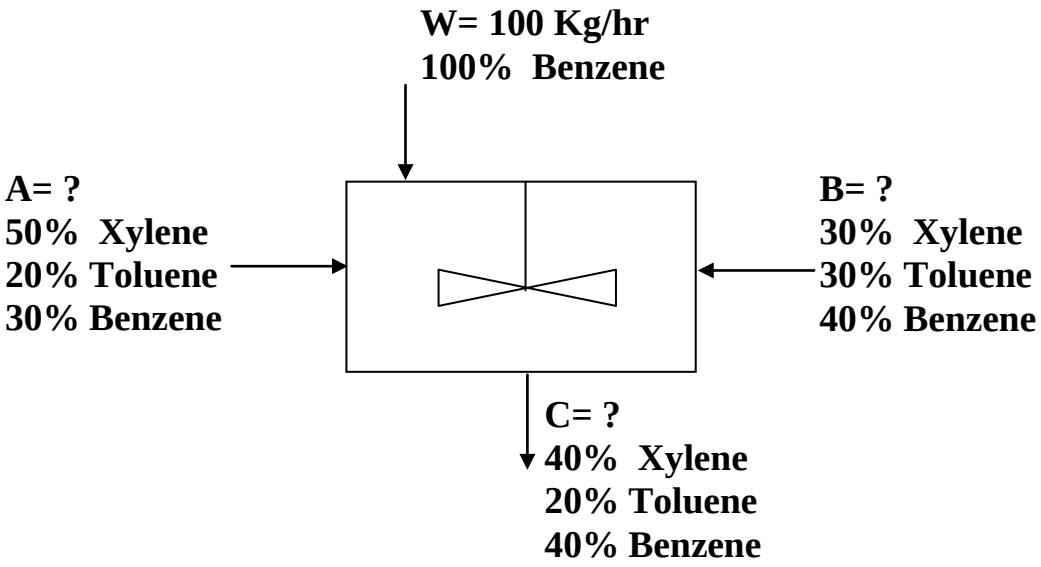
```
>> syms x
>> limit((x^2-4)/(x-2), x, 2)
ans =
4
```

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:

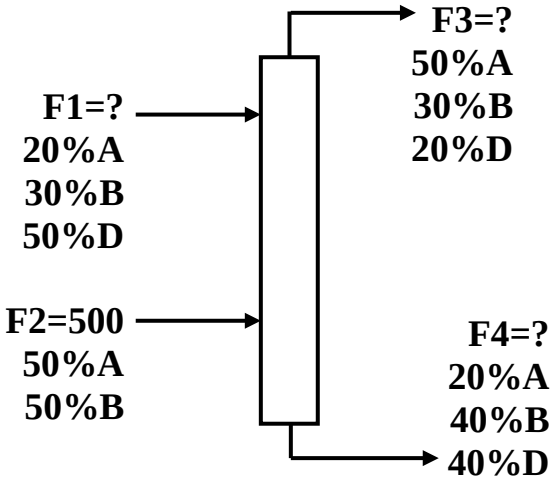
```
[A,B,C]=solve('.5*A+.3*B=.4*C','.2*A+.3*B=.2*C','.3*A+.4*B+100=.4*C')
```

The results will be:

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

Exercise 2:

For the following distillation column calculate the values of F1, F3 and F4?



solution:

```
[F1,F3,F4]=solve('.2*F1+250=.5*F3+.2*F4','.3*F1+250=.3*F3+.4*F4','.5*F1=.2*F3+.4*F4')
```

The results will be:

```
F1=
1000
 F3=
 500
 F4 =
1000
```

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
```

Practice problems:

1. Factor $x^3+3x^2y+3xy^2+y^3$.
2. Simplify $(x^3-8)/(x-2)$.
3. Expand $(x^2+1)(x-5)(2x+3)$.
4. Solve $\sin x = 2-x$ for x.
5. Solve $5x+2y+4z = 8$, $-3x+y+2z = -7$, $2x+y+z = 3$ for x, y and z.
6. Solve $y^2-5xy-y+6x^2+x = 2$ for x.
7. Find the first derivative of the function $(\sin x/(\ln(x^2+1))-e^x$ and evaluate it at x=3.
8. Find the 12th derivative of the function $(x/2+1)^{65}$
9. Find the first and second partial derivatives of the function $e^{x2}\sin xy$

Vectors

An **array** is a collection of numbers, called **elements**, referenced by one or more indices running over different index sets. In MATLAB, the index sets are always sequential integers starting with 1. The **dimension** of the array is the number of indices needed to specify an element. The **size** of an array is a list of the sizes of the index sets.

A **matrix** is a two-dimensional array with special rules for addition, multiplication, and other operations. The two dimensions are called the **rows** and the **columns**.

A **vector** is a matrix in which one dimension has only the index 1. A **row vector** has only one row and a **column vector** has only one column.

1. Entering Vectors and Matrices

There are several ways to enter vectors and matrices in Matlab. These include:

1- Entering an explicit list of elements.

2- Loading matrices from external data files.

3- Generating matrices using functions.

To enter a vector or matrix explicitly, there are a few basic rules to follow:

• Separate the elements of a row with spaces or commas.

• Use a semicolon ; or returns, to indicate the end of each row.

• Surround the entire list of elements with square brackets, [ ].

For example, to input a 3×1 vector:

```
>> u=[1 2 3]
u =
    1     2     3
```

The entries must be enclosed in square brackets.

```
>> v =[ 1 3  sqrt(5)]
v =
   1.0000 3.0000 2.2361
```

Spaces is very important:

```
>> v2 =[3+ 4 5]
v2 =
   7     5
>> v3 =[3 +4 5]
v3 =
   3     4     5
```

We can do certain arithmetic operations with vectors of the same length, such as v and v3 in the previous section.

```
>> v + v3
ans =
   4.0000 7.0000 7.2361
>> v4 =3*v
v4 =
   3.0000 9.0000 6.7082
>> v5 =2*v -3*v3
v5 =
   -7.0000 -6.0000 -10.5279
```

We can build row vectors from existing ones:

```
>> w =[1 2 3], z = [8 9]
w =
    1    2    3
z =
    8    9
>> v6=[w z]
v6 =
    1    2    3    8    9
```

A vector can be defined by previously defined vectors.

```
>> x = [2 4 -1]
x =
    2    4   -1
>> x1 = [x  5 8 ]
x1 =
    2    4   -1    5    8
```

An special array is the **empty matrix**, which is entered as [] .and can be used to delete a part of any matrix or vector.

```
>> w=[4 5 6 7]
w =
    4    5    6    7
>>w(4)=[];w
w =
    4    5    6
```

The elements of a matrix can be defined with algebraic expressions placed at the appropriate location of the element. Thus

```
>> a = [ sin(pi/2)  sqrt(2)  3+4  6/3  exp(2) ]
a =
   1.0000    1.4142    7.0000    2.0000    7.3891
```

2. Column Vectors

The column vectors have similar constructs to row vectors. When defining them, entries are separated by ; or "newlines".

Note how the column vector is defined, using brackets and semicolons to separate the different rows. To define a column vector x:

```
x=[1; -2; 4]
x =
1
-2
4
```

or write

```
>>x=[1
2
3]
x =
1
-2
4
>> c =[ 1; 3; sqrt(5)]
c =
1.0000
3.0000
2.2361
```

3. Transposing

We can convert a row vector into a column vector (and vice versa) by a process called *transposing*, denoted by **'**

```
> A=[ 1 2 3]
A =
   1    2    3
```

```
>>B= A'
B =
    1
    2
    3
```

4. Vectors Addition and subtraction

Addition and subtraction of a number to or from a vector can be made. In this case, the number is added to or subtracted from all the elements of the vector. For example

```
>> x=[-1; 0; 2];
>> y=x-1
y =
   -2
   -1
    1
```

If we look to make a simple addition and subtraction of vectors. The notation is the same as found in most linear algebra. We will define two vectors then add or subtract them:

```
>> v = [1; 2; 3]
v =
    1
    2
    3
>> b = [2; 4; 6]
b =
    2
    4
    6
>> v+b
ans =
    3
    6
    9
>> v-b
ans =
   -1
```

```
    -2
    -3
>> sin(v)
ans =
    0.8415
    0.9093
    0.1411
>> log(v)
ans =
         0
    0.6931
    1.0986
>> pi*v
ans =
    3.1416
    6.2832
    9.4248
```

5. Vectors multiplication

Multiplication of vectors and matrices must follow special rules. In the example above, the vectors are both column vectors with three entries. You cannot add a row vector to a column vector. In the case of multiplication vectors, the number of columns of the vector on the left must be equal to the number of rows of the vector on the right.

```
>> b = [2; 4; 6];
>> v = [1; 2; 3];
>> v*b
??? Error using ==> *
Inner matrix dimensions must agree.
>> v*b'
ans =
     2     4     6
     4     8    12
     6    12    18
>> v'*b
ans =
    28
```

6. element-wise operation

There are many times where we want to do an operation to every entry in a vector or matrix. Matlab will allow you to do this with "element-wise" operations. For example, suppose you want to multiply each entry in vector v with its corresponding entry in vector b. In other words, suppose you want to find v(1)*b(1), v(2)*b(2), and v(3)*b(3). It would be nice to use the "*" symbol since you are doing some sort of multiplication, but since it already has a definition, we have to come up with something else. The programmers who came up with Matlab decided to use the symbols ".*" to do this.

```
>> v.*b
ans =
     2
     8
    18
```

Also for division we must use "./"

```
>> v./b
ans =
    0.5000
    0.5000
    0.5000
```

Note that

v .* b multiplies each element of v by the respective element of b.

v ./ b divides each element of v by the respective element of b.

v .\ b divides each element of b by the respective element of v.

v .^ b raise each element of v by the respective b element.

7. The Colon Operator

The colon operator ' : ' is understood by 'matlab' to perform special and useful operations. If two integer numbers are separated by a colon, 'matlab' will generate all of the integers between these two integers.

In particular, you will be able to use it to extract or manipulate elements of matrices. The following command creates a row vector whose components increase arithmetically:

```
>> 1:5
ans =
     1 2 3 4 5
```

And, if three numbers, integer or non-integer, are separated by two colons, the middle number is interpreted to be a "step" and the first and third are interpreted to be "limits". Thus

```
>> b = 0.0 : 0.2 : 1.0
b =
    0    0.2000    0.4000    0.6000    0.8000    1.0000
```

Suppose you want to create a vector with elements between 0 and 20 evenly spaced in increments of 2. Then you have to type:

```
>> t = 0:2:20
t =
    0    2    4    6    8   10   12   14   16   18   20
>> m=0.32:0.1:0.6
m =
   0.3200   0.4200   0.5200
>>w= -1.4:-0.3:-2
w =
   -1.4000  -1.7000  -2.0000
```

The format is **first:step:last**. The result is always a row vector.

A negative step is also allowed. The command has similar results; it creates a vector with linearly spaced entries.

There is another way to create row arrays is to use linspace functions:

```
>> A=linspace(1,2,5)
A =
   1.0000   1.2500   1.5000   1.7500   2.0000
>> A=linspace(0,20,11)
A =
    0    2    4    6    8   10   12   14   16   18   20
```

8. Referencing elements

It is frequently necessary to call one or more of the elements of a vector. Each dimension is given a single index. Some examples using the definitions above:

Evaluate a vector A:

```
>> A=0:10:100
A =
    0   10   20   30   40   50   60   70   80   90  100
```

Now after definition of a vector A, try to type:

```
>> A(10)
ans =
   90
>> B=A(1:5)
B =
    0    10    20    30    40
>> C=A(1:2:10)
C =
    0    20    40    60    80
>> A(6:2:10)
ans =
   50    70    90
```

Exercise 1:

Calculate Reynold Number at a diameter D=0.2 m, using different velocity's as u=0.1,0.2,0.3 …1 m/s knowing that:

Re= (ρud)/μ , μ=0.001 , ρ =1000

Solution:

Type the code in the command window

```
d=0.2;
u=0.1:.1:1;
m=0.001;
p=1000;
Re=u*p*d/m
```

The results will be

```
Re =
1.0e+005 *
0.2000    0.4000    0.6000    0.8000    1.0000    1.2000    1.4000    1.6000
1.8000    2.0000
```

Exercise 2:

Estimate the average density of a Water/Ethanol mixture at different water compositions knowing that.

Water density =1000 kg/m^3

Ethanol density =780 kg/m^3

Mixture density= Xwater × Water density + Xethanol × Ethanol density

Solution:

```
Pwater=1000; Pethanol=780;
Xwater=0:.1:1
Xethanol=1-Xwater;
Pav=Pwater*Xwater+Pethanol*Xethanol
Xwater =
         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000
0.8000    0.9000    1.0000
Pav =
         780         802         824         846         868         890         912         934
956         978        1000
```

Practice Problems

1. Create a vector of the even whole numbers between 31 and 75.

2. Let x = [2 5 1 6].
   a) Add 16 to each element.
   b) Add 3 to just the odd-index elements.
   c) Compute the square root of each element.
   d) Compute the square of each element.

3. Let x = [3 2 6 8]' and y = [4 1 3 5]' .
   a) Add the elements in x to y
   b) Raise each element of x to the power specified by the corresponding element in y.
   c) Divide each element of y by the corresponding element in x
   d) Multiply each element in x by the corresponding element in y, calling the result "z".

4. When A and B contain the values shown below, give the values of C vector after executing the following statements.

A = [2 -1 5 0]; B = [3 2 -1 4]

   a) C = A-B
   b) C = B + A -3
   c) C = B./A
   d) C = A.^B
   e) C = 2.^B+A
   f) C = 2*A+A.^B

9. Other Operations on Vectors

MATLAB has a large number of built-in functions. You will only become familiar with them by using them.

Try to make the vector **v** in command window and use the functions below.

v=[23 0 3 16 -8 13]

length(v) number of elements in v.

6

size(v) size of matrix v (raw, column).

1 6

find(v) finds indices of non-zero elements.

1 3 4 5 6

find(v==0) finds indices of elements equal to zero.

2

find(v==16) finds indices of elements equal to 16.

4

find(v>7) finds indices of elements greater than 7.

1 4 6

v(find(v>7)) finds the values of elements greater than 7.

23 16 13

sum(v) sum of elements

47

max(v) maximum element.

23

min(v) minimum element.

-8

mean(v) mean of elements.

7.8333

sort(v) sorts elements from minimum to maximum value.

-8 0 3 13 16 23

all(v) equal to 1 if all elements nonzero, 0 if any element nonzero.

0

abs(v) vector with absolute value of all element

23 0 3 16 8 13

Exercise 1:

Write a program to calculate average density, conductivity and specific heat for water in the range of temperatures from 0 to 50 C . Knowing that this parameters for water are a function of temperature such as the following equations.

The Density

$\rho = 1200.92 - 1.0056\ T_K^o + 0.001084 * (T_K^o)^2$

The condactivity

$K = 0.34 + 9.278 * 10^{-4}\ .T_K^o$

The Specific heat

$C_P = 0.015539\ (T_K^o - 308.2)^2 + 4180.9$

Note: take 11 point of temperatures

Solution:

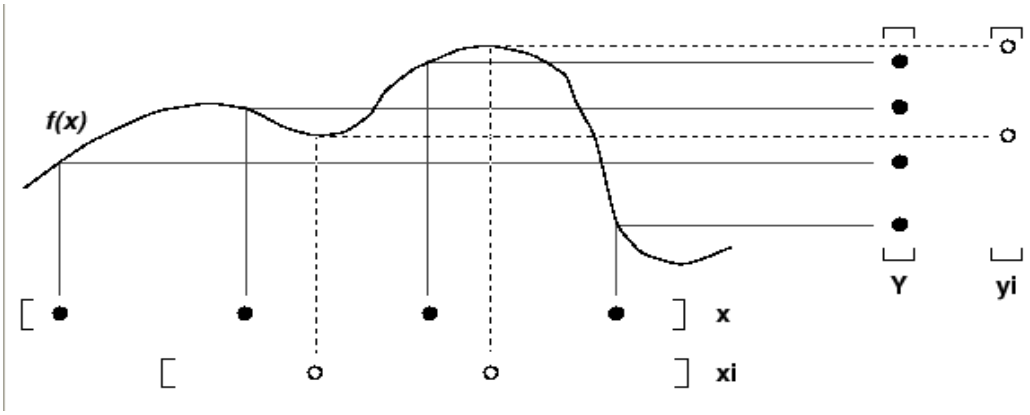
```
T=[0:5:50]+273;
p= 1200.92 - 1.0056*T+ 0.001084 * T.^2;
Kc = 0.34 + 9.278 * 10^-4 *T;
Cp = 0.015539*(T - 308.2).^2 + 4180.9;
Average_density=mean(p)
Average_conductivity=mean(Kc)
Average_specificheat=mean(Cp)
```

Gives the results

```
Averagedensity =
  997.7857
Averageconductivity =
0.6165
Averagespecificheat =
4.1864e+003
```

10. Polynomial Interpolation

The command **interp1** interpolates between data points. It finds values at intermediate points, of a one-dimensional function that underlies the data. This function is shown below, along with the relationship between vectors x, y, xi, and yi.



Interpolation is the same operation as table lookup. Described in table lookup terms, the table is [x,y] and **interp1** looks up the elements of xi in x, and, based upon their locations, returns values yi interpolated within the elements of y.

Syntax

yi = interp1(x,y,xi)

where xi may be single element or a vector of elements.

Exercise 2:

The vapor pressures of 1-chlorotetradecane at several temperatures are tabulated here.

T (^{o}C)	98.5	131.8	148.2	166.2	199.8	215.5
P*(mmHg)	1	5	10	20	60	100

Calculate the value of vapor pressure corresponding to 150 ^{o}C?

Solution:

```
T= [98.5 131.8 148.2 166.2 199.8 215.5];
P= [1 5 10 20 60 100];
Pi=interp1 (T, P, 150)
```

The result will be:

```
Pi=
   11.0000
```

Exercise 3:

The heat capacity of a gas is tabulated at a series of temperatures:

T (^{o}C)	20	50	80	110	140	170	200	230
Cp j/mol.^{o}C	28.95	29.13	29.30	29.48	29.65	29.82	29.99	30.16

Calculate the values of heat capacity corresponding to 30, 70, 100 and 210 ^{o}C.

Solution:

```
T= [20 50 80 110 140 170 200 230];
P= [28.95 29.13 29.30 29.48 29.65 29.82 29.99 30.16];
Pv=interp1 (T, P, [30 70 100 210])
```

The results will be

```
Pv =
 29.0100   29.2433   29.4200   30.0467
```

11. Polynomials in Matlab

The equation $p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ is called a polynomial in x. The terms of this polynomial are arranged in descending powers of x while the ai's are called the coefficients of the polynomial and are usually real or complex numbers. The degree of the polynomial is n (the highest available power of x).

In Matlab, a polynomial is represented by a vector. To create a polynomial in Matlab, simply enter each coefficient of the polynomial into the vector in descending order (from highest-order to lowest order). Let's say you have the following polynomial:

$y=x^4+3x^3-15x^2-2x+9$

To enter this polynomial into Matlab, just enter it as a vector in the following manner:

y = [1 3 -15 -2 9]

Also to represent the polynomial $y=2x^3 + 2x^2 + 4x + 1$.

y = [2 2 4 1];

Matlab can interpret a vector of length n+1 as an n^{th} order polynomial. Thus, if your polynomial is missing any coefficients, you must enter zeros in the appropriate place in the vector. For example, $p(x) = x^6-2x^4-x^3+2x^2+5x-8$ is a polynomial that is arranged in descending powers of x. The coefficients of the polynomial are 1, 0,−2, −1, 2, 5, and −8. In this case, there is an understood term of x^5. However, Matlab doesn't "understand" that a term is missing unless you explicitly tell it so.

For other example, the polynomial $y=x^4+1$ would be represented in Matlab as:

y = [1 0 0 0 1]

12. Roots

To calculate the roots of a polynomial, enter the coefficients in an array in descending order. Be sure to include zeroes where appropriate.

For example to find the roots of the polynomial $y= x^4 + 6x^3 + 7x^2 - 6x - 8 = 0$ type the command:

```
p = [ 1 6 7 -6 -8 ];
r = roots(p)
```

yields

```
r =
   -4.0000
   -2.0000
   -1.0000
    1.0000
```

Note: The coefficients could be entered directly in the roots command. The same answer as above would be obtained using the following expression.

```
 r = roots([ 1 6 7 -6 -8 ])
r =
   -4.0000
    1.0000
   -2.0000
   -1.0000
```

For example finding the roots of $y=x^4+3x^3-15x^2-2x+9=0$ would be as easy as entering the following command;

```
 r=roots([1 3 -15 -2 9])
r =
   -5.5745
    2.5836
   -0.7951
    0.7860
```

The **roots** command can find imaginary roots.

```
p = [ 1 -6 18 -30 25 ];
r = roots(p)
r =
   1.0000 + 2.0000i
   1.0000 - 2.0000i
   2.0000 + 1.0000i
   2.0000 - 1.0000i
```

It can also find repeated roots. Note the imaginary portion of the repeated roots is displayed as zero.

```
 p = [ 1 7 12 -4 -16 ];
 r = roots(p)
r =
   -4.0000
   -2.0000 + 0.0000i
   -2.0000 - 0.0000i
   1.0000
```

13. PolyVal

You can use polyval and the fitted polynomial p to predict the y value of the data you've fitted for some other x values. The syntax for determining the value of a polynomial $y=x^4 + 6x^3 + 7x^2 - 6x - 8$ at any point is as follows.

```
p = [ 1 6 7 -6 -8 ];
y= polyval(p, 3)
y=
   280
```

Where p is the vector containing the polynomial coefficients, (see above). Similarly, the coefficients can be entered directly in the polyval command.

```
 y = polyval([1 6 7 -6 -8], 3)
y =
   280
```

The polynomial value at multiple points (vector) can be found.

```
 z = [ 3 5 7];
 y = polyval(p,z)
y =
   280   1512   4752
```

14. Polyfit

To determining the coefficients of a polynomial that is the best fit of a given data you can use **polyfit** command. The command is **polyfit(x, y, n),** where x, y are the data vectors and 'n' is the order of the polynomial for which the least-squares fit is desired.

Exercise 4:

Fit x, y vectors to 3 rd order polynomial

```
 x = [ 1.0 1.3 2.4 3.7 3.8 5.1 ];
 y = [ -6.3 -8.7 -5.2 9.5 9.8 43.9 ];
 coeff = polyfit(x,y,3)
coeff =
   0.3124   1.5982   -7.3925   -1.4759
```

After determining the polynomial coefficients, the **polyval** command can be used to predict the values of the dependent variable at each value of the independent variable.

```
ypred = polyval(coeff,x)
ypred =
   -6.9579   -7.6990   -5.6943   8.8733   10.6506   43.8273
```

Its clear that there is a deviation between the actual y points and predicted y points because that the polynomial is best fit to this actual points.

Exercise 5:

Fit the following data describing the accumulation of species A over time to a second order polynomial, then by using this polynomial, predict the accumulation at 15 hours.

Time (hr)	1	3	5	7	8	10
Mass A acc.	9	55	141	267	345	531

Solution: First, input the data into vectors, let:

```
a = [9, 55, 141, 267, 345, 531];
time = [1, 3, 5, 7, 8, 10];
```

Now fit the data using polyfit

```
coeff = polyfit(time,a,2)
coeff =
    5.0000 3.0000 1.0000
```

So, Mass A = $5*(time)^2 + 3 * (time) + 1$

Therefore to calculate the mass A at 15 hours

```
MApred = polyval(coeff,15)
MApred =
  1.1710e+003
```

Exercise 6:

Fit the following vapor pressure vs temperature data in fourth order polynomial. Then calculate the vapor pressure when T=100 ^{O}C.

Temp (C)	-36.7	-19.6	-11.5	-2.6	7.6	15.4	26.1	42.2	60.6	80.1
Pre. (kPa)	1	5	10	20	40	60	100	200	400	760

Solution:

```
vp = [ 1, 5, 10, 20, 40, 60, 100, 200, 400, 760];
T = [-36.7, -19.6, -11.5, -2.6, 7.6, 15.4, 26.1, 42.2, 60.6, 80.1];
p=polyfit(T,vp,4)
pre= polyval(p,100)
```

the results will be:

```
p =
   0.0000    0.0004    0.0360    1.6062   24.6788
pre =
  1.3552e+003
```

Exercise 7:

The calculated experimental values for the heat capacity of ammonia are:

T (C)	Cp (cal /g.mol C)
0	8.371
18	8.472
25	8.514
100	9.035
200	9.824
300	10.606
400	11.347
500	12.045

1. Fit the data for the following function

$$Cp = a + bT + CT^2 + DT^3$$

Where T is in C

2. Then calculate amount of heat Q required to increase the temperature of 150 mol/hr of ammonia vapor from 0 C to 200 C if you know that:

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

Solution:

```
T=[0,18,25,100,200,300,400,500]
Cp=[8.371, 8.472, 8.514, 9.035, 9.824, 10.606, 11.347, 12.045]
P=polyfit(T,Cp,3)
n=150;
syms t
Cpf=P(4)+P(3)*t+P(2)*t^2+P(1)*t^3;
Q= n*int(Cpf, 0,200)
  2.7180e+005
```

Practice Problems

1) Find the 3rd order polynomial that satisfies the data of water for saturation temperature and pressure. Using the predicted polynomial, compute the saturation pressure at 65 C.

Temp (C)	0	10	20	30	40	50	60	70	80	90	100
Pre. (kPa)	.6108	1.227	2.337	4.241	7.375	12.335	19.92	31.16	47.36	70.11	101.33

2) Write a MATLAB program to fit the following vapor pressure vs. temperature data to calculate the values of constants A and B in following equation.

$$\log(P^0) = A - \frac{B}{T + 273.15}$$

Temp (C)	-36.7	-19.6	-11.5	-2.6	7.6	15.4	26.1	42.2	60.6	80.1
Pre. (kPa)	1	5	10	20	40	60	100	200	400	760

3)The experimental velocity of an incompressible fluid in a pipe of radius 1 m is tabulated as below:

r (m)	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
V	1.0	0.99	0.96	0.91	0.84	0.75	0.64	0.51	0.36	0.19	0.0

Where: r is the distance from the centre of the pipe and u is the velocity of the fluid.

Write a MATLAB program to fit the experimental velocity to the following function: $u=a+br+cr^2$

4) Given the following experimental data:

Flow rate (q, m3/s)	1	2	3	4	5	6
Temperature (T, ^oC)	2.2	7.8	13.9	27.8	41.2	62.1

Find the temperature for q = 2.5, 3.5, 4.5;

5) The rate at which a substance passes through a membrane is determined by the diffusivity D,(cm^2/s) of the gas. D varies with the temperature T (K) according to the following law:

$D = D_o \exp(-E/RT)$

Where:

D_o is the pre-exponential factor

E is the activation energy for diffusion

R =1.987 cal/mol K.

Diffusivities of SO_2 in a certain membrane are measured at several temperatures with the data listed below.

$T(K)$	347	374.2	396.2	420.7	447.7	471.2
$D(cm^2/s)\times10^6$	1.34	2.5	4.55	8.52	14.07	19.99

Write a MATLAB program to calculate the values of D_o and E.

Matrices

1. Entering matrices

Entering matrices into Matlab is the same as entering a vector, except each row of elements is separated by a semicolon (;) or a return:

```
>>B = [1 2 3 4; 5 6 7 8; 9 10 11 12]
B =
    1    2    3    4
    5    6    7    8
    9   10   11   12
```

Alternatively, you can enter the same matrix as follows:

```
>>B = [ 1 2 3 4
5 6 7 8
9 10 11 12]
B =
    1    2    3    4
    5    6    7    8
    9   10   11   12
```

Note how the matrix is defined, using brackets and semicolons to separate the different rows.

2. Transpose

The special character prime ' denotes the transpose of a matrix e.g.

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

3. Matrix operations

3.1 Addition and subtraction

Addition and subtraction of matrices are denoted by + and -. This operations are defined whenever the matrices have the same dimensions.

For example: If A and B are matrices, then Matlab can compute A+B and A-B when these operations are defined.

```
>> A = [1 2 3;4 5 6;7 8 9]
A =
    1    2    3
    4    5    6
    7    8    9
>> B = [1 1 1;2 2 2;3 3 3]
B =
  1    1    1
  2    2    2
  3    3    3
>> C = [1 2;3 4;5 6]
C =
    1    2
    3    4
    5    6
>> A+B
ans =
    2    3    4
    6    7    8
   10   11   12
>> A+C
??? Error using ==>+
Matrix dimensions must agree.
```

Matrices can be joined together by treating them as elements of vectors:

```
>>D=[A B]
D =
    1    2    3    1    1    1
    4    5    6    2    2    2
    7    8    9    3    3    3
>>A - 2.3
ans =
  -1.3000   -0.3000    0.7000
   1.7000    2.7000    3.7000
   4.7000    5.7000    6.7000
```

3.2 Matrix multiplication

Matrix operations simply act identically on each element of an array. We have already seen some vector operations, namely + and - , which are defined for vectors the same as for matrices. But the operators * , / and ^ have different matrix interpretations.

```
>> A=[1,2,3;4,5,6;7,8 0]
A =
    1    2    3
    4    5    6
    7    8    0
>> B=[1,4,7;2,5,8;3,6,0]
B =
    1    4    7
    2    5    8
    3    6    0
>> A*B
ans =
   14   32   23
   32   77   68
   23   68  113
```

3.3 Matrix division

To recognize how the two operator / and \ work ;

X = A\B is a solution to A*X = B

X = B/A is a solution to X*A = B

```
>>A=[1,2,3;4,5,6;7,8 0]; B=[1,4,7;2,5,8;3,6,0];
>>X= A\B
ans =
  -0.3333   -3.3333   -5.3333
   0.6667    3.6667    4.6667
   0         -0.0000    1.0000
>> X = B/A
X =
   3.6667   -0.6667    0.0000
   3.3333   -0.3333    0.0000
   4.0000   -2.0000    1.0000
```

3.4 Element-wise operation

You may also want to operate on a matrix element-by-element. To get element-wise behavior appropriate for an array, precede the operator with a dot. There are two important operators here .* and ./

A.*B is a matrix containing the elements of A multiplied by the corresponding elements of B. Obviously A and B must have the same size. The ./ operation is similar but does a division. There is a similar operator .^ which raises each element of a matrix to some power.

```
>> E = [1 2;3 4]
E =
    1    2
    3    4
>> F = [2 3;4 5]
F =
    2    3
    4    5
>> G = E .* F
G =
    2    6
   12   20
```

If you have a square matrix, like E, you can also multiply it by itself as many times as you like by raising it to a given power.

```
>>E^3
ans =
   37   54
   81  118
```

If wanted to cube each element in the matrix, just use the element-by-element cubing.

```
>> E.^3
ans =
    1    8
   27   64
>> A = [1 2 3;4 5 6;7 8 9];
1./A
ans =
   1.0000   0.5000   0.3333
   0.2500   0.2000   0.1667
```

```
    0.1429    0.1250    0.1111
>> A./A
ans =
     1     1     1
     1     1     1
     1     1     1
```

Most elementary functions, such as sin, exp, etc., act element-wise.

```
>> cos(A*pi)
ans =
    -1     1    -1
     1    -1     1
    -1     1    -1
>> exp(A)
ans =
  1.0e+003 *
    0.0027    0.0074    0.0201
    0.0546    0.1484    0.4034
    1.0966    2.9810    8.1031
```

4. The Colon Operator

The colon operator can also be used to create a vector from a matrix. Define:

```
>> A = [1 2 3;4 5 6;7 8 9];
>> B=A(:,1)
B =
     1
     4
     7
```

Note that the expressions before the comma refer to the matrix rows and after the comma to the matrix columns.

```
>> B=A(:,2)
B =
     2
     5
     8
>> B=A(1,:)
B =
```

```
    1     2     3
```

The colon operator is also useful in extracting smaller matrices from larger matrices. If the 4 x 3 matrix C is defined by

```
>> C = [ -1 0 0;1 1 0;1 -1 0;0 0 2 ]
C =
   -1     0     0
    1     1     0
    1    -1     0
    0     0     2
>> D=C(:,2:3)
```

creates the following 4 x 2 matrix:

```
D =
    0     0
    1     0
   -1     0
    0     2
>> D= C(3:4,1:2)
```

Creates a 2 x 2 matrix in which the rows are defined by the 3rd and 4th row of C and the columns are defined by the 1st and 2nd columns of the matrix, C.

```
D =
    1    -1
    0     0
```

5. Referencing elements

The colon is often a useful way to construct these indices.

```
>> A = [1 2 3;4 5 6;7 8 9];
>> A(:,3)=0
```

Evaluated the third column to zero.

```
A =
    1     2     0
    4     5     0
    7     8     0
>> A(:,3)=[]
```

Deleted the third column.

```
A =
    1     2
```

```
    4    5
    7    8
>> A(3,:)=[]
```

Deleted the third row.

```
A =
    1    2
    4    5
>> A(:,3)=5
```

Expand the matrix into 2× 3 matrix, with a the values of the third column equal to 5.

```
A =
    1    2    5
    4    5    5
>> A(3,:)=7:9
```

Expand the matrix into 3× 3 matrix, with a values of the third column equal to 7, 8, 9:

```
A =
    1    2    5
    4    5    5
    7    8    9
```

An array is resized automatically if you delete elements or make assignments outside the current size. (Any new undefined elements are made zero.)

```
>> A(:,5)=10
```

Expand the matrix into 3× 5 matrix, with a values of the fourth column equal to 0 and the last column equal to 10:

```
A =
    1    2    5    0   10
    4    5    5    0   10
    7    8    9    0   10
```

6. Matrix Inverse

The function inv is used to compute the inverse of a matrix. Let, for instance, the matrix A be defined as follows:

```
>> A = [1 2 3;4 5 6;7 8 10]
A =
1   2   3
4   5   6
7   8   10
```

Then,

```
>> B = inv(A)
B =
-0.6667 -1.3333 1.0000
-0.6667 3.6667 -2.0000
1.0000 -2.0000 1.0000
```

The inverse of matrix A can be found by using either A^(-1) or inv(A).

```
>> A=[2 1 1; 1 2 2; 2 1 2]
A =
2 1 1
1 2 2
2 1 2
>> Ainv=inv(A)
Ainv =
2/3 -1/3 0
2/3 2/3 -1
-1 0 1
```

Let's verify the result of A*inv(A).

```
>> A*Ainv
ans =
1 0 0
0 1 0
0 0 1
```

Also let's verify the result of inv(A)*A

```
>> Ainv*A
ans =
1 0 0
0 1 0
0 0 1
```

Note: There are two matrix division symbols in Matlab, / and \ in which

a/b = a*inv(b)

a\b = inv(a)*b.

7. Predefined Matrix

Sometimes, it is often useful to start with a predefined matrix providing only the dimension. A partial list of these functions is:

zeros: matrix filled with 0.

ones: matrix filled with 1.

eye: Identity matrix.

Finally, here are some examples on this special matrices

```
>>A=zeros(2,3)
A =
0 0 0
0 0 0
>>B=ones(2,4)
B =
1 1 1 1
1 1 1 1
>>C=eye(3)
C =
1 0 0
0 1 0
0 0 1
```

8. Other Operations on Matrix

Define a matrix M and examine the effect of each command separately:

```
>>M=[23 0 3;16 8 5;13 2 4;1 10 7]
M =
   23    0    3
   16    8    5
   13    2    4
    1   10    7
```

>>length(M) number of rows in M

```
 4
```

>>size(M) matrix size (rows, columns)

```
 4    3
```

>>find(M>7) finds indices of elements greater than 7.

```
    1
    2
```

3
6
8

>>sum(M) sum of elements in each column

53 20 19

>>max(M) maximum element in each column.

23 10 7

>>min(M) minimum element in each column

1 0 3

>>mean(M) mean of elements in each column

13.2500 5.0000 4.7500

>>sort(M) sorts each column prod(M) product of elements in each column

1 0 3
13 2 4
16 8 5
23 10 7

>>all(M) 1 if all elements nonzero, 0 if any element nonzero

1 0 1

>>abs(M) vector with absolute value of all elements

23 0 3
16 8 5
13 2 4
1 10 7

>>rand returns a random value from 0 to 1.

0.9058

>>rand(2,3) returns a random matrix with 2 rows and 3 columns

0.1270 0.6324 0.2785
0.9134 0.0975 0.5469

>>rand(3) returns a random matrix 3x3

0.9575 0.9706 0.8003
0.9649 0.9572 0.1419
0.1576 0.4854 0.4218

Exercise 1:

Start with a fresh M-file editing window. Write a code to convert the temperature in Celsius into °F and then into °R for every temperature from 0 increasing 15 to 100°C. Combine the three results into one matrix and display them as a table.

Solution:

```
tc = [0:15:100]; % tc is temperature Celsius, tf is temp deg F,
tf = 1.8.*tc + 32; % and tr is temp deg Rankin.
tr = tf + 459.69;
t = [tc',tf',tr'] % combine answer into one matrix
```

The results will be

```
t =
     0         32.0000  491.6900
  15.0000   59.0000  518.6900
  30.0000   86.0000  545.6900
  45.0000  113.0000  572.6900
  60.0000  140.0000  599.6900
  75.0000  167.0000  626.6900
  90.0000  194.0000  653.6900
```

Use vectors with the aid of **interp1** command to find the bubble point of ternary system (Ethanol 40 mol%, Water 20 mol% and Benzene 40 mol%). Knowing that the vapor pressure for three components are calculated by:

Ethanol $P^o_e=\exp(18.5242-3578.91/(T-50.5))$

Water $P^o_w=\exp(18.3036-3816.44/(T-46.13))$

Benzene $P^o_b=\exp(15.9008-2788.51/(T-52.36))$

Where

$K_i= P^o_i /P_t$, $P_t=760$, $y_i =K_i \times x_i$, At Bubble point $\sum yi=\sum Ki \times xi =1$

Solution:

```
T=[60:5:100]+273.15;
Pe=exp(18.5242-3578.91./(T-50.5));
Pw=exp(18.3036-3816.44./(T-46.13));
Pb=exp(15.9008-2788.51./(T-52.36));
Ke=Pe/760; Kw=Pw/760; Kb=Pb/760;
Xe=0.4; Xw=0.2; Xb=0.4;
Ye=Ke*Xe; Yw=Kw*Xw; Yb=Kb*Xb;
```

```
Ys=Ye+Yw+Yb;
A=[T',Ye',Yw',Yb',Ys']
TBp=interp1(Ys,T,1)
```

The output of the above code will be:

```
A =
  333.1500   0.1850   0.0393   0.2060   0.4304
  338.1500   0.2305   0.0494   0.2451   0.5250
  343.1500   0.2852   0.0615   0.2899   0.6366
  348.1500   0.3502   0.0761   0.3409   0.7672
  353.1500   0.4271   0.0935   0.3987   0.9194
  358.1500   0.5176   0.1141   0.4640   1.0958
  363.1500   0.6235   0.1384   0.5373   1.2992
  368.1500   0.7466   0.1668   0.6194   1.5328
  373.1500   0.8890   0.2000   0.7107   1.7997
TBp =
  355.4352
```

Practice Problems

1) Write a program to make a table of the physical properties for water in the range of temperatures from 273 to 323 K.

Exercise 2:

The Density : $\rho = 1200.92 - 1.0056\ T + 0.001084\ T^2$

The conductivity: $K = 0.34 + 9.278 * 10^{-4}\ T$

The Specific heat: $C_P = 0.015539\ (T - 308.2)^2 + 4180.9$

2) Define the 5 x 4 matrix, g.

$$g = \begin{bmatrix} 0.6 & 1.5 & 2.3 & -0.5 \\ 8.2 & 0.5 & -0.1 & -2.0 \\ 5.7 & 8.2 & 9.0 & 1.5 \\ 0.5 & 0.5 & 2.4 & 0.5 \\ 1.2 & -2.3 & -4.5 & 0.5 \end{bmatrix}$$

Find the content of the following matrices and check your results for content using Matlab.

1. a = g(:,2)
2. b = g(4,:)
3. c = g(4:5,1:3)
4. d = g(1:2:5,:)
5. e=sum(g)

Matrix Algebra

1. Introduction

There are a number of common situations in chemical engineering where systems of linear equations appear. There are at least three ways in MATLAB for solving these system of equations;

(1) using matrix **algebra commands** (Also called matrix inverse or Gaussian Elimination method)

(2) using the **solve** command (have been discussed).

(3) using the **numerical** equation solver.

The first method is the preferred, therefore we will explained and demonstrated it. Remember, you always should work in an m-file.

2. Solving Linear Equations Using Matrix Algebra

One of the most common applications of matrix algebra occurs in the solution of linear simultaneous equations. Consider a set of n equations in which the unknowns are x_1, x_2, ... x_n.

$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots\ldots\ldots\ldots a_{1n}x_n = b_1$

$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots\ldots\ldots\ldots a_{2n}x_n = b_2$

$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots\ldots\ldots\ldots a_{3n}x_n = b_3$

. . . . .

$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots\ldots\ldots\ldots a_{nn}x_n = b_n$

where

x_j is the j^{th} variable.

a_{ij} is the constant coefficient of the j^{th} variable in the i^{th} equation.

b_j is constant “right-hand-side” coefficient for equation i.

The system of equations given above can be expressed in the matrix form as.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & : & a_{1n} \\ a_{21} & a_{22} & a_{23} & : & : \\ : & : & : & : & : \\ : & : & : & : & : \\ a_{n1} & a_{n2} & a_{n3} & : & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ : \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ : \\ b_n \end{bmatrix}$$

Or

$AX = b$

Where

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & : & a_{1n} \\ a_{21} & a_{22} & a_{23} & : & a_{2n} \\ : & : & : & : & : \\ : & : & : & : & : \\ a_{n1} & a_{n2} & a_{n3} & : & a_{nn} \end{bmatrix} \qquad b=\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ : \\ b_n \end{bmatrix} \qquad X=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ : \\ x_n \end{bmatrix}$$

To determine the variables contained in the column vector 'x', complete the following steps.

(a) Create the coefficient matrix 'A'. Remember to include zeroes where an equation doesn't contain a variable.

(b) Create the right-hand-side column vector 'b' containing the constant terms from the equation. This must be a ***column*** vector, ***not*** a ***row***.

(c) Calculate the values in the 'x' vector by left dividing 'b' by 'A', by typing x = A\b.

Note: this is different from x = b/A.

As an example of solving a system of equations using the matrix inverse method, consider the following system of three equations.

$x_1 - 4x_2 + 3x_3 = -7$

$3x_1 + x_2 - 2x_3 = 14$

$2x_1 + x_2 + x_3 = 5$

These equations can be written in matrix format as;

$$\begin{bmatrix} 1 & -4 & 3 \\ 3 & 1 & -2 \\ 2 & 1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} -7 \\ 14 \\ 5 \end{bmatrix}$$

To find the solution of the following system of equations type the code.

```
A =[1,-4, 3; 3, 1, -2; 2, 1, 1]
B = [ -7;14; 5]
x = A\B
```

the results will be results in

```
 x = [ 3
       1
      -2 ]
```

in which $x_1=3$, $x_2=1$, $x_3= -2$

to extract the value of each of x_1, x_2, x_3 type the command:

```
x1=x(1),x2=x(2),x3=x(3)
```

The results will be:

```
x1 =
    3
x2 =
    1
x3 =
   -2
```

Exercise 1:

For the following separation system, we know the inlet mass flow rate (in Kg/hr) and the mass fractions of each species in the inlet flow (F) and each outlet flow (F1, F2 and F3). We want to calculate the unknown mass flow rates of each outlet stream.

Solution:

If we define the unknowns as x1=F1, x2=F2 , x3=F3

and set up the mass balances for

1. the total mass flow rate

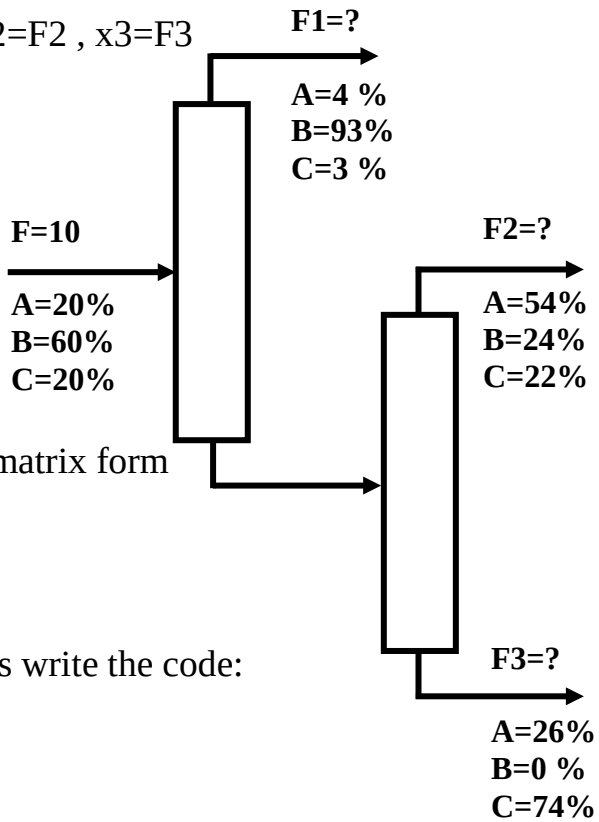
x1 +x2 +x3 =10

2. the mass balance on species 1

0.04x1+0.54x2+0.26x3=0.2*10

3. the mass balance on species 2

0.93x1+0.24X2 =0.6*10



These three equations can be written in matrix form

$$\begin{bmatrix} 1 & 1 & 1 \\ 0.04 & 0.54 & 0.26 \\ 0.93 & 0.24 & 0 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} 10 \\ 2 \\ 6 \end{bmatrix}$$

To find the values of unknown flow rates write the code:

```
A=[1, 1, 1; .04, .54, .26; .93, .24 ,0];
B=[10; .2*10; .6*10];
X=A\B;
F1=X(1),F2=X(2),F3=X(3)
```

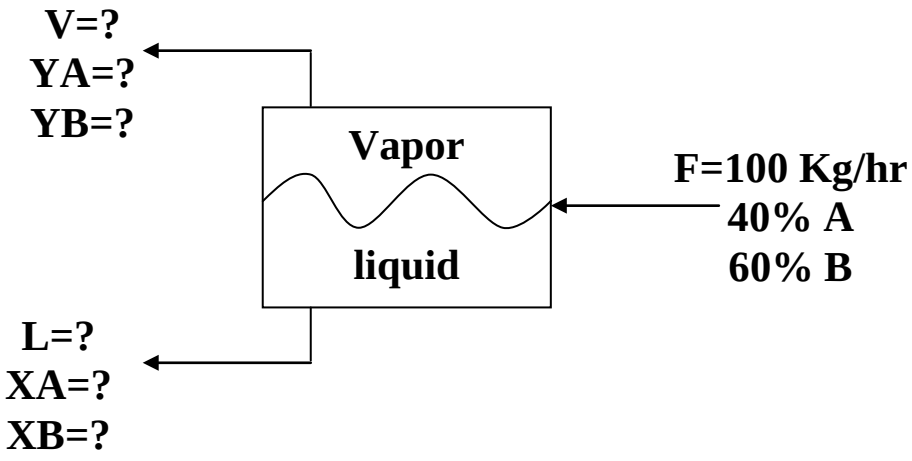
The results will be:

```
F1 =
   5.8238
F2 =
   2.4330
F3 =
   1.7433
```

Write a program to calculate the values of XA,XB,YA,YB, L and V for the vapor liquid separator shown in fig.

If you know:

XA+XB=1
YA+YB=1
YA=KA*XA=1.9XA
YB=KB*XB=0.6XB



Solution:

```
A=[1,1,0,0;0,0,1,1;-1.9,0,1,0;0,-0.6,0,1];
B=[1;1;0;0];
X=A\B;
xa=X(1),xb=X(2),ya=X(3),yb=X(4)
a=[xa,ya;xb,yb];
b=[.4*100;.6*100];
x=a\b;
L=x(1),V=x(2)
```

Gives the results

```
xa =
   0.3077
xb =
   0.6923
ya =
   0.5846
yb =
   0.4154
L =
  66.6667
V =
  33.3333
```

Exercise 2:

Xylene, styrene, toluene and benzene are to be separated with the array of distillation columns that is shown below. Write a program to calculate the amount of the streams D, B, D1, B1, D2 and B2 also to calculate the composition of streams D and B.

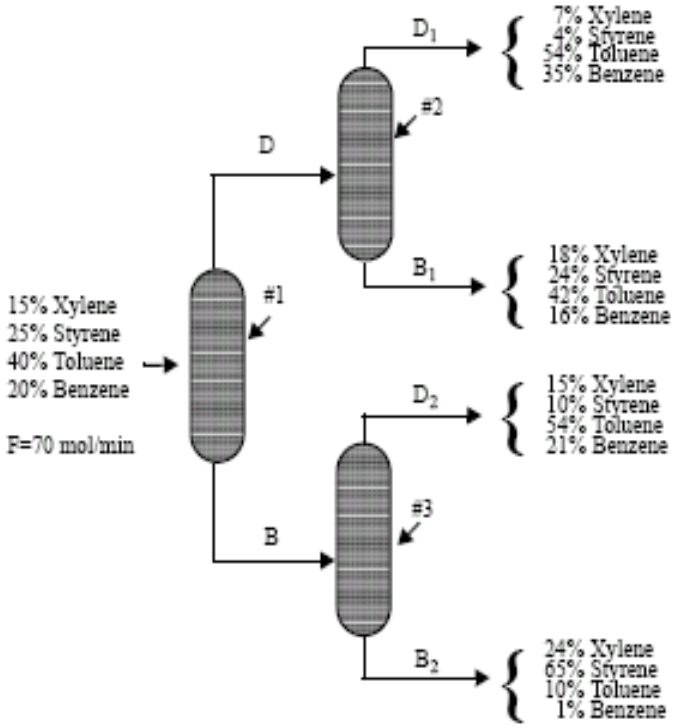


Figure A1 Separation Train

Solution:

By making material balance on individual components on the overall separation train yield the equation set

Xylene: 0.07D1+ 0.18B1+ 0.15D2+ 0.24B2= 0.15× 70

Styrene: 0.04D1+ 0.24B1+ 0.10D2+ 0.65B2= 0.25× 70

Toluene: 0.54D1+ 0.42B1+ 0.54D2+ 0.10B2= 0.40× 70

Benzene: 0.35D1+ 0.16B1+ 0.21D2+ 0.01B2= 0.20× 70

Overall material balances and individual component balances on column 2 can be used to determine the molar flow rate and mole fractions from the equation of stream D.

Molar Flow Rates: D = D1 + B1

Xylene: XDxD = 0.07D1 + 0.18B1

Styrene: XDsD = 0.04D1 + 0.24B1

Toluene: XDtD = 0.54D1 + 0.42B1

Benzene: XDbD = 0.35D1 + 0.16B1

where

XDx = mole fraction of Xylene.

XDs = mole fraction of Styrene.

XDt = mole fraction of Toluene.

XDb =mole fraction of Benzene.

Similarly, overall balances and individual component balances on column 3 can be used to determine the molar flow rate and mole fractions of stream B from the equation set.

Molar Flow Rates: B = D2 + B2

Xylene: XBxB = 0.15D2 + 0.24B2

Styrene: XBsB = 0.10D2 + 0.65B2

Toluene: XBtB = 0.54D2 + 0.10B2

Benzene: XBbB = 0.21D2 + 0.01B2

where F, D, B, D1, B1, D2 and B2 are the molar flow rates in mol/min.

Now type the following code in command window

```
A=[0.07, 0.18, 0.15, 0.24; 0.04, 0.24, 0.10, 0.65; 0.54, 0.42, 0.54, 0.1;0 .35, 0.16, 0.21, 0.01];
B=[0.15*70; 0.25*70; 0.4*70; 0.2*70];
X=A\B;
D1=X(1),B1=X(2),D2=X(3),B2=X(4),
D=D1+B1
B=D2+B2
XDx=(.07*D1+.18*B1)/D
XDs=(.04*D1+.24*B1)/D
XDt=(.54*D1+.42*B1)/D
XDb=(.35*D1+.16*B1)/D
XBx=(.15*D2+.24*B2)/B
XBs=(.1*D2+.65*B2)/B
XBt=(.54*D2+.1*B2)/B
XBb=(.21*D2+.01*B2)/B
```

The results will be

```
D1 =
   26.2500
B1 =
   17.5000
D2 =
    8.7500
B2 =
   17.5000
```

```
D =
   43.7500
B =
   26.2500
XDx =
    0.1140
XDs =
    0.1200
XDt =
    0.4920
XDb =
    0.2740
XBx =
    0.2100
XBs =
    0.4667
XBt =
    0.2467
XBb =
    0.0767
```

Exercise 4:

Balance the following chemical equation:

$x1\ CH_4 + x2\ O_2 \rightarrow x3\ CO_2 + x4\ H_2O$

Solution:

There are three elements involved in this reaction: carbon (C), hydrogen (H), and oxygen (O). A balance equation can be written for each of these elements:

Carbon (C): $1 \cdot x1 + 0 \cdot x2 = 1 \cdot x3 + 0 \cdot x4$

Hydrogen (H): $4 \cdot x1 + 0 \cdot x2 = 0 \cdot x3 + 2 \cdot x4$

Oxygen (O): $0 \cdot x1 + 2 \cdot x2 = 2 \cdot x3 + 1 \cdot x4$

Re-write these as homogeneous equations, each having zero on its right hand side:

$x1 - x3 = 0$

$4x1 - 2x4 = 0$

$2x2 - 2x3 - x4 = 0$

At this point, there are three equations in four unknowns. To complete the system, we define an auxiliary equation by arbitrarily choosing a value for one of the coefficients:

$x4 = 1$

To solve these four equations write the code:

```
A =[1,0,-1,0;4,0,0,-2;0,2,-2,-1;0,0,0,1];
B=[0,0,0,1];
X=A\B
```

The result will be

```
X =
   0.5000
   1.0000
   0.5000
   1.0000
```

Finally, the stoichiometric coefficients are usually chosen to be integers.
Divide the vector X by its smallest value:

```
X =X/min(X)
X =
    1
    2
    1
    2
```

Thus, the balanced equation is

$CH4 + 2O2 \rightarrow CO2 + 2H2O$

Exercise 5:

Balance the following chemical equation:

$x1\ P_2I_4 + x2\ P_4 + x3\ H_2O \rightarrow x4\ PH_4I + x5\ H_3PO_4$

Solution:

We can easily balance this reaction using MATLAB:

```
A = [2 4 0 -1 -1
4 0 0 -1 0
0 0 2 -4 -3
0 0 1 0 -4
0 0 0 0 1];
B= [0;0;0;0;1];
X = A\B;
X =
   0.3125
   0.4063
   4.0000
   1.2500
   1.0000
```

We divide by the minimum value (first element) of **x** to obtain integral coefficients:

```
X=X/min(X)
```

```
X =
    1.0000
    1.3000
   12.8000
    4.0000
    3.2000
```

This does not yield integral coefficients, but multiplying by 10 will do the trick:

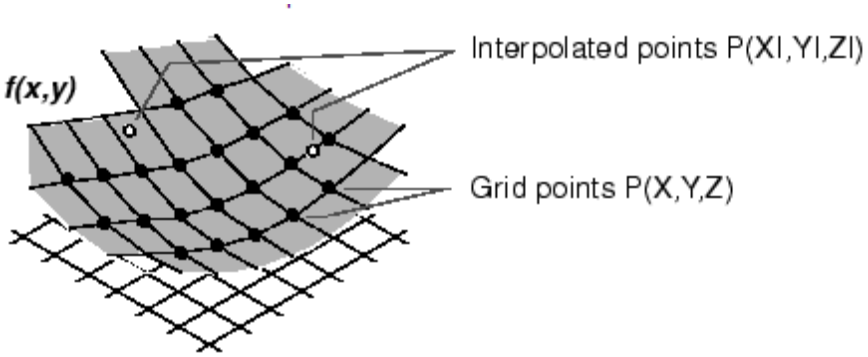
```
x = x * 10
X =
   10
   13
  128
   40
   32
```

The balanced equation is

$10\ P_2I_4 + 13\ P_4 + 128\ H_2O \rightarrow 40\ PH_4I + 32\ H_3PO_4$

3.Two-Dimensional Interpolation

The **interp2** command performs two-dimensional interpolation between data points. It finds values of a two-dimensional function underlying the data at intermediate points>



Its most general form is:

Zi = interp2(X, Y, Z, Xi, Yi)

Zvector = interp2(X, Y, Z, Xvector, Yvector)

Note: the number of elements of the X vector and Z matrix rows must be the same, while the number of elements of the Y vector and Z matrix columns must be the same.

Exercise 6:

Calculate the values of z corresponding to (x,y)=(1.3, 1.5) and (1.5,2.3) from data as following:

x=1, 2

y= 1, 2, 3

z = 10 20

40 50

70 80

Solution

```
x = [1 2];
y = [1 2 3];
z = [10 20
     40 50
     70 80];
z1 = interp2(x,y,z,1.3,1.5)
```

The results will be

```
z1 =
   28000
```

To interpolate a vector of x, y point repeat the same code with small change:

```
z12 = interp2(x,y,z,[1.3,1.5],[1.5,2.3])
z12 =
   28.0000   54.0000
```

Practice Problems

1) Solve each of these systems of equations by using the matrix inverse method.

A) $r + s + t + w = 4$
$2r - s + w = 2$
$3r + s - t - w = 2$
$r - 2s - 3t + w = -3$

B) $x_1+2x_2-x_3=3$
$3x_1-x_2+2x_3=1$
$2x_1-2x_2+3x_3=2$
$x_1-x_2+x_4=-1$

C) $5x_1+3x_2+7x_3-4=0$
$3x_1+26x_2-2x_3-9=0$
$7x_1+2x_2+10x_3-5=0$

D) $2x_1 + x_2 - 4x_3 + 6x_4 + 3x_5 - 2x_6 = 16$

$-x_1 + 2x_2 + 3x_3 + 5x_4 - 2x_5 = -7$

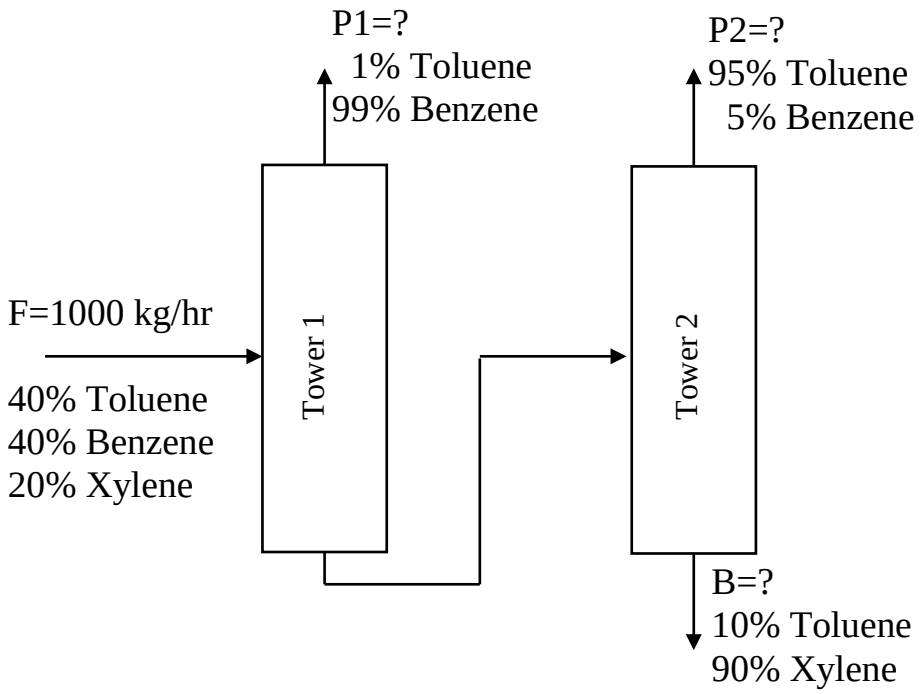
$x_1 - 2x_2 - 5x_3 + 3x_4 + 2x_5 + x_6 = 1$

$4x_1 + 3x_2 - 2x_3 + 2x_4 + x_6 = -1$

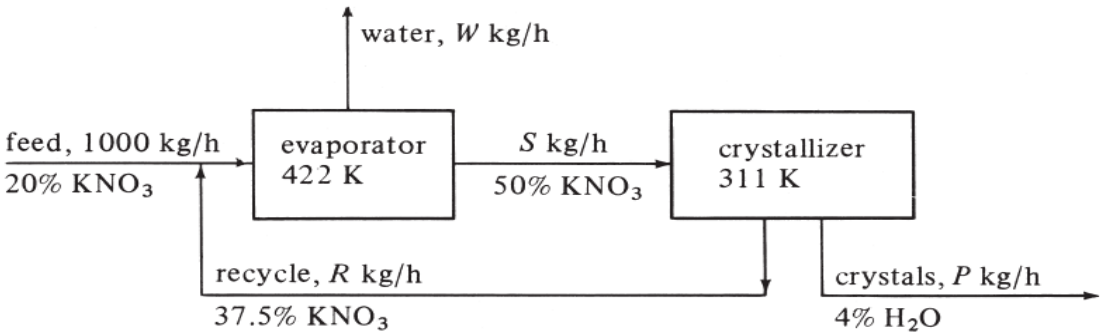
$3x_1 + x_2 - x_3 + 4x_4 + 3x_5 + 6x_6 = -11$

$5x_1 + 2x_2 - 2x_3 + 3x_4 + x_5 + x_6 = 5$

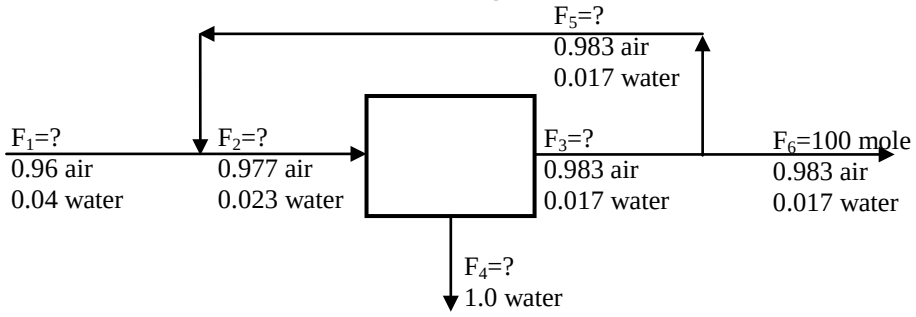
2) For the following figure calculate the values of the unknown flow rates A, B, C by using matrix inverse method.



3) In a process producing KNO3 salt, 1000 kg/h of a feed solution containing 20 wt % KNO3 is fed to an evaporator, which evaporates some water at 422 K to produce a 50 wt % KNO3 solution. This is then fed to a crystallizer at 311 K, where crystals containing 96 wt % KNO3 is removed. The saturated solution containing 37.5 wt % KNO3 is recycled to the evaporator. Write a code to calculate the amount of recycle stream R in kg/h and the product stream of crystals P in kg/h.



4) For the following figure, write a program to calculate the values of the unknown flow rates F1, F2, F3, F4 and F5 using matrix inverse method.



5) Use **interp2** to calculate the enthalpy, internal energy and specific volume of superheated steam when pressure is 2 bar ant temperature is 120 ^{o}C.

P(bar) ($T_{sat.}$°C)		Sat'd Water	Sat'd Steam	Temperature (°C)→ 50	75	100	150	200	250	300	350
0.0	$\hat{H}$	—	—	2595	2642	2689	2784	2880	2978	3077	3177
(—)	$\hat{U}$	—	—	2446	2481	2517	2589	2662	2736	2812	2890
	$\hat{V}$	—	—	—	—	—	—	—	—	—	—
0.1	$\hat{H}$	191.8	2584.8	2593	2640	2688	2783	2880	2977	3077	3177
(45.8)	$\hat{U}$	191.8	2438.0	2444	2480	2516	2588	2661	2736	2812	2890
	$\hat{V}$	0.00101	14.7	14.8	16.0	17.2	19.5	21.8	24.2	26.5	28.7
0.5	$\hat{H}$	340.6	2646.0	209.3	313.9	2683	2780	2878	2979	3076	3177
(81.3)	$\hat{U}$	340.6	2484.0	209.2	313.9	2512	2586	2660	2735	2811	2889
	$\hat{V}$	0.00103	3.24	0.00101	0.00103	3.41	3.89	4.35	4.83	5.29	5.75
1.0	$\hat{H}$	417.5	2675.4	209.3	314.0	2676	2776	2875	2975	3074	3176
(99.6)	$\hat{U}$	417.5	2506.1	209.2	313.9	2507	2583	2658	2734	2811	2889
	$\hat{V}$	0.00104	1.69	0.00101	0.00103	1.69	1.94	2.17	2.40	2.64	2.87
5.0	$\hat{H}$	640.1	2747.5	209.7	314.3	419.4	632.2	2855	2961	3065	3168
(151.8)	$\hat{U}$	639.6	2560.2	209.2	313.8	418.8	631.6	2643	2724	2803	2883
	$\hat{V}$	0.00109	0.375	0.00101	0.00103	0.00104	0.00109	0.425	0.474	0.522	0.571
10	$\hat{H}$	762.6	2776.2	210.1	314.7	419.7	632.5	2827	2943	3052	3159
(179.9)	$\hat{U}$	761.5	2582	209.1	313.7	418.7	631.4	2621	2710	2794	2876
	$\hat{V}$	0.00113	0.194	0.00101	0.00103	0.00104	0.00109	0.206	0.233	0.258	0.282

Condition

1. If Statement

What are you doing if you want certain parts of your program to be executed **only** in limited circumstances? The way to do that is by putting the code within an "if" statement. The most basic structure for "if" statement is as following:

```
if  (relation)
   (matlab commands)
end
```

More complicated structures are also possible including combinations like the following:

```
if  (relation)
   (matlab commands)
elseif  (relation)
   (matlab commands)
elseif (relation)
   (matlab commands)
.
.
else
   (matlab commands)
end
```

Standard comparisons can be made by using relations. The relational operators in MATLAB are;

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

For example, the following code will set the variable j to be –1 if a less than b:

```
a = 2;
b = 3;
```

```
if a<b
   j = -1
end
```

Additional statements can be added for more decision. The following code sets the variable j to be 2 when a greater than b.

```
a = 4;
b = 3;
if a < b
   j = -1
elseif a > b
   j = 2
end
```

The **else** statement provides all that will be executed if no other condition is met. The following code sets the variable j to be 3 when a is not greater and less than b.

```
a = 4;
b = 4;
if a<b
   j = -1
elseif a>b
   j = 2
else
   j = 3
end
```

Matlab allows you to write together multiple condition expressions using the standard logic operators, "&" *(and)*, "¦" *(or)*, and "˜" *(not)*.

For example to check if *a* is less than *b* and at the same time *b* is greater than or equal to *c* you would use the following commands:

```
if a < b & b >= c
  Matlab commands
end
```

For example

```
x=1; y=0;
if x < 0 & y < 0
```

```
z = -x * y
elseif x == 0 ¦ y == 0
z = 0
else
z = x^2
end
```

The output of the above code is

```
z=
    0
```

2.Loop

Many programs require iteration, or repetitive execution of a block of statements. MATLAB has two loop statements: **for** and **while** statements. All of the loop structures in matlab are started with a keyword such as "for" or "while" and all of them end with the word "end".

2.1 For loop

If you want to repeat a certain commands in a predetermined way, you can use the "for" loop. The "for" loop will loop around some statement, and you must tell Matlab where to start and where to end. Basically, you give a vector in the "for" statement, and Matlab will loop through for each value in the vector:

The for statements have the following structure:

```
For variable = expression
      statement
end
```

For example, a simple loop will go around four times:

```
for j=1:4
j
end
j =
    1
j =
    2
j =
    3
```

```
j =
    4
```

For another example, if we define a vector and later want to change it elements, we can step though and change each individual entry:

```
v = [1:3:10];
for j=1:4
v(j) = j;
end
>>v
v =
    1    2    3    4
```

Matrices may also be constructed by loop. Here is an example using two "for" loops.

```
for i=1:10
for j=1:10
t(i,j) = i*j;
end
end
```

Notice that there isn't any output, since the only line that would produce any (**t(i,j) = i/j;**) ends in a semi-colon. Without the semi-colon, Matlab would print the matrix t 100 times!

Example 1

Find summations of even numbers between 0 and 100

```
sum = 0;
for i = 0 : 2 : 100
      sum = sum + i;
end
sum
sum =
      2550
```

Note: if the increment is one you may write the first and the last limit without writing a step. In this case, the for-line above would become **for i = 0:100**.

warning: If you are using complex numbers, you cant use i as the loop index.

2.2 While Loop

If you don't like the "for" loop, you can also use a "while" loop. It is sometimes necessary to repeat statements based on a condition rather than a fixed number of times. The "while" loop repeats a sequence of commands as long as some condition is met. The general form of a while loop is

```
while relation
    statements
end
```

Example 2

```
x=1;
while x < 10
x = x+1;
end
>> x
x =
   10
```

The statements will be repeatedly executed as long as the relation remains true.

Example 3

```
x=10;
while x > 1
x = x/2;
end
>>x
x =
   0.6250
```

The condition is evaluated before the body is executed, so it is possible to get zero iteration when x equal to 1 or less.

Example 4

```
sum = 0;
x = 1;
while x < 4
sum = sum + 1/x;
x = x + 1;
```

```
end
>>sum
sum =
   1.8333
```

2.3 Break statement

It's often a good idea to limit the number of repetitions, to avoid infinite loops (as could happen in above example if x=Inf). This can be done using **break**. A break can be used whenever you need to stop the loop (break statement can be used to stop both for and while loops in same way), for example.

```
n = 0;
x=100
while x > 1
x = x/2;
n = n+1;
if n > 50
break
end
end
>> x
x =
   0.7813
```

A break immediately jumps execution to the first statement after the loop.

Note: When using programs with loops, you should also know how to stop a program in the middle of its execution, such as in the case of your programs contains an infinite loop. During a program is running, the **Ctrl+C** command will stop it.

Exercise 1:

Use loop in 20 iteration to calculate x in equation x=2sin(x)?

Solution:

```
format long
x=1
for i=1:20
   x=2*sin(x)
end
```

The results will be:

x = 1
x = 1.682941969615793
x = 1.987436530272152
x = 1.828907552623580
x = 1.933747642340163
x = 1.869706153630775
x = 1.911316179125262
x = 1.885162348212226
x = 1.901985209663949
x = 1.891312851651419
x = 1.898145620030117
x = 1.893795924017278
x = 1.896575152213355
x = 1.894803506466833
x = 1.895934551140671
x = 1.895213162228076
x = 1.895673549670181
x = 1.895379846173585
x = 1.895567260289186
x = 1.895447688999369
x = 1.895523983862163

You can see the convergence through the digits after the dot. The best guess for x is

x = 1.895523983862163

Exercise 2:

Use loop to find the bubble point of ternary system (Ethanol 40 mol%, Water 20 mol% and Benzene 40 mol%). Knowing that the vapor pressure for three components are calculated by:

Ethanol $P^o_e=\exp(18.5242-3578.91/(T-50.5))$

Water $P^o_w=\exp(18.3036-3816.44/(T-46.13))$

Benzene $P^o_b=\exp(15.9008-2788.51/(T-52.36))$

Where

$K_i= P^o_i /P_t$

$P_t=760$

$y_i =K_i \times x_i$

At Bubble point $\sum yi=\sum Ki \times xi =1$

Solution:

```
Xe=.4;
Xw=.2;
Xb=.4;
for T=273.15:.01:450;
Pe=exp(18.5242-3578.91/(T-50.5));  % Vapor pressure of Ethanol
Pw=exp(18.3036-3816.44/(T-46.13)); % Vapor pressure of Water
Pb=exp(15.9008-2788.51/(T-52.36)); % Vapor pressure of Benzene
% evaluation of equilibrium constants
Ke=Pe/760; % Ethanol
Kw=Pw/760; % Water
Kb=Pb/760; % Benzene
% The condition to find the boiling point
sum=Ke*Xe+Kw*Xw+Kb*Xb;
if sum>1
break
end
end
T
```

Gives the result

```
T =
  355.5300
```

Exercise 3:

Given that the vapor pressure of methyl chloride at 333.15 K is 13.76 bar, write a code to calculate the molar volume of saturated vapor at these conditions using Redlich/Kwong equation. Knowing;

$a=0.42748*R^2T_c^{2.5}/P_c$

$b=0.08664*RT_C/P_C$

$V_{i+1=}(RT/P)+b-(a*(V_i-b))/(T^{1/2}PV_i(V_i+b))$

R=83.14 , Tc= 416.3 k , Pc= 66.8 bar

Solution:

```
T=333.15; Tc=416.3;
P=13.76; Pc=66.8;R=83.14;
a=0.42748*R^2*Tc^2.5/Pc;
b=0.08664*R*Tc/Pc;
```

```
V(1)=R*T/P;
for i=1:1:10
V(i+1)=(R*T/P)+b-(a*(V(i)-b))/(T^.5*P*V(i)*(V(i)+b));
end
    V'
```

The program results as follows

```
ans =
1.0e+003 *
   2.0129
   1.7619
   1.7219
   1.7145
   1.7131
   1.7129
   1.7128
   1.7128
   1.7128
   1.7128
   1.7128
```

Note that after the 7 th trail the value of V_i is constant and further trails does not have any effect on its value.

Exercise 4:

A simple force balance on a spherical particle reaching terminal velocity in a fluid is given by; $V_t = \sqrt{\frac{4g(\rho_P - \rho)Dp}{3C_D\rho}}$

Where

V_t : Terminal velocity in m/s

g : Acceleration of gravity

p_p: Particle density

D_p: The diameter of the spherical particle in m

C_D: Dimensionless drag coefficient.

The drag coefficient on a spherical particle at terminal velocity varies with Reynolds number (Re) as followings:

C_D=24/Re for Re< 0.1

C_D=24*(1+0.14* Re^0.7)/Re for 0.1=<Re=< 1000

C_D=0.44 for 1000<Re=< 350000

C_D=0.19-8*10^4/Re for 350000 <Re

Where

Re =(D_p v_tp)/μ

g=9.80665 m/s^2

ρ=994.6 kg/m^3

p_p=1800 kg/m^3

μ=8.931×10^{-4} kg/m.s.

D_p=0.000208 m

Calculate the terminal velocity of spherical particle?

Solution:

The above problem cannot be solved without using trial and error method. Therefore you must assume a velocity to calculate the C_D which is a important to calculate new velocity. Write the following code:

```
Format long
g=9.80665; p=994.6; pp=1800; mu=8.931e-4; Dp=0.000208;
vt=1;
Velocity(1)=vt;
for m=1:1:20
Re=(Dp*vt*p)/mu;
if Re < 0.1
CD=24/Re;
elseif Re >= 0.1 & Re <= 1000
CD=24*(1+0.14*Re^0.7)/Re;
elseif Re >1000  &  Re <= 350000
CD=0.44;
elseif Re>350000
CD=0.19-8*10^4/Re;
end
vt=((4*g*(pp-p)*Dp)/(3*CD*p))^.5;
Velocity(m+1)=vt;
if abs (Velocity(m+1)-Velocity(m))<.0001
   break
end
end
Velocity'
```

This code gives the result:

```
ans =
   1.000000000000000
   0.053845727997707
   0.025084090993404
   0.018981846393486
   0.017008369140870
   0.016271175897107
   0.015980109059108
   0.015862612988699
   0.015814754689716
```

Note: the loop will stop when the desired accuracy is reached

Practice Problems

1) Write a computer program to calculate molar volume and compressibility factor for gaseous ammonia by using of the Redlich-Kwong equation of state. The equations and variables are listed below.

$$P = \frac{RT}{(V-b)} - \frac{a}{V(V+b)\sqrt{T}} \qquad a = 0.42747\left(\frac{R^2 T_C^{5/2}}{P_C}\right) \qquad b = 0.08664\left(\frac{RT_C}{P_C}\right)$$

The variables are defined by

P= 56 atm

V= molar volume in L/g-mol

T=450 K

R= gas constant (R = 0.08206 atm·L/g-mol·K)

Tc= the critical temperature (405.5 K for ammonia)

Pc= the critical pressure (111.3 atm for ammonia)

Compressibility factor is given by

$$Z = \frac{PV}{RT}$$

2D Graphics

One of 'matlab' most powerful features is the ability to create graphic plots. There are so many methods of generating graphics in Matlab.

1. X-Y Plots

A Cartesian or x,y plot is based on plotting the x,y data pairs from the specified vectors. Clearly, the vectors x and y must have the same number of elements. Given a vector x of *x*-coordinates x_1 through x_n and a vector y of *y*-coordinates y_1 through y_n, plot(x,y) graphs the points (x_1,y_1) through (x_n,y_n).

For example, to plot the two dimensions corresponding to (0,0), (1,1), (4,2) , (5,1) and (0,0) points .

```
x=[0 1 4 5 0];
y=[0 1 2 -1 0];
plot(x,y)
```

The command plot(x,y) opens a graphics window and draws an *x*-*y* plot of the elements of *x* versus the elements of *y*.

To plot the graph of $y=x^3$ on the interval [2,2], first define a row vector whose components range from -2 to 2 in increments of .05. Then define a vector y; of the same size as x whose components are the cubes of the components of x. Finally use the plot function to display the graph.

```
x=-2:.05:2;
y=x.^3;
plot(x,y)
```

You can, also for example, draw the graph of the sine function over the interval -4 to 4 with the following commands:

```
x = -4:.01:4;
y = sin(x);
plot(x,y)
```

The resulting plot is shown in figure 1.

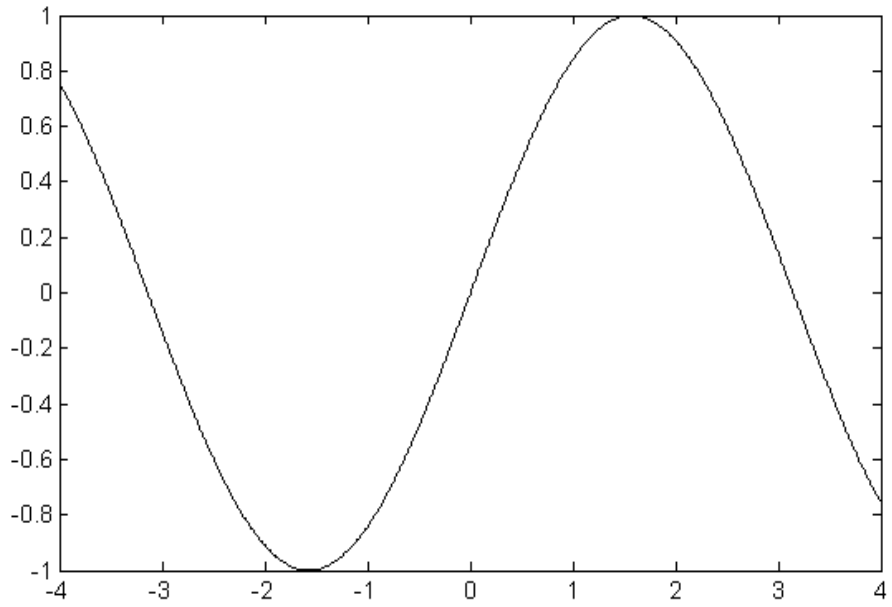


Figure 1. Plotting sine over the interval -4 to 4

Plots of parametrically defined curves can also be made. Try, for example,

```
t=0:.001:2*pi;
x=cos(3*t);
y=sin(2*t);
plot(x,y)
```

The resulting plot is shown in figure 2.

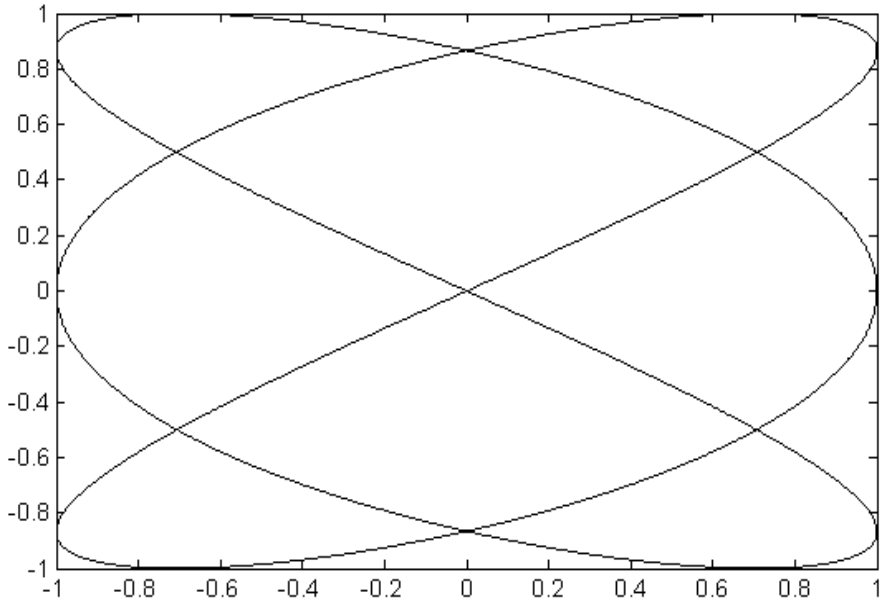


Figure 2. Plotting parametrically curves

2. Grid

A dotted grid may be added to the plot by:

grid on

This grid can be removed using grid off.

grid off

3. Controlling Axes

Once a plot has been created in the graphics window you may wish to change the range of *x* and *y* values, this can be done by using the command "axis" as follows:

```
x =0:.01:1;
y =sin(3*pi*x);
plot(x,y)
axis([-0.5 1.5 -1.2 1.2])
```

The axis command has four parameters, the first two are the minimum and maximum values of *x* to use on the axis and the last two are the minimum and maximum values of *y*.

4. Plotting Line Style

Matlab connects a line between the data pairs described by the vectors. You may wish to present data points and different type of connecting lines between these points. Data points can be described by a variety of characters such as (. , + , * , o and x .).

Various line types, plot symbols and colors may be obtained with plot(x,y,S) where S is a character string made from one element from any or all the following 3 columns:

Character color	Character symbol	character line style
b blue .	. point	- solid
g green	o circle	: dotted
r red	x x-mark	-. dashdot
c cyan	+ plus	-- dashed
m magenta	* star	
y yellow	s square	
k black	d diamond	
	v triangle (down)	
	^ triangle (up)	
	< triangle (left)	
	> triangle (right)	
	p pentagram	
	h hexagram	

For example,

plot(x,y,'k+:') plots a black dotted line with a plus at each data point.

plot(x,y,'bd') plots blue diamond at each data point but does not draw any line.

plot(x,y,'y-',x,y,'go') plots the data twice, with a solid yellow line interpolating green circles at the data points.

5. Annotation

The last step before printing or saving a plot is usually to put a title and label the axes. You can put labels, titles, and text on a plot by using the commands:

xlabel('text')

ylabel('text')

zlabel('text')

title('text')

text(x,y,'text') places text at position x,y

gtext('text') use mouse to place text

For example the graphs can be given titles, axes labeled, and text placed within the graph with the following commands;

```
t = 0:0.02*pi:2*pi;
plot(t,sin(t))
xlabel('time')
ylabel('amplitude')
title('Simple Harmonic Oscillator')
```

The resulting plot is shown in figure 3.

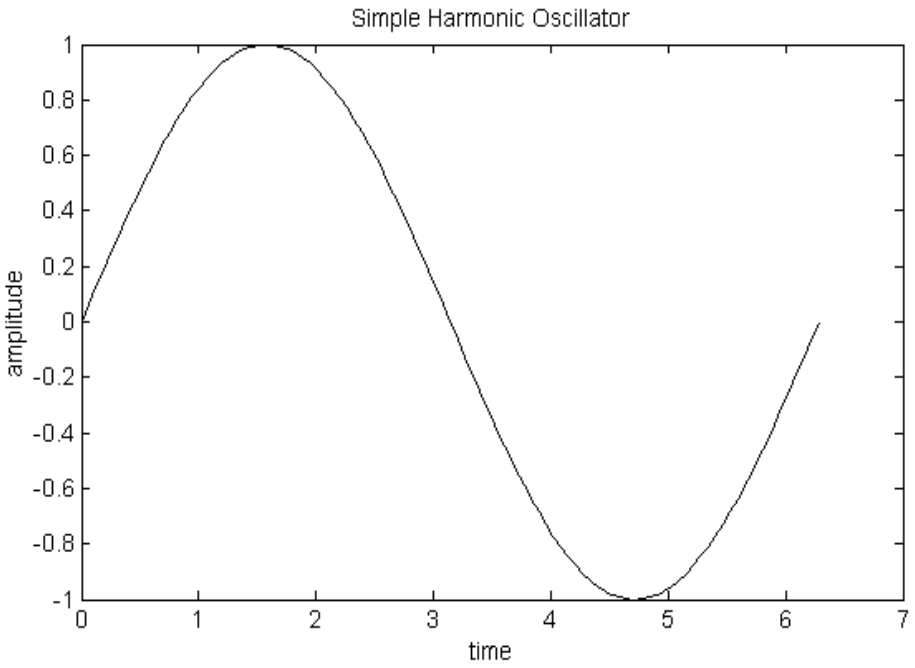


Figure 3. Using annotation in plot

6. Multi–plots

There are two ways to make multiple plots on a single graph are illustrated by.

6.1 Multiple Data Vectors Plots

You can create multiple graphs by using multiple data vectors, for example:

```
x=0:.01:2*pi;
y1=sin(x);
y2=sin(2*x);
y3=sin(4*x);
plot(x,y1,x,y2,x,y3)
```

The re resulting plot is shown in figure 4.

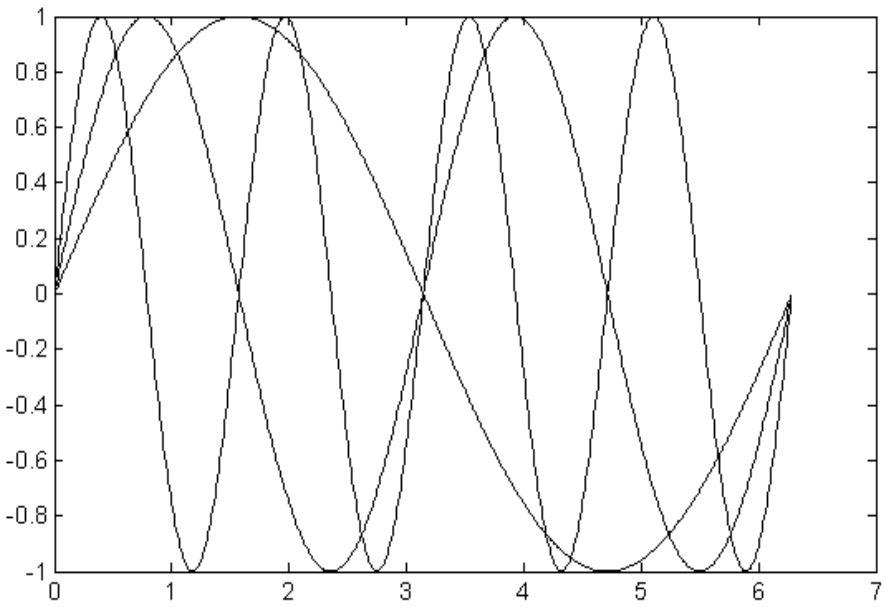


Figure 4. Multiple plots of sine vectors

6.2 Multiple Plots with Hold

Another way is with hold on. The command hold on holds the old graph when the new one is plotted. The axes may, however, become rescaled. "hold on" holds the current picture; "hold off" releases it (but does not clear the window).

Here is an example. Suppose we want to compare the log function with the square root function graphically. We can put them on the same plot. By default, both plots will appear in blue, so we will not know which log. We could make them different colors using options. Here is the final answer,

```
x=1:100;
y=log(x);
z=sqrt(x);
```

```
plot(x,y,'r'); % plot log in red
hold on;
plot(x,z,'b'); % plot log in blue
hold off;
```

The resulting plot is shown in figure 5.

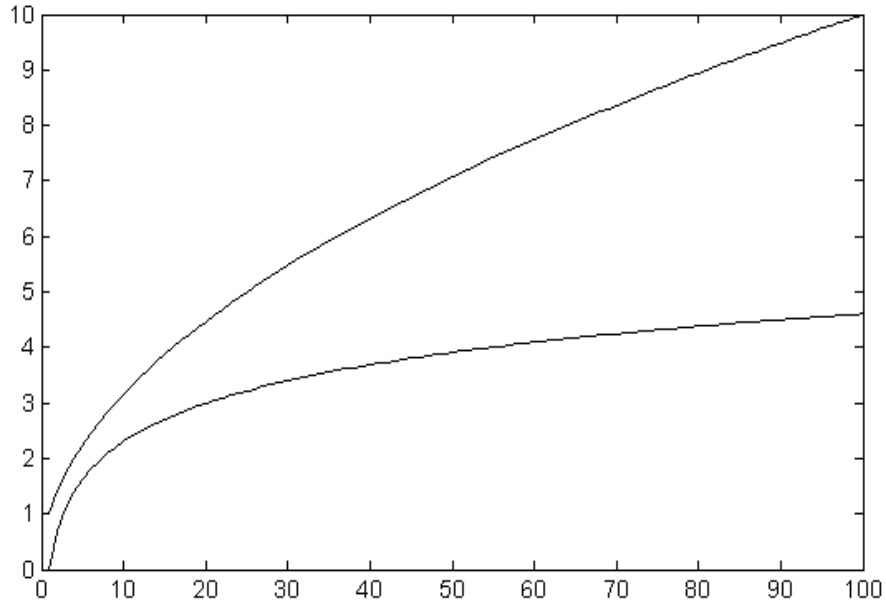


Figure 5. Multiple plots with hold

The "hold" command will remain active until you turn it off with the command 'hold off'.

7. Subplot

The graphics window may be split into an *m by n* array of smaller windows into which we may plot one or more graphs. The windows are numbered 1 to *m by n* row–wise, starting from the top left. All of plot properties such as hold and grid work on the current subplot individually.

```
x=0:.01:1;
subplot(221), plot(x,sin(3*pi*x))
xlabel('x'),ylabel('sin (3pix)')
subplot(222), plot(x,cos(3*pi*x))
xlabel('x'),ylabel('cos (3pix)')
subplot(223), plot(x,sin(6*pi*x))
xlabel('x'),ylabel('sin (6pix)')
subplot(224), plot(x,cos(6*pi*x))
xlabel('x'),ylabel('cos (6pix)')
```

The re resulting plot is shown in figure 6.

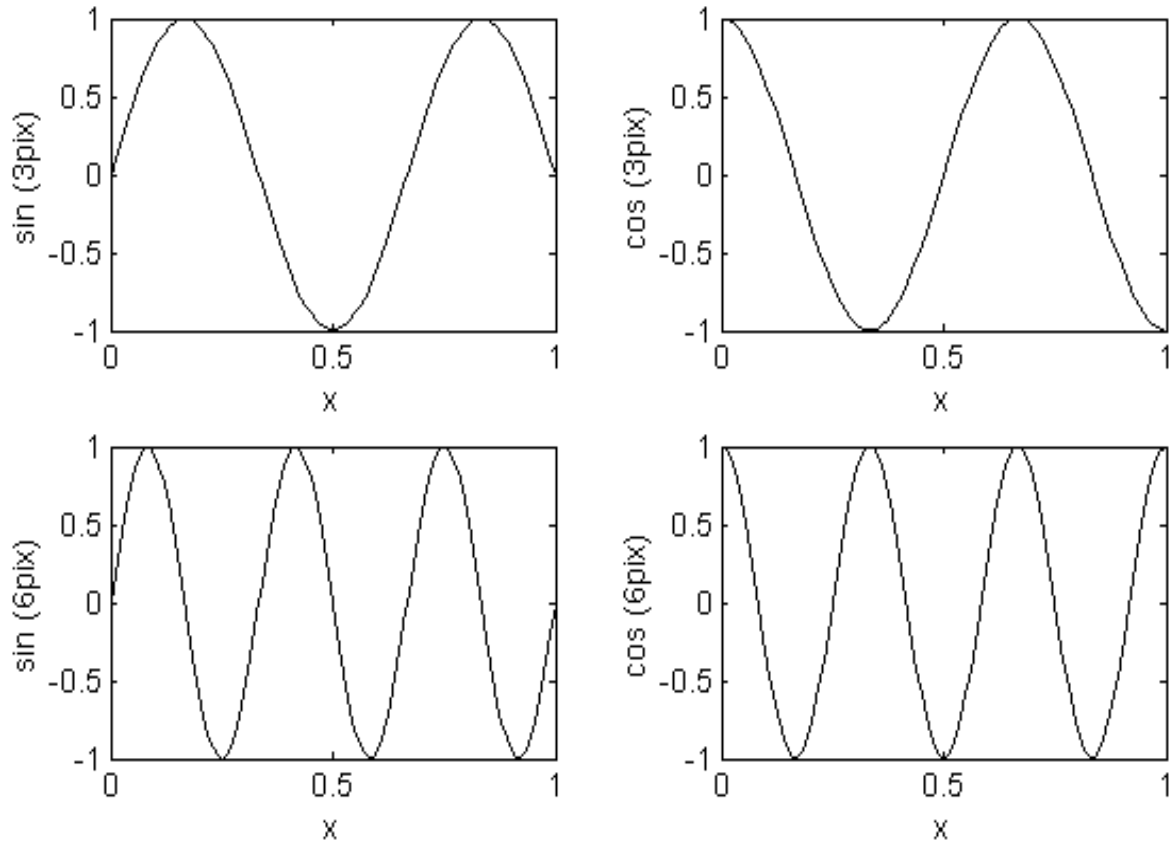


Figure 7. Using subplot

Subplot(221) (or subplot(2,2,1)) specifies that the window should be split into a 2 by 2 array and we select the first sub-window.

Exercise 1:

The laminar velocity profile in a pipe, can be given in equation $V_x=V_{max}(1-(2*r/di)^2)$

Plot the velocity profile in a pipe? If you know that:s

r= distance from pipe center

di: Inner diameter of pipe (di=0.2 m)

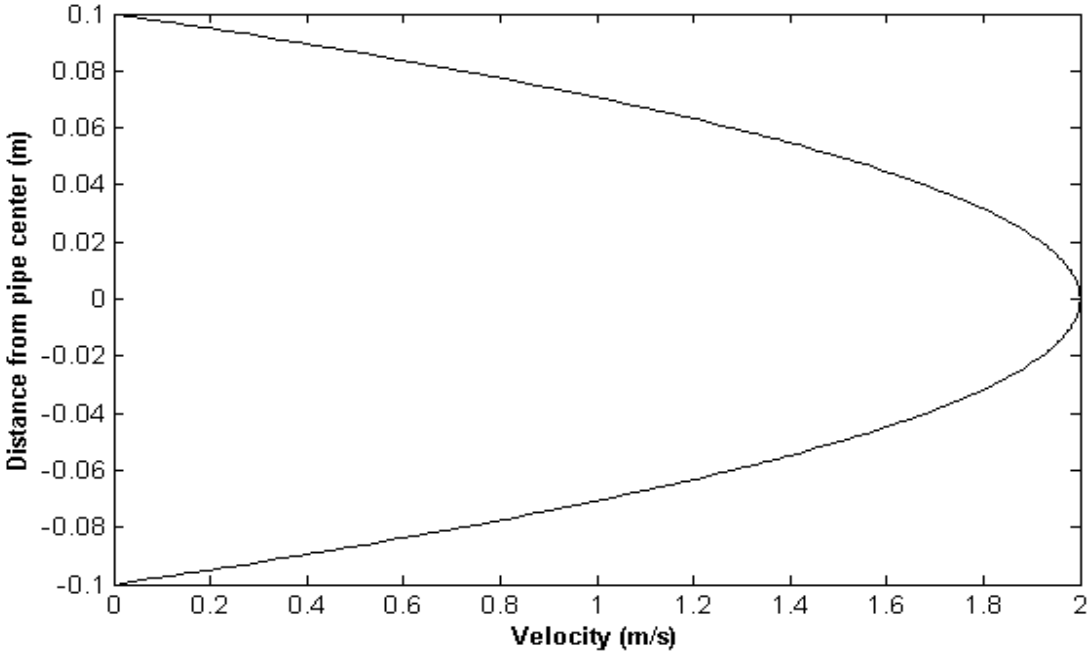
V_{max}: maximum velocity in pipe (V_{max}=2 m)

Solution

Write the following code

```
di=.2; Vmax=2
r=-.1:.001:.1
Vx=Vmax*(1-((2*r)/di).^2)
plot(Vx,r)
xlabel('Velocity (m/s)')
ylabel('Distance from pipe center (m)')
```

The result will be the figure.



Exercise 2:

Plot k as a function of temperature from 373 to 600 k (use 50 point of Temperature).

If you know:

$K=A \exp(B-(C/T)-\ln(D*T)-E*T+F*T^2)$

where

A=3.33

B=13.148

C=5639.5

D=1.077

$E=5.44*10^{-4}$

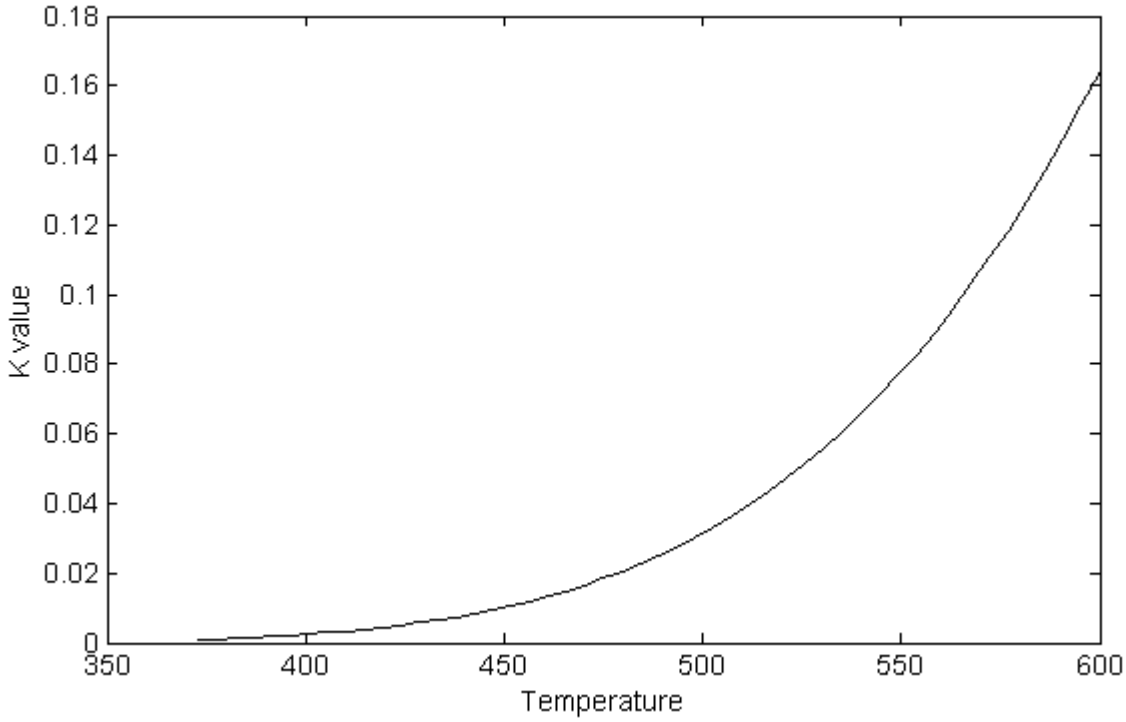
$F=1.125*10^{-7}$

Solution:

Write the following code

```
A=3.33;
B=13.148;
C=5639.5;
D=1.077;
E=5.44e-4;
F=1.125e-7;
Ts=(600-373)/49;%Ts means temperature increment
T=373:Ts:600
K=A*exp(B-(C./T)-log(D*T)-E*T+F*T.^2)
plot(T,K)
```

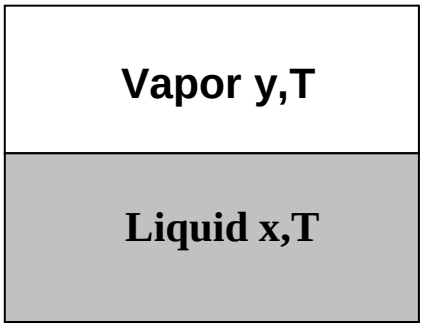
xlabel('Temperature')
ylabel('K value')
The result will be the figure.



Exercise 3:

Plot Txy diagram for ethanol/water system. Knowing that the vapor pressure for three components are calculated by:

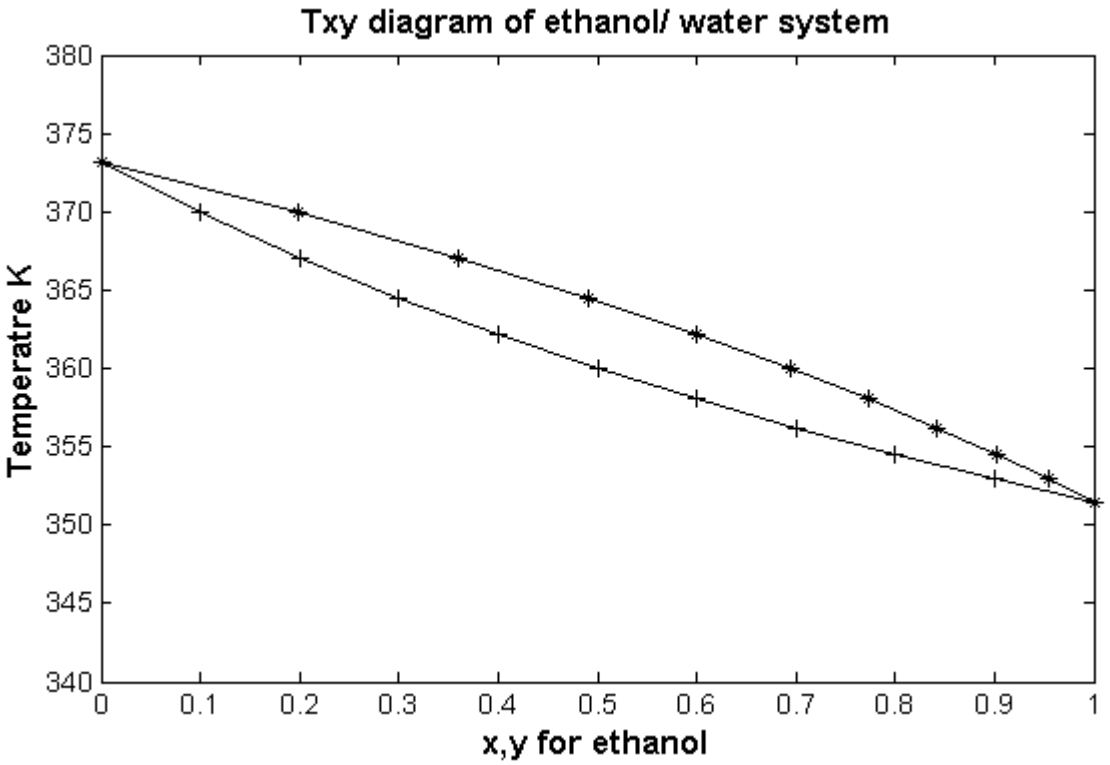
Ethanol P^o_e=exp(18.5242-3578.91/(T-50.5))
Water P^o_w=exp(18.3036-3816.44/(T-46.13))
Where
$K_i = P^o_i / P_t$
P_t=760
$y_i = K_i \times x_i$
At Bubble point $\sum yi = \sum Ki \times xi = 1$



Solution:

```
Xe=0:.1:1;
Xw=1-Xe;
for m=1:11
for T=273.15:.01:450;
Pe=exp(18.5242-3578.91/(T-50.5));  % Vapor pressure of Ethanol
Pw=exp(18.3036-3816.44/(T-46.13)); % Vapor pressure of Water
Ke=Pe/760; % Ethanol
```

```
Kw=Pw/760; % Water
sum=Ke*Xe(m)+Kw*Xw(m);
if sum>1
break
end
end
Temp(m)=T;
ye(m)=Ke*Xe(m);
end
plot(Xe,Temp,'k-+',ye,Temp,'k-*')
axis ([0 1 340 380])
xlabel('x,y for ethanol')
ylabel('Temperatre K')
title('Txy diagram of ethanol/ water system')
```



Practice Problems

1) The heat capacity for a mixture of ethanol and water is given as follows:

$Cp_m = x\ Cp_E(T) + (1-x)\ Cp_w(T)$

where x is the mole fraction of ethanol. The pure component heat capacities can be computed uing the following correlation:

$Cp(T) = a + bT + cT^2 + dT^3$

where T in degree Kelvin and Cp in J/(mole K)

Draw a plot for Cp_E , Cp_w and Cp_m covering mole fraction range of 0-1 for T=100 degree Kelvin. Knowing that:

Ethanol: a = -3.25137e+2 b = 4.13787e+0 c = -1.40307e-2 d = 1.70354e-5

Water: a = 1.82964e+1 b = 4.72118e-1 c = -1.33878e-3 d = 1.31424e-6

2) Plot the log of the values from 1 to 100.

3) Plot the parametric curve x = t cos(t), y = t sin(t) for 0 < t < 10.

4) Plot the cubic curve $y = x^3$. Label the axis and put a title "a cubic curve" on the top of the graph.

5) Plot y=(x3+x+1)/x, for –4< x <4 and –10< y < 10.

6) Plot y=ln(x+1) and $y=1-x^2$ on the same window for -2 < x <6 and –4 <y <4.

7) Plot $\cos(x^2+1)$ on [-2pi,2pi].

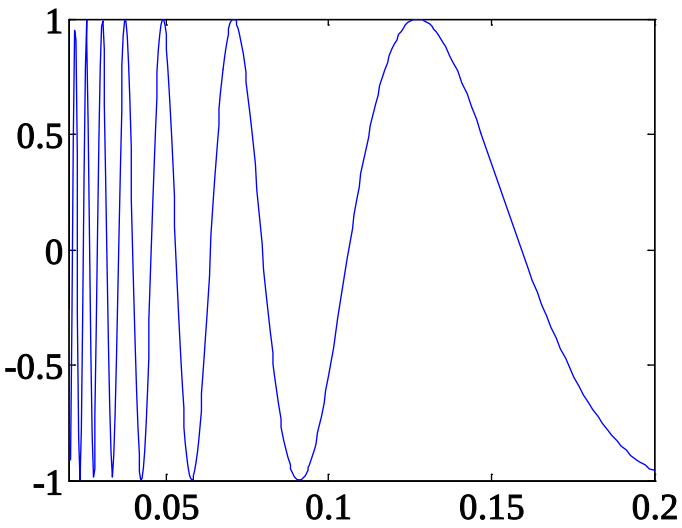
8. Specialized 2-D plotting functions

MATLAB includes a variety of specialized plotting in addition to the ones described above. The following below briefly describes some of the other plotting functions available in MATLAB.

fplot: evaluates a function and plots the results

Example:

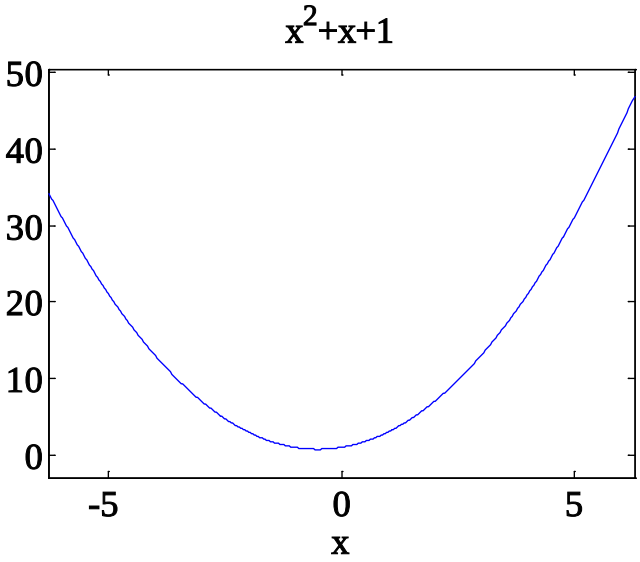
```
fplot('sin(1/x)', [0.02 0.2]);
```



ezplot: simplest way to graph a function (easy plot).

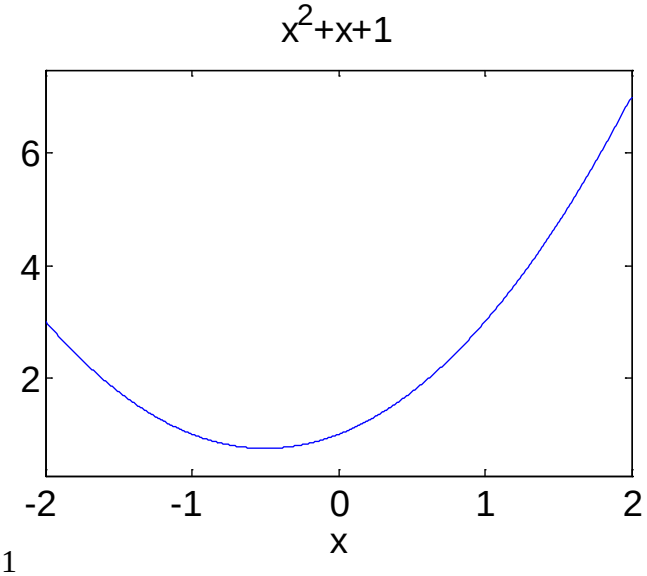
Example:, to graph the function $y=x^2+x+1$, you write:

```
ezplot('x^2+x+1')
```



If you want to sketch the same function in between –2 and 2 you simply write:

```
ezplot('x^2+x+1', [-2, 2])
```

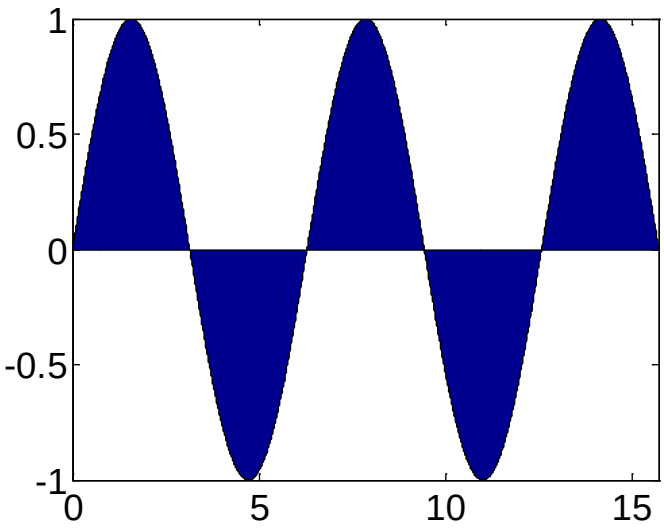


area Filled area plot.

area(X,Y) produces a stacked area plot suitable for showing the contributions of various components to a whole. For vector X and Y, area(X,Y) is the same as plot(X,Y) except that the area between 0 and Y is filled.

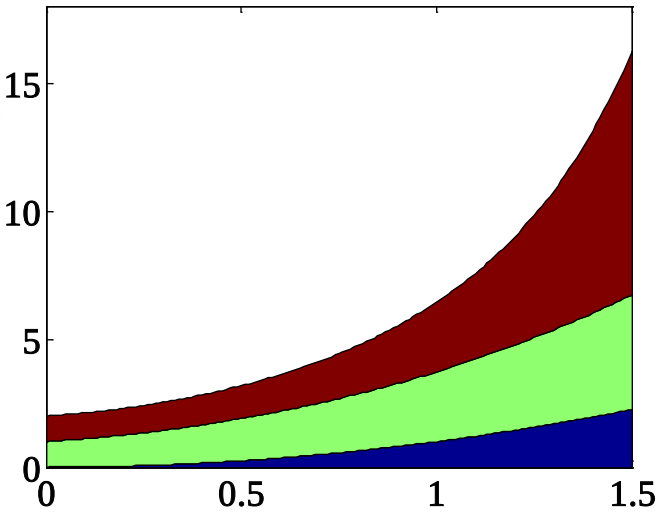
Example:

```
t = 0:.01:5*pi;
 area(t,sin(t))
```



Example:

```
x = 0:0.01:1.5;
area(x,[(x.^2)',(exp(x))',(exp(x.^2))'])
```

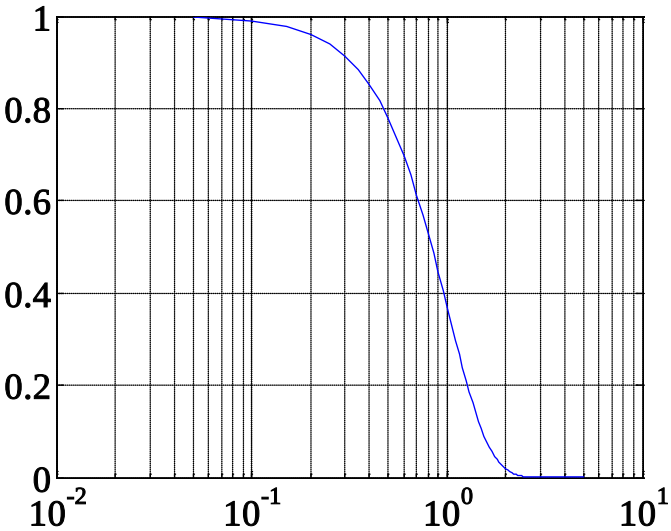


semilogx: log-scaled x axis

semilogx is the same as **plot**, except a logarithmic (base 10) scale is used for the X-axis.

Example:

```
x=0:0.05:5;
y=exp(-x.^2);
semilogx(x,y);
grid
```

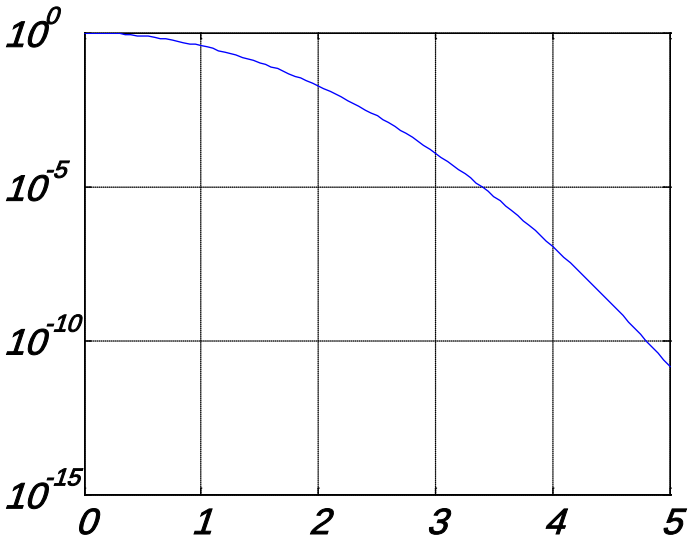


semilogy: log-scaled y axis

semilogy is the same as **plot**, except a logarithmic (base 10) scale is used for the Y-axis.

Example:

```
x=0:0.05:5;
y=exp(-x.^2);
semilogy(x,y);
grid
```

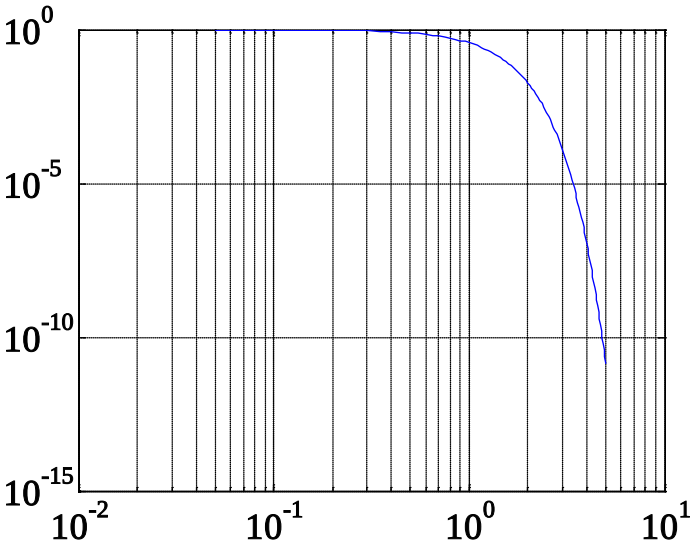


Loglog: Log-log scale plot.

Loglog is the same as **plot**, except logarithmic scales are used for both the X- and Y- axes.

Example:

```
x=0:0.05:5;
y=exp(-x.^2);
loglog(x,y);
grid
```

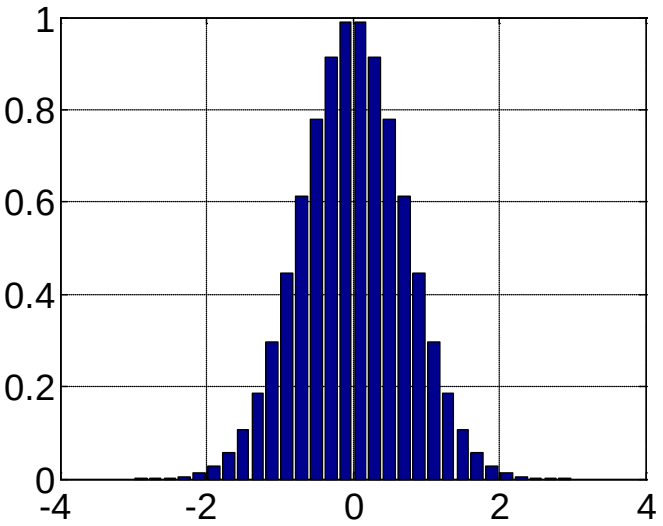


bar: creates a bar graph.

bar(X,Y) draws the columns of the M-by-N matrix Y as M groups of N vertical bars. The vector X must be monotonically increasing or decreasing

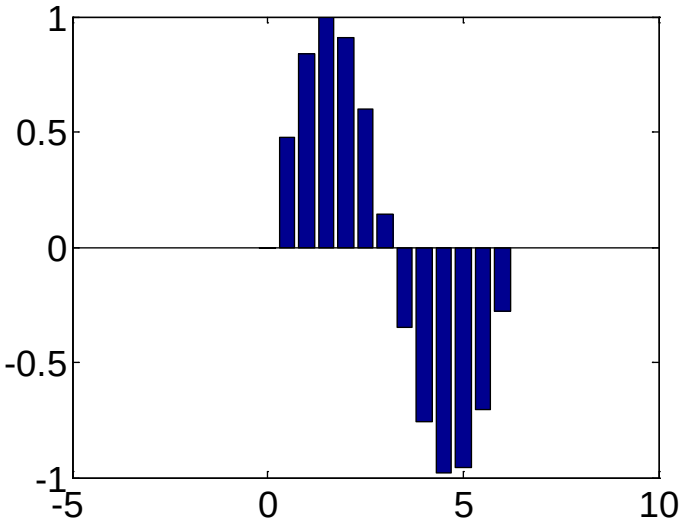
Example:

```
x = -2.9:0.2:2.9;
bar(x,exp(-x.*x));
grid
```



Example:

```
x = 0:0.5:2*pi;
bar(x, sin(x));
```

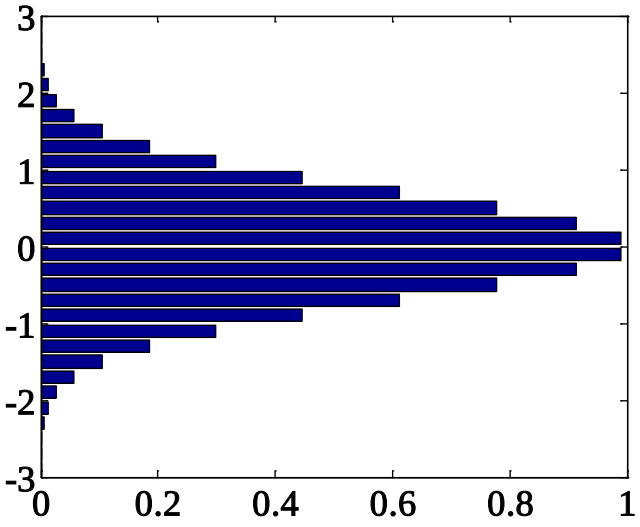


barh: horizontal bar graph

barh(X,Y) draws the columns of the M-by-N matrix Y as M groups of N horizontal bars.

Example:

```
x = -2.9:.2:2.9;
y = exp(-x.*x);
barh(x,y);
```

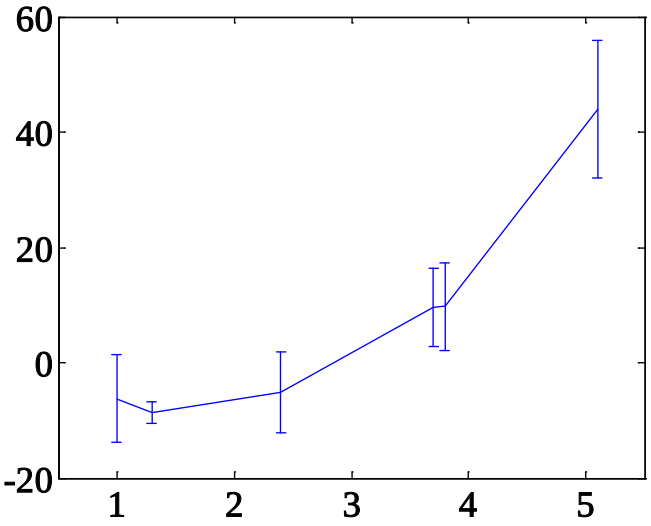


errorbar: creates a plot with error bars

errorbar (X,Y,L) plots the graph of vector X vs. vector Y with error bars specified by the vectors L and U.

Example:

```
x = [ 1.0 1.3 2.4 3.7 3.8 5.1 ];
y = [ -6.3 -8.7 -5.2 9.5 9.8 43.9 ];
coeff = polyfit(x,y,1)
yp=polyval(coeff,x)
e=abs(yp-y )
errorbar(x,y,e); grid
e =
```



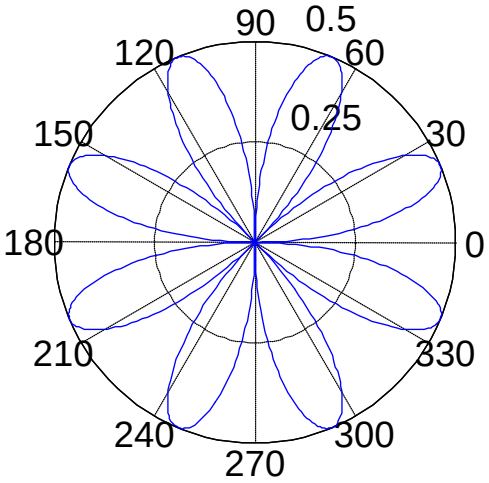

```
   7.6079    1.8509    6.9582    6.8052    7.6242   11.9288
```

polar: creates a plot in polar coordinates of angles versus radin

polar(theta,rho) makes a plot using polar coordinates of the angle **theta**, in radians, versus the radius **rho**.

Example:

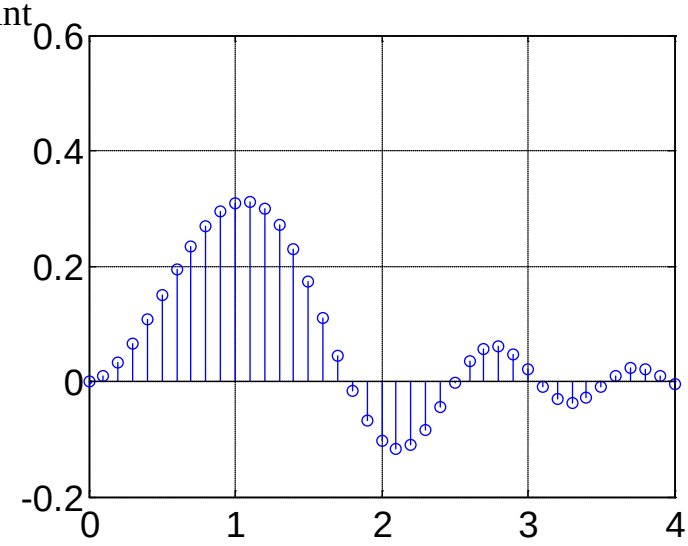
```
t=0:.01:2*pi;
polar(t,sin(2*t).*cos(2*t));
```



stem: generates stems at each data point

Example:

```
x = 0:0.1:4;
y = sin(x.^2).*exp(-x);
stem(x,y);
grid
```

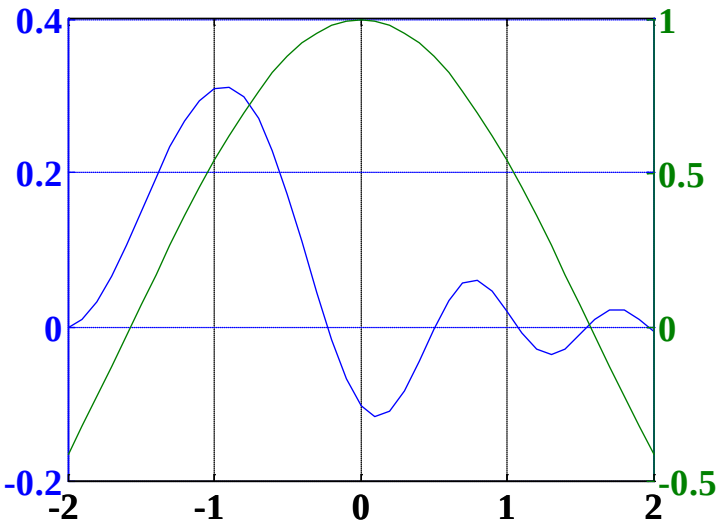


plotyy: graphs with y tick labels on the left and right

plotyy(X1,Y1,X2,Y2) plots Y1 versus X1 with y-axis labeling on the left and plots Y2 versus X2 with y-axis labeling on the right.

Example:

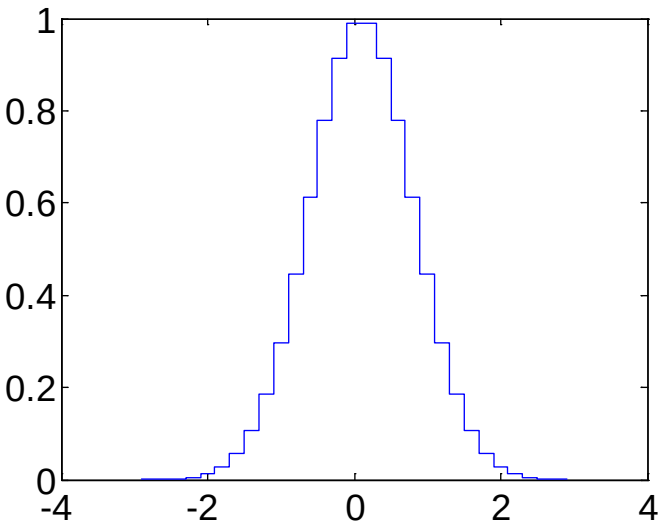
```
x=-2:0.1:2;
y1=sin(x);
y2=cos(x);
plotyy(x,y,x,y2);
grid
```



stairs: creates a graph similar to a bar graph, but without internal lines

Example:

```
x = -2.9:.2:2.9;
y = exp(-x.*x);
stairs(x,y);
title('Stair Chart');
```

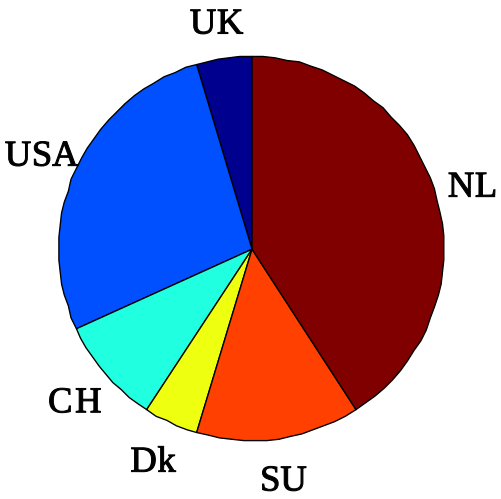


pie: Pie chart.

pie(X) draws a pie plot of the data in the vector **X**. The values in X are normalized via X/sum(X) to determine the area of each slice of pie.

Example:

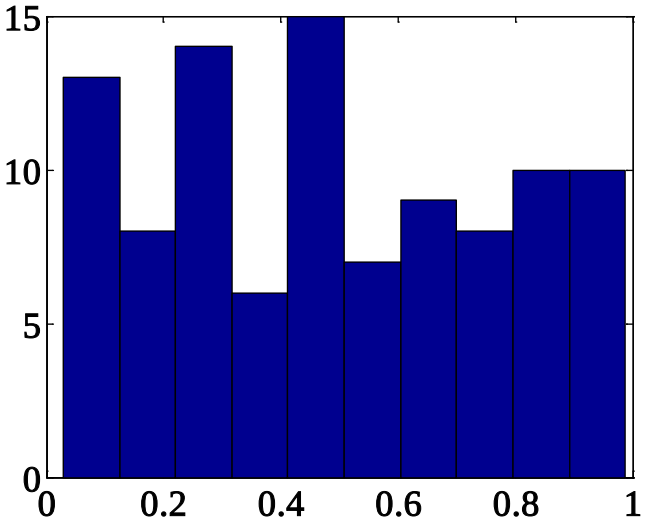
```
x = [1 6  2 1 3 9];
label = {'UK','USA','CH','Dk','SU','NL'};
pie(x, label)
```



hist: creates a histogram

Example:

```
x=rand(1,100);
hist(x);
grid
```



3-D graphics

Matlab provides extensive facilities for visualization of three-dimensional data. The most common are plots of curves in a three-dimensional space, mesh plots and surface plots. The command **plot3(x,y,z,'style option')** produces a curve in the three-dimensional space. The **title**, **xlabel**, **ylabel**, etc., may be used for three-dimensional plots. The **mesh** and **surf** commands have several optional arguments and are used for plotting meshes and surfaces.

Following is a list of elementary 3-D plots and some specialized 3-D graphs.

plot3 Plot lines and points in 3-D space
mesh 3-D mesh surface
surf 3-D colored surface
fill3 Filled 3-D polygons
comet3 3-D comet-like trajectories
ezgraph3 General purpose surface plotter
ezmesh Easy to use 3-D mesh plotter
ezmeshc Easy to use combination mesh/contour plotter
ezplot3 Easy to use 3-D parametric curve plotter
ezsurf Easy to use 3-D colored surface plotter
ezsurfc Easy to use combination surf/contour plotter
meshc Combination mesh/contour plot
meshz 3-D mesh with curtain
scatter3 3-D scatter plot
stem3 3-D stem plot
surfc Combination surf/contour plot
trisurf Triangular surface plot
trimesh Triangular mesh plot
cylinder Generate cylinder
sphere Generate sphere

plot3: Plot lines and points in 3-D space

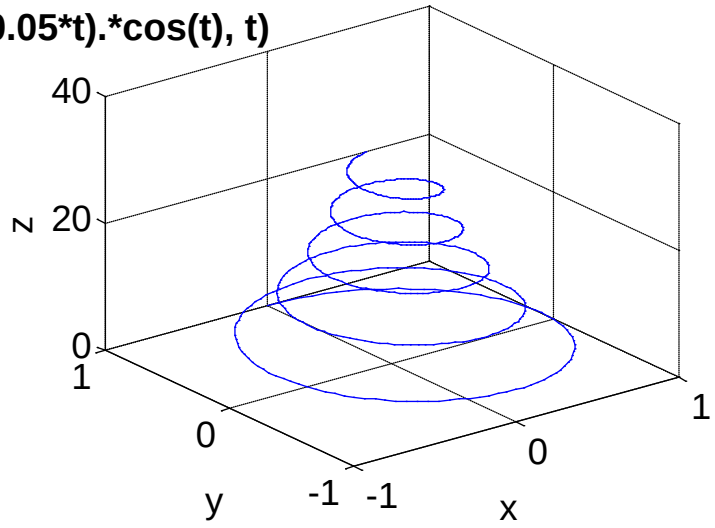
Three-dimensional plots completely analogous to plot in two dimensions, the command **plot3** produces curves in three-dimensional space. If *x*, *y*, and *z* are three vectors of the same size, then the command **plot3(x,y,z)** will produce a perspective plot of the piecewise linear curve in 3-space passing through the points whose coordinates are the respective elements of *x*, *y*, and *z*. These vectors are usually defined parametrically.

For example, the following equations specify a curve in 3 dimensions:

$x = e^{-0.05t} \sin(t)$
$y = e^{-0.05t} \cos(t)$
$z = t$

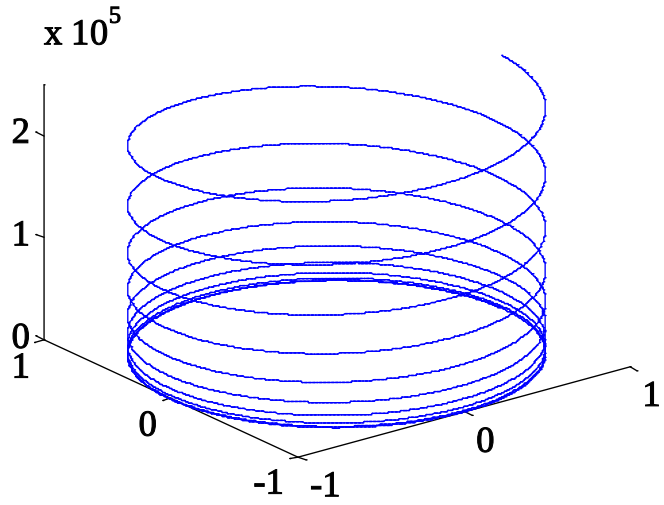
If t ranges from 0 to 10 to plot type:

```
t = [0:pi/50:10*pi];
plot3(exp(-0.05*t).*sin(t), exp(-0.05*t).*cos(t), t)
xlabel('x')
ylabel('y')
zlabel('z')
grid on
```



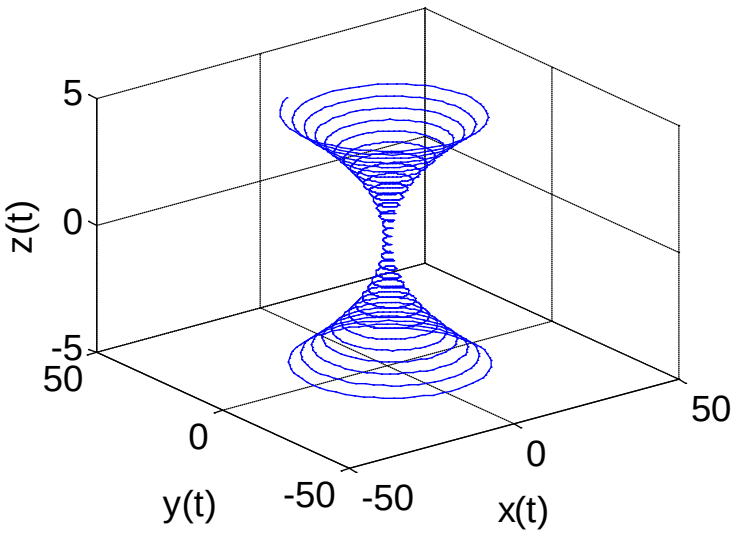
Example:

```
t=.01:.01:20*pi;
x=cos(t);
y=sin(t);
z=t.^3;
plot3(x,y,z)
```



Example:

```
t=-5:.005:5;
x=(1+t.^2).*sin(20*t);
y=(1+t.^2).*cos(20*t);
z=t;
plot3(x,y,z)
grid on
xlabel('x(t)')
ylabel('y(t)')
zlabel('z(t)')
```



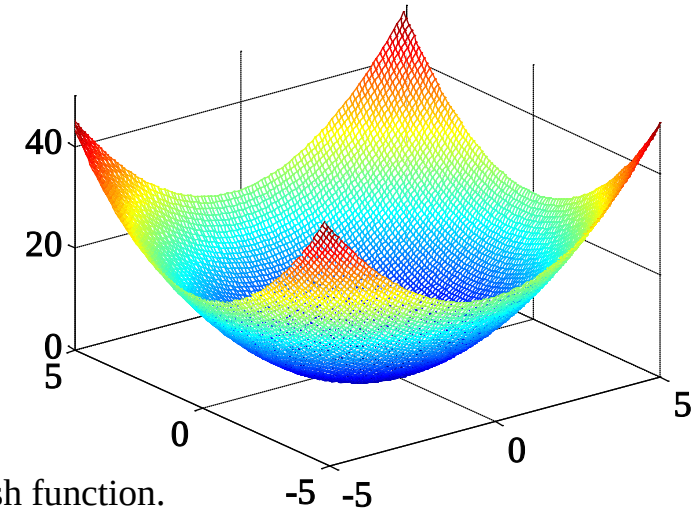
Creating 3-D Graphs

MATLAB defines a surface by the *z*-coordinates of points above a rectangular grid in the *x*-*y* plane. The plot is formed by joining adjacent points with straight lines. Surface plots are useful for visualizing matrices that are too large to display in numerical form and for graphing functions of two variables.

mesh: A mesh surface is defined by the **z** coordinates of points above a rectangular grid in the x-y plane. The plot is formed by joining adjacent points with straight lines.

Example: Generate the matrix table of the function f=x.^2+y.^2; where 0 <= x <= 5, 0 <= y <= 5

```
xv=[-5:.1:5];
yv=[-5:.1:5];
nx=length(xv);
ny=length(yv);
for i=1:nx
for j=1:ny
zmat(i,j)=(xv(j).^2+yv(i).^2);
end
end
```



To display the function, we can use mesh function.

```
mesh(xv,yv,zmat);
```

meshgrid

To generate the matrix table, we needed to perform a double nested loop which is time consuming. MATLAB has it's own method of generating matrix tables using the meshgrid function. To see what meshgrid does

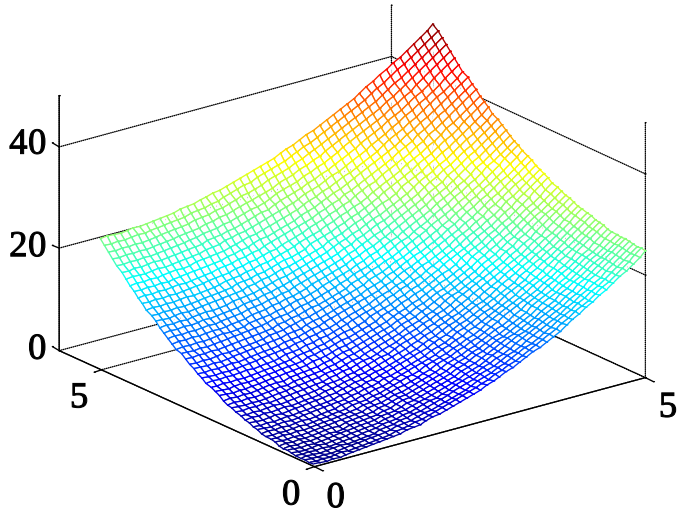
```
x=[6:1:10];
y=[1:1:5];
[X,Y]=meshgrid(x,y);
X =
    6    7    8    9   10
    6    7    8    9   10
    6    7    8    9   10
    6    7    8    9   10
    6    7    8    9   10
Y =
    1    1    1    1    1
    2    2    2    2    2
    3    3    3    3    3
    4    4    4    4    4
    5    5    5    5    5
```

X is a matrix in which the x domain [6 7 8 9 10] is copied in every row and Y is a matrix in which the y domain [1 2 3 4 5] is copied in every column.

Now since Z(i,j)=Z(x(j),y(i)) by definition and since X(i,j)=x(j) for any i and Y(i,j)=y(i) for any j, Z(i,j)=Z(X(i,j),Y(i,j)) which is the definition of a matrix array operation.

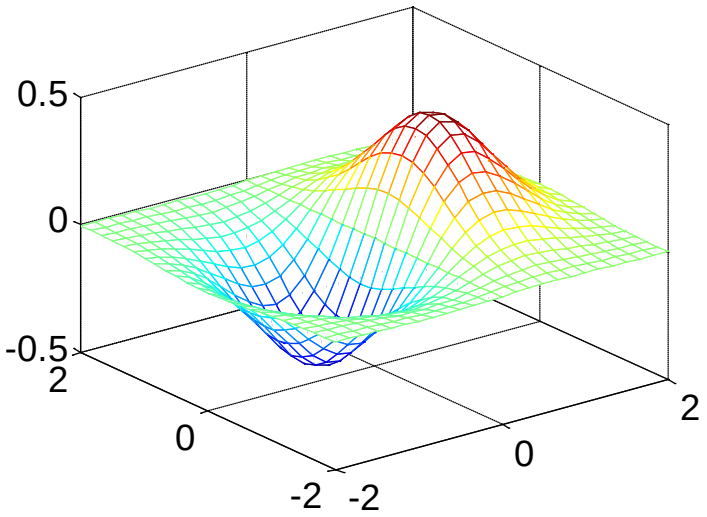
For example, **meshgrid** can be used to generate X- and Y-grid matrices for use when computing the displayed Z matrix.

```
x=[0:.1:5];
y=[0:.1:5];
[X,Y]=meshgrid(x,y);
Z=(X.^2)+(Y.^2);
mesh(x,y,Z)
```



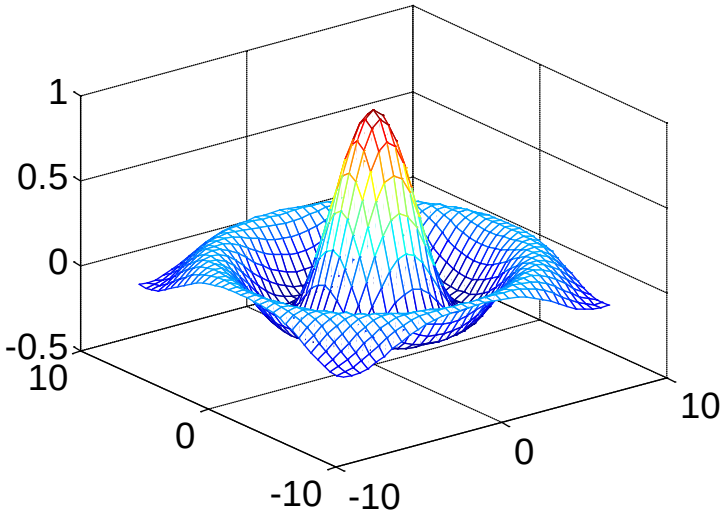
Example:

```
x=linspace(-2, 2, 25);
y=linspace(-2, 2, 25);
[xx,yy]=meshgrid(x, y);
zz=xx.*exp(-xx.^2-yy.^2);
mesh(xx, yy, zz);
```



Example: Graphing the sinc function.

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
mesh(X,Y,Z)
```

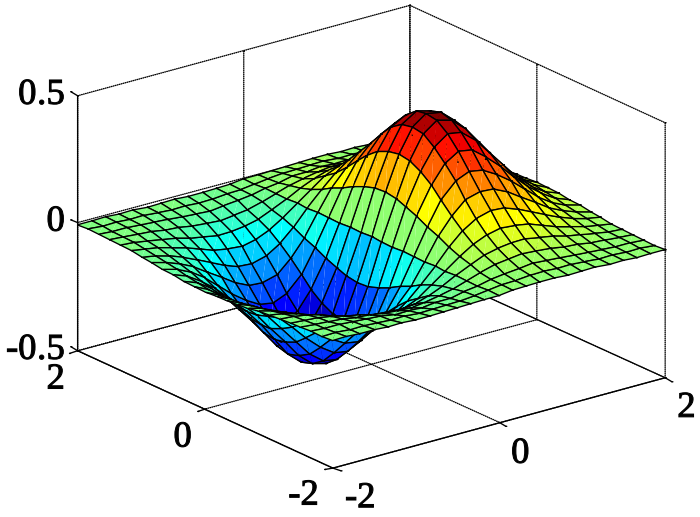


surf: produces a solid filled surface plot.

A surface plot is similar to a mesh plot except the rectangular faces of the surface are colored.

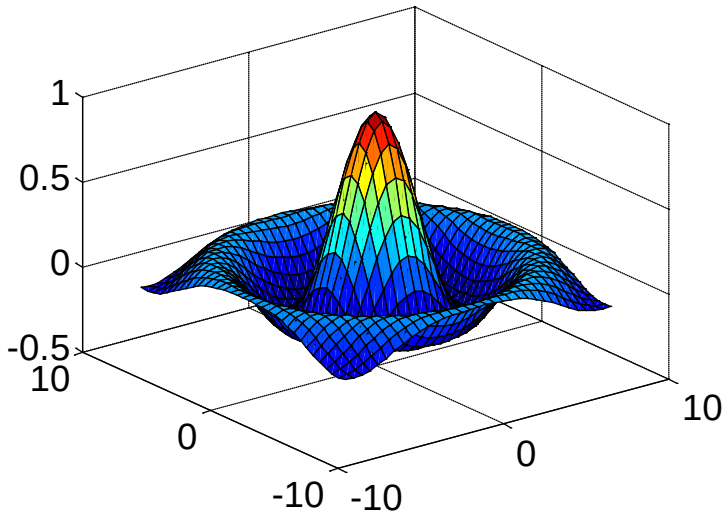
Example:

```
x=linspace(-2, 2, 25);
y=linspace(-2, 2, 25);
[xx,yy]=meshgrid(x, y);
zz=xx.*exp(-xx.^2-yy.^2);
surf(xx, yy, zz);
```



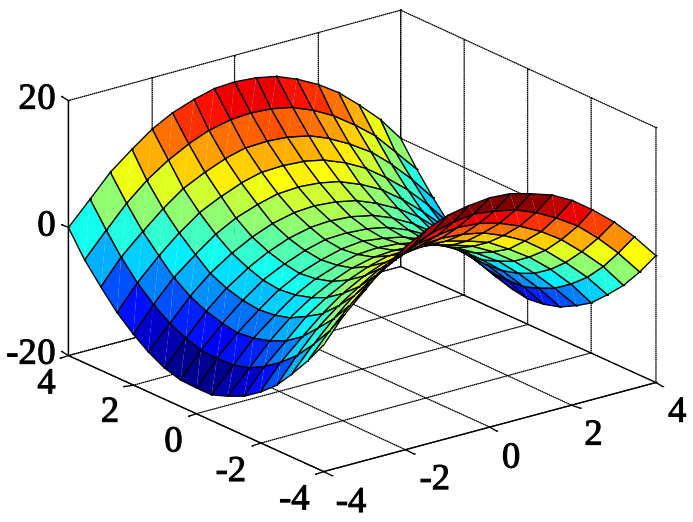
Example:

```
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2) + eps;
Z = sin(R)./R;
surf(X,Y,Z)
```



Example: $z = y^2 - x^2, -4 \le x \le 4, -4 \le y \le 4$

```
[xx yy]=meshgrid(-4:0.5:4,-4:0.5:4);
z=yy.^2-xx.^2;
surf(xx,yy,z)
```

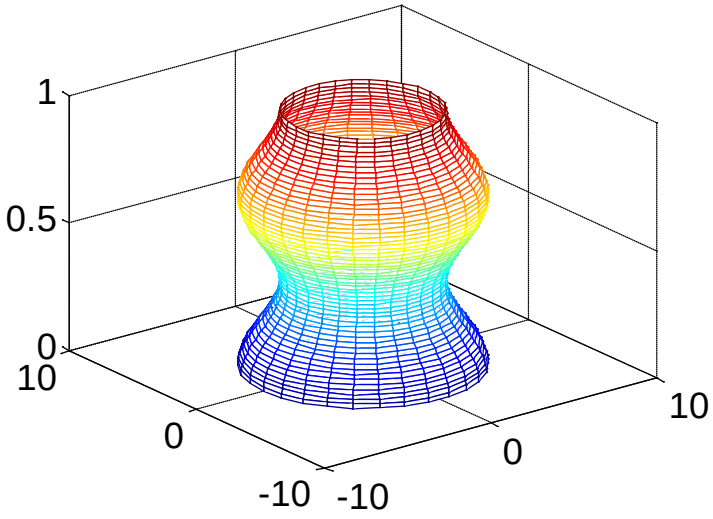


cylinder Generate cylinder.

[X,Y,Z] = cylinder (R,N) forms the unit cylinder based on the generator curve in the vector R. Vector R contains the radius at equally spaced points along the unit height of the cylinder. The cylinder has N points around the circumference.

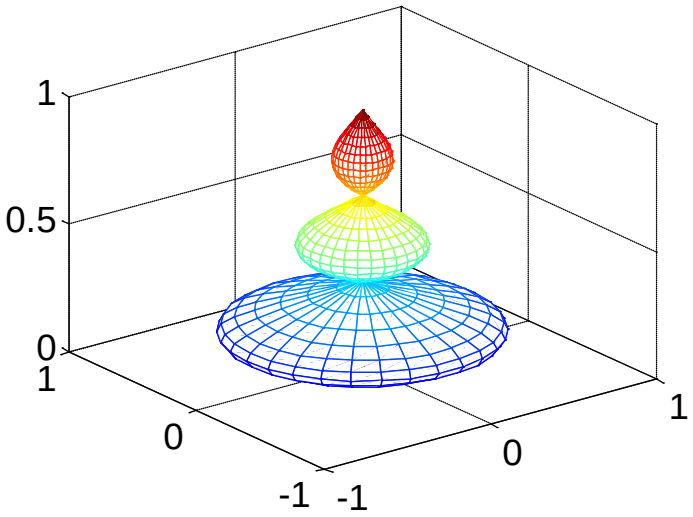
Example:

```
x=0:pi/20:pi*3;
r=5+cos(x);
[a,b,c]=cylinder(r,30);
mesh(a,b,c)
```



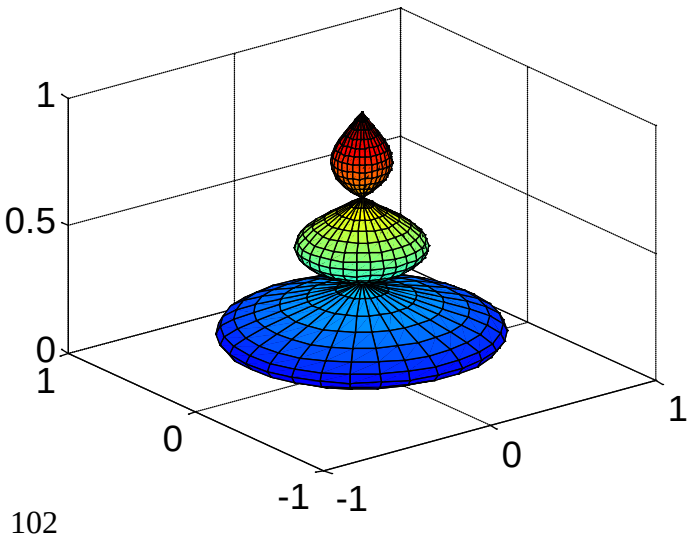
Example:

```
t=0:pi/12:3*pi;
r=abs(exp(-0.25*t).*sin(t));
[X,Y,Z]=cylinder(r,30);
mesh(X,Y,Z)
```



Example:

```
t=0:pi/12:3*pi;
r=abs(exp(-0.25*t).*sin(t));
[X,Y,Z]=cylinder(r,30);
surf(X,Y,Z)
```

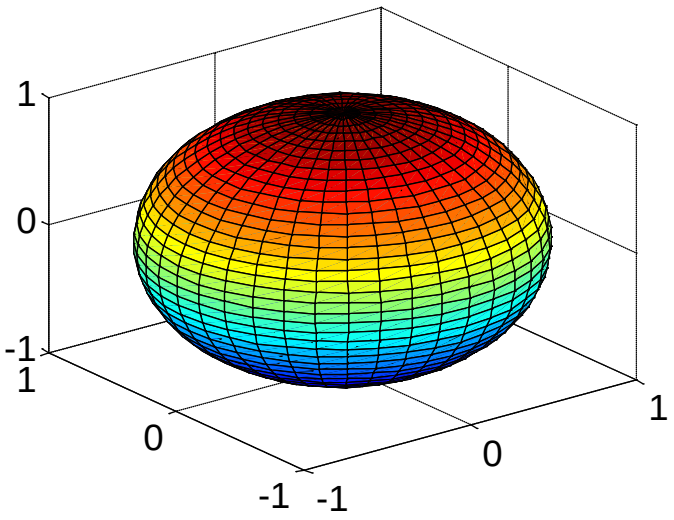


sphere Generate sphere.

[**X,Y,Z] = sphere(N)** generates three (N+1) by (N+1) matrices so that **surf(X,Y,Z)** produces a unit sphere.

Example:

```
[a,b,c]=sphere(40);
surf(a,b,c);
```



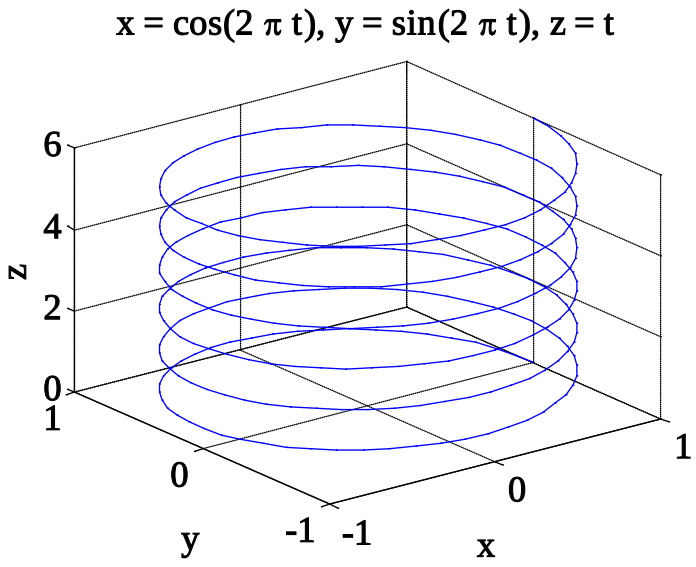
ezplot3 Easy to use 3-d parametric curve plotter.

ezplot3 (FunX,FunY,FunZ) plots the spatial curve FunX(T), FunY(T), and FunZ(T) over the default domain 0 < T < 2*Pi.

ezplot3 (FunX,FunY,FunZ,[Tmin,Tmax]) plots the curve FunX(T), FunY(T), and FunZ(T) over Tmin < T < Tmax.

Example:

```
ezplot3('cos(2*pi*t)', 'sin(2*pi*t)', 't', [0, 6])
```



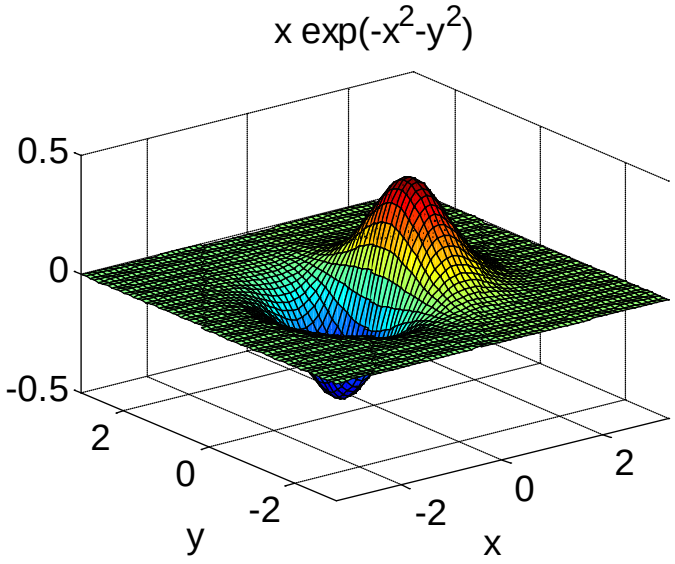
ezsurf Easy to use 3-D colored surface plotter.

ezsurf(Fun) plots a graph of the function Fun(X,Y) using surf. Fun is plotted over the default domain -2*pi < X < 2* pi, -2* pi < Y < 2* pi.

ezsurf(Fun,Domain) plots Fun over the specified Domain instead of the default domain. Domain can be the vector [Xmin,Xmax,Ymin,Ymax] or the vector [A,B] (to plot over A < X < B, A < Y < B).

Example: $f(x,y) = xe^{-x^2-y^2}$

ezsurf('x*exp(-x^2-y^2)')



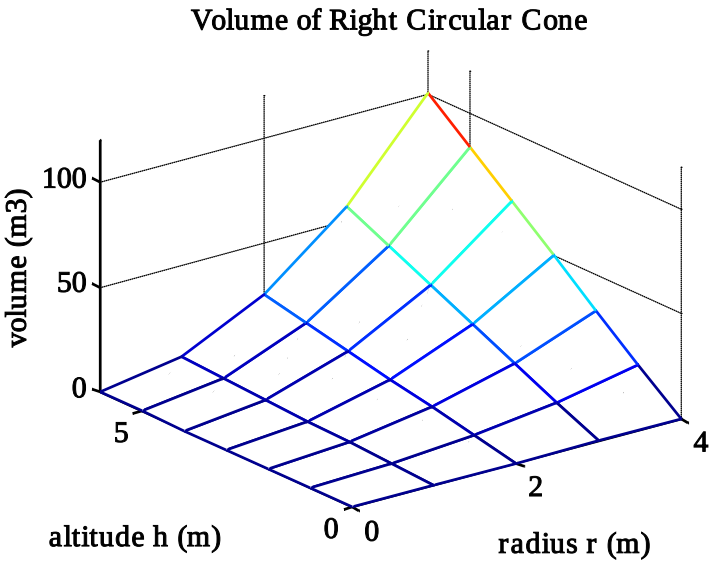
Exercise 1:

The volume of a right circular cone of radius and height is given by $V = \frac{1}{3}\pi r^2 h$

Plot the volume of the cone as and vary on the radius and height.

Solution:

```
[R,H]=meshgrid(0: 4, 0: 6);
V=(pi .* R .^ 2 .* H) ./ 3;
mesh(R, H, V)
xlabel('radius r (m)');
ylabel('altitude h (m)');
zlabel('volume (m^3)');
title('Volume of Right Circular Cone');
```



The three−dimensional plot of Figure above, shows how the volume of the cone increases as the radius and height are increased.

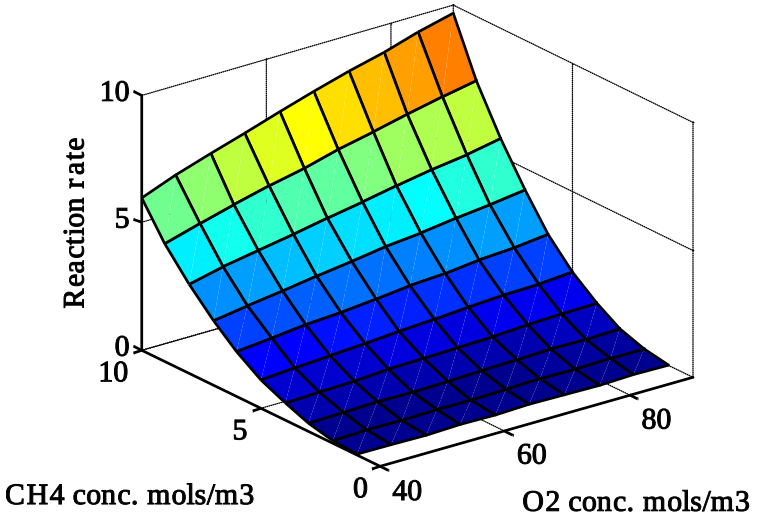
Exercise 2:

We would like to plot a surface for the rate of reaction of methane oxidation. The reaction rate is represented power law $R = k_r\, c_{CH4}^{0.6} c_{O2}^{2.4}$, where k is the reaction rate constant $k_r = 2.6\times10^{-3}$

The concentration of O_2 in the range between 1 to 10 mols/m^3, where the Methane concentration in the range 40 to 90 mols/m. Taken 10 point of each component plot a surface for the reaction rate.

Solution

```
kr=2.6e-3;
X1=linspace(40,90,10);
X2=linspace(1,10,10);
[Xch4,Xo2]=meshgrid(X1,X2);
R=kr*(Xch4.^0.6).*(Xo2.^2.4);
Surf(Xch4,Xo2,R);
xlabel('O2 conc. mols/m3')
ylabel('CH4 conc. mols/m3')
zlabel('Reaction rate')
```



Practice Problems

1) Create surface plots for the functions

$z = (x - 2)^2 + 2xy + y^2$

$z = x^2 - 2xy + 4y^2$

$z = -x^2 + 2xy + 3y^2$

$z = (x - y^2)\,(x - 3y^2)$

Q 1: Chemical engineer, as well as most other engineers, use thermodynamics extensively in their work. The following polynomial can be used to relate specific heat of dry air, C_p KJ/(Kg K), to temperature (K):

$C_p = 0.99403 + 1.617\times10^{-4}T + 9.7215\times10^{-8}T^2 - 9.5838\times10^{-11}T^3 + 1.9520\times10^{-14}T^4$

Determine the temperature that corresponds to a specific heat of 1.2 KJ/(Kg K).

Solution 1:

```
solve('1.2=0.99403+1.617e-4*T+9.7215e-8*T^2-9.5838e-11*T^3+1.9520e-14*T^4')
ans =
1175.1599
2507.4830+851.5920*i
-1280.3925
2507.4831-851.5920*i
```

By neglecting the negative and imaginary roots, the correct answer is the first root T=1175.1599

Solution 2:

```
T=0:1:10000;
Cp=0.99403+1.617e-4*T+9.7215e-8*T.^2-9.5838e-11*T.^3+1.9520e-14*T.^4;
Ti=interp1(Cp,T,1.2)
```

Gives the result

```
Ti =
  1.1752e+003
```

Q 2: Evaluate the following double integral

$$\int_0^{\pi}\int_0^{\sin x}(x^2+y^2)dy\cdot dx$$

Solution:

MATLAB can also do multiple integrals. The following command computes the double integral:

```
syms x y;
int(int(x^2 + y^2, y, 0, sin(x)), 0, pi)
ans =
-32/9+pi^2
```

To convert the way of the result displaying, type the code:

```
single(-32/9+pi^2)
ans =
   6.3140
```

Q 3: Evaluate the triple integral:

$$\int_{-4}^{4}\int_{0}^{6}\int_{0}^{6}(x^3-2yz)dx\cdot dy\cdot dz$$

Solution:

The following command computes the triple integral:

```
single(int(int(int('x^3 -2*y*z','z',-1, 3),'y',0,6),'x',-4,4))
ans =
     -1152
```

Or type the code which gives the same result:

```
syms x y z
single(int(int(int(x^3 -2*y*z,z,-1, 3),y,0,6),x,-4,4))
ans =
     -1152
```

Q 4: The average values of a specific heat can be determined by $Cp_{mh} = \frac{\int_{T_1}^{T_2} CpdT}{T_2-T_1}$

Use this relationship to verity the average value of specific heat of dry air in the range from 300 K to 450 K, C_p KJ/(Kg K), to temperature (K):

$C_p = 0.99403 + 1.617\times10^{-4}T + 9.7215\times10^{-8}T^2 - 9.5838\times10^{-11}T^3 + 1.9520\times10^{-14}T^4$

Solution:

```
syms T
Cp=0.99403+1.617e-4*T+9.7215e-8*T.^2-9.5838e-11*T.^3+1.9520e-
14*T.^4;
Cpmh=single((int(Cp,300,450))/(450-300))
```

The result will be:

```
Cpmh =
   1.0637
```

Q 5: Compute the average molecular weight of a mixture of equal fractions of CH_4, H_2, O_2, CO, CO_2, H_2O, CH_3OH, C_2H_6.

The average molecular weight: $Mw_{average} = \sum_{i=1}^{N_c} Mw_i \times x_i$

Solution 1

```
MW=[16 2 32 28 44 18 32 30];
X(1:8)=1/8;
Mavearge=sum(MW.*X)
```

The result will be:

```
Mavearge =
  25.2500
```

Solution 2

```
MW=[16 2 32 28 44 18 32 30];
mean(MW)
ans =
  25.2500
```

Q 6: Use the following set of pressure-volume data to find the best possible virial constants (B and C) for the equation of state shown below. R=82.05 ml atm/gmol K and T= 303 K.

$$\frac{PV}{RT} = 1 + \frac{B}{V} + \frac{C}{V^2}$$

P(atm)	0.983	1.108	1.363	1.631
V (ml)	25,000	22,200	18,000	15,000

Solution:

```
R=82.05;
T=303;
P=[0.983, 1.108, 1.363, 1.631]
V=[25000, 22200, 18000, 15000]
Y=P.*V/(R*T)
X=1./V
Poly=polyfit(X,Y,2)
B=Poly(2)
C=Poly(1)
```

The result will be:

```
B =
  600.4060
C =
-7.3553e+006
```

Q 7: Fully developed flow moving a 40 cm diameter pipe has the following velocity profile:

Radius r, cm	0.0	2.5	5.0	7.5	10.0	12.5	15.0	17.5	20.0
Velocity v, m/s	0.914	0.890	0.847	0.795	0.719	0.543	0.427	0.204	0

Find the volumetric flow rate Q using the relationship Q = $\int_0^R 2\pi r v\, dr$. Where r is the radial axis of the pipe, R is the radius of the pipe and v is the velocity.

Solve the problem using two steps.

(a) Fit a polynomial curve to the velocity data using **polyfit**.

(b) Integrate the equation using **int**.

Solution:

```
R=[0,2.5,5,7.5,10,12.5,15,17.5,20]/100;
V=[0.914, 0.890, 0.847, 0.795, 0.719, 0.543, 0.427, 0.204,0];
P=polyfit(R,V,3);
syms r
Equ=P(1)*r^3+ P(2)*r^2+ P(3)*r+P(4);
Q=single(int(2*pi*r*Equ,0,0.20))
```

The result will be:

```
Q =
   0.0582
```

Q 8: A constant temperature, pressure-volume thermodynamic process has the following data:

Pressure (kpa)	420	368	333	326	312	242	207
Volume (m^3)	0.5	2	3	6	8	10	11

We know that W = $\int pdV$. Where W is work, p is pressure, and V is volume. Calculate the work required to compression from 5 m^3 to 2 m^3.

Solution:

```
Vol=[420,368,333,326,312,242,207];
Pre=[0.5,2,3,6,8,10,11];
Poly=polyfit(Vol,Pre,3);
syms V
Equ=Poly(1)*V^3+ Poly(2)*V^2+ Poly(3)*V+Poly(4);
Q=single(int(Equ,V,5,2))
```

The result will be:

```
Q =
  1.9726e+002
```

Q 9: The Redlich-Kwong equation of state is given by P = $\frac{RT}{v-b} - \frac{a}{v(v+b)\sqrt{T}}$

Where R= the universal gas constant [=0.518 kj/(Kg K)], T= absolute temperature (K), p= absolute pressure (KPa). And v = volume of a Kg of gas (m^3/Kg). The parameters a and b are calculated by a =0.427 $\frac{R^2 T_c^{2.5}}{p_c}$ b= 0.0866 $R\frac{T_c}{p_c}$

Where p_c= critical pressure (KPa) and T_c = critical temperature (K). As a chemical engineer, you are asked to determine the amount of methane fuel (pc = 4600 KPa and T_c = 191 K) that can be held in a 3 m^3 tank at temperature of -40 ^oC with a pressure of 65,000 KPa.

Solution:

```
T=-40+273.15; Tc=191; P=65000; Pc=4600; R=0.518;
a=0.427*(R^2*Tc^2.5)/Pc; b=0.0866*(R*Tc)/Pc;
x=[0.1:0.1:2]*(R*T)/P;
```

```
y=(R*T)./(x-b)-a./(x.*(x+b)*sqrt(T));
v=interp1(y,x,P)
amount_of_methane=3/v
```

The result will be:

```
v =
   0.0028
amount_of_methane =
   1.0665e+003
```

Q 10: The volume V of liquid in a spherical tank of radius r is related to the depth h of the liquid by $V = \frac{\pi h^2(3r-h)}{3}$ determine h given r=1 m and V=0.5 m^3.

Solution:

```
r=1;
x=0.01:0.01:10;
y=pi*x.^2.*(3*r-x)/3;
h=interp1(y,x,0.5)
h =
   0.4311
```

To check the result, type the code in command window

```
V=pi*h^2*(3*r-h)/3
V =
   0.5000
```

Q 11: A liquid mixture of benzene and toluene is in equilibrium with its vapor in closed system. At what temperature would the vapor phase composition of both benzene and toluene 50% at equilibrium, given that the pressure in closed container is 1.4 atm?

$$P^o_{benzene} = \exp(10.4 - \frac{3740}{T+5.8})$$

$$P^o_{Toluene} = \exp(9.0 - \frac{3500}{T+10})$$

Where: P^o in atm and T in K.

Solution:

```
X=1:1:1000;
Pe=exp(10.4-3740./(X+5.8));
Pw=exp(10.4-3740./(X+5.8));
Y=0.5*Pe+0.5*Pw;
T=interp1(Y,X,1.4)
```

The result will be:

```
T =
  365.8376
```

Q 12: Reported values for the virial coefficient of isopropanol vapor at 200 ^{o}C are:

B=-388 cm^3/mol and C=-26000 cm^6/mol^2 calculate V and Z for isopropoanol at 200 ^{o}C and 10 bar using virial equation

$$Z = \frac{PV}{RT} = 1 + \frac{B}{V} + \frac{C}{V^2}$$

Solution:

Re-write the equation as below:

$$V = \frac{RT}{P}\left(1 + \frac{B}{V} + \frac{C}{V^2}\right)$$

The absolute temperature is T=473.15 K, and appropriate value of gas constant is R=83.14 cm^3 bar/mol K. type the code in command window.

```
T=473.15; P=10; R=83.14;
B=-388; C=-26000; V=(R*T)/P;
for i=1:10
V=((R*T)/P)*(1+(B/V)+(C/V^2));
end
V
Z=(P*V)/(R*T)
```

The result will be

```
V =
  3.4877e+003
Z =
    0.8866
```

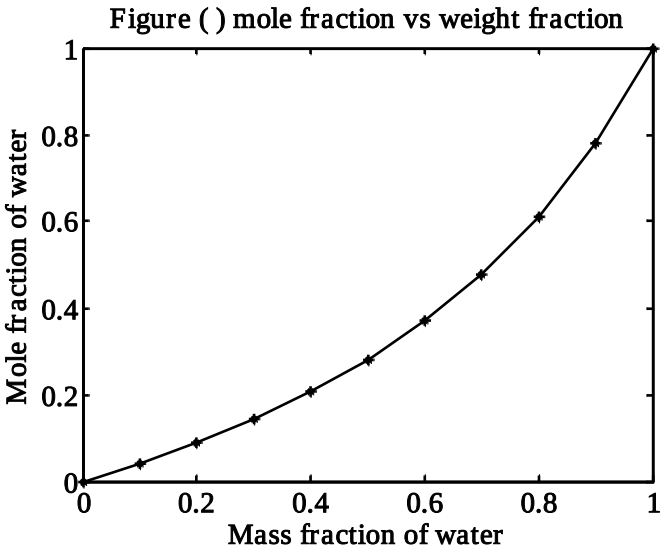
Q 13: Water and ethanol having molar masses [18, 40]x10^{-3}kg/mol respectively, generate a graph that shows the variation of water mole fraction vs water mass fraction. Where the relation between mole and mass fraction is given by:

$$x_A = \frac{X_A \times M_A}{X_A \times M_A + (1 - X_A) \times M_B}$$

Solution:

```
Xw=0:0.1:1;
Mww=18e-3;
Mwe=46e-3;
xw=(Xw*Mww)./(Xw*Mww+(1-Xw)*Mwe)
plot(Xw,xw,'k*-')
xlabel('Mass fraction of water');
ylabel('Mole fraction of water')
title('Figure ( ) mole fraction vs weight fraction')
```

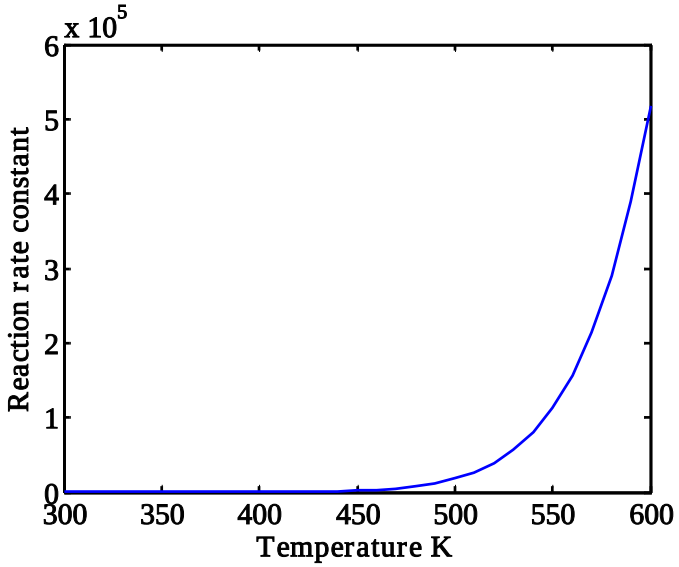
Which generates the figure



Q 14: The reaction rate constant for a first-order reaction given by the Arrhenius law $k=A\ e^{-E/RT}$, at temperatures from 300 K to 600 K and 10 K temperature intervals. Plot the reaction rate constant with respect to temperature and x-axis and y-axis labels if you know that the activation energy is E=20000 cal/mol and the pre-exponential factor is $A=10^{13}\ s^{-1}$ and R=1.987.

Solution:

```
T=300:10:600;
k=10^13*exp(-20000./(1.987*T));
plot(T,k)
xlabel('Temperature K')
ylabel('Reaction rate constant')
```



Q 15: Plot Pxy diagram for the Binary system of acetonitrile(1)/nitromethane(2). Vapor pressures for the pure species are given by the following equations.

For acetonitrile(1) $P^{o}_{1}=\exp(14.2724-2945.47/(T+224))$

For nitromethane(2) $P^{o}_{2}= \exp(14.2043-2972.64/(T+209))$

In which T=75 C^o and P_1 and P_2 in kpa

$K_i = P^o_i / P_t$

At Bubble point $\sum yi = \sum Ki \times xi = 1$

Solution:

Write the following code

```
X1=0:.1:1;
X2=1-X1;
T=75;
P1=exp(14.2724-2945.47/(T+224));% Vapor pressure of acetonitrile
P2=exp(14.2043-2972.64/(T+209));% Vapor pressure of nitromethane
for m=1:11
for Pt=.1:.01:120;
K1=P1/Pt; % acetonitrile
K2=P2/Pt; % nitromethane
sum=K1*X1(m)+K2*X2(m);
if sum<1
break
end
end
Press(m)=Pt;
Y1(m)=K1*X1(m);
end
plot(X1,Press,'k-+',Y1,Press,'k-*')
axis ([0 1 20 100])
xlabel('x,y for acetonitrile')
ylabel('Pt (kpa)')
title('Pxy diagram of system acetonitrile/nitromethane system')
```

