University of Technology
Chemical Engineering Departement

Computer Programming (II)
MATLAB For Chemical Engineer

Second Year

By
Dr. Saad Raheem

Computer Programming (II)
MATLAB for Chemical Engineer

Contents

References Books:

1. Rudra Pratap: Getting started with MATLAB 7, Oxford Press (Indian edition),2006.
2. Desmond J. Higham and Nicolas J. Higham: Matlab Guide, SIAM, 2000.
3. Duane Hanselman and Bruce Littlefield: Mastering Matlab 6: A Comprehensive Tutorial and Reference, Prentice Hall, 2001.
4. Delores M. Etter: Engineering problem solving with Matlab, Prentice Hall,1993.
5. Schilling R. J., Harries S.L., Applied Numerical Methods for Engineers using MATLAB & C,Thomson Books, 2002.

Web Sites:

1. Documentation from www.Mathworks.com
2. File exchange from www.matlabcentral.com
3. http://www.mathworks.com/access/helpdesk/help/techdoc/learn_matlab/
4. http://www.mathworks.com/academia/student_center/tutorials/

Introduction to MATLAB

MATLAB (**MAT**rix **LAB**oratory) is an interactive software system for numerical computations and graphics. As the name suggests, MATLAB is especially designed for matrix computations, solving systems of linear and nonlinear systems of equations, integration of ordinary and partial differential equations, and many others.

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.

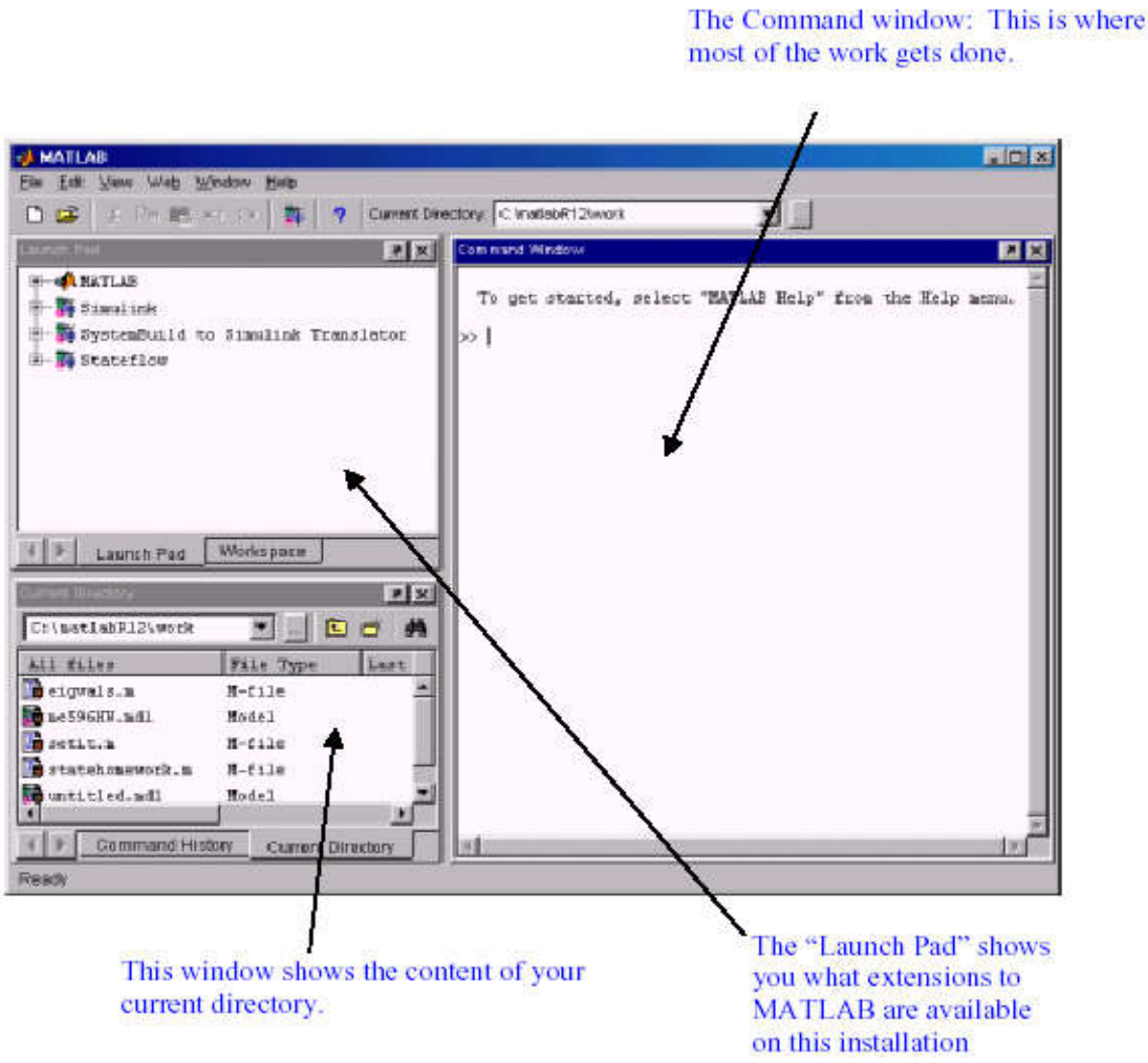


Figure 1: The MATLAB Window

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.

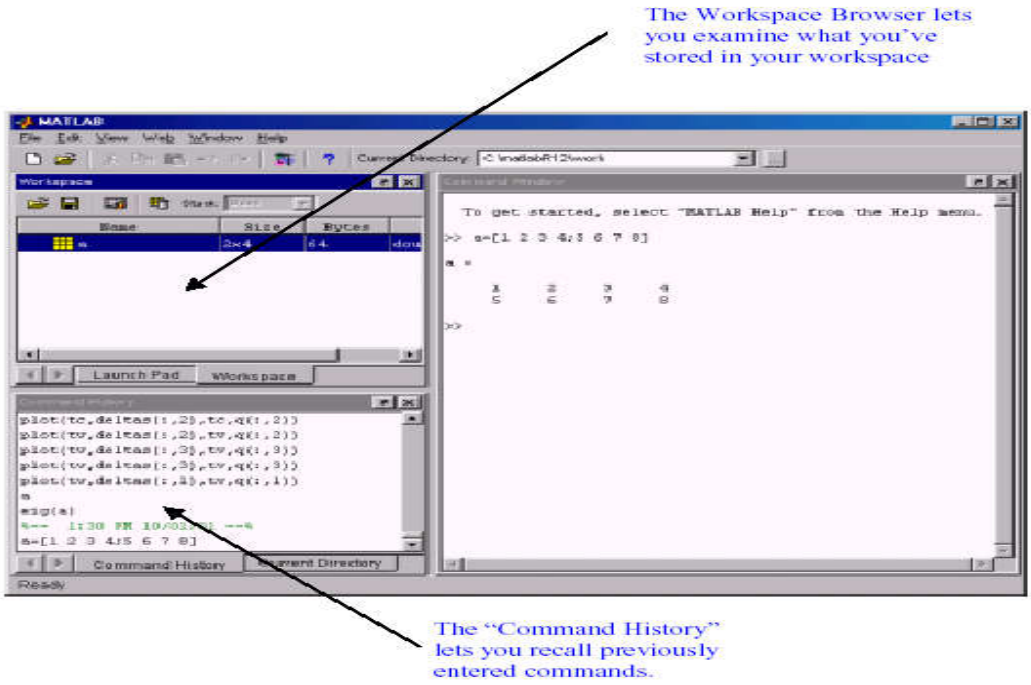


Figure 2: Alternate windows in the default frame

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.

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
```

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

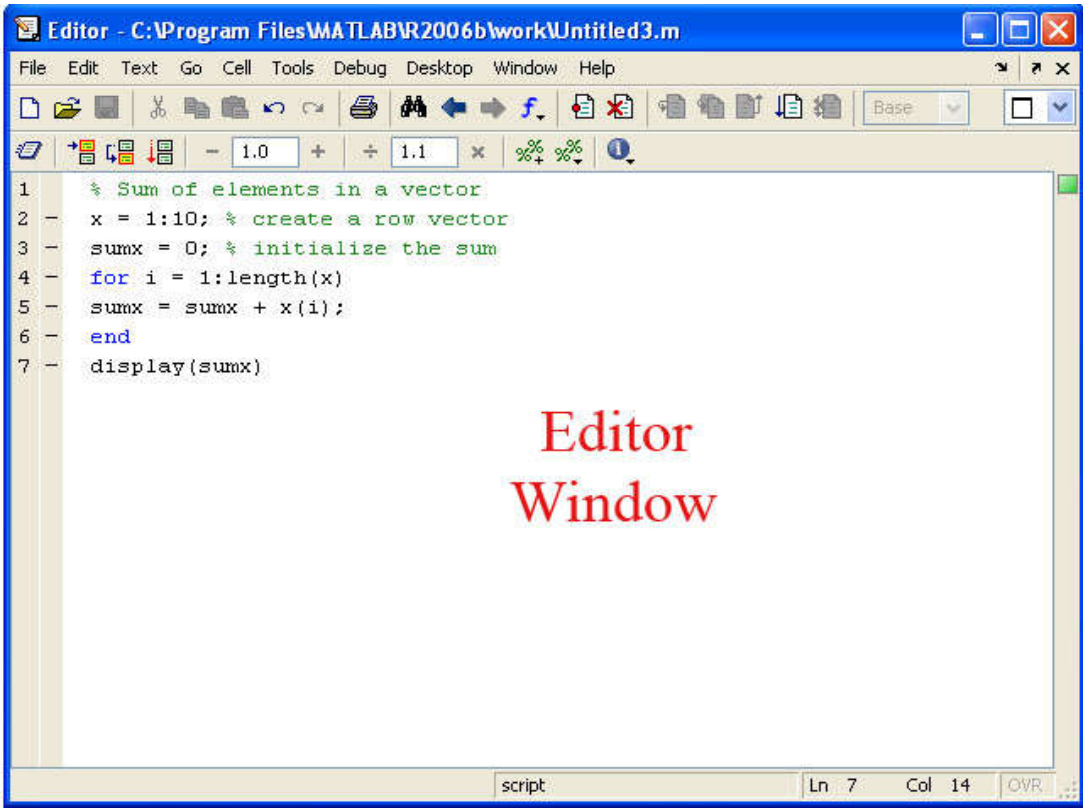


Figure 3: MATLAB Editor Window

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
>> 5^3
ans=
   125
>>3.89*4.1
ans =
   15.9490
>>99.3-25
ans =
   74.3000
>> 3*(23+14.7-4/6)/3.5
ans=
   31.7429
```

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

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

Also type:

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

3. Relational operations

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

A < B Less than

A > B Greater than

A <= B Less than or equal

A >= B Greater than or equal

A == B Equal

A ~= B Not equal

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

Logical AND &

Logical OR |

Logical NOT ~

Colon :

Subscripting ( )

Brackets [ ]

Decimal point .

Continuation ...

Separator ,

Semicolon ; (suppresses the output of a calculation)

Assignment =

Quote ' *statement* '

Transpose '

Comment %

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

4. Variables

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

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

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

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

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

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

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

Of course Matlab will also allow more complicated operations:

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

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

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

4.1 Predefined variables

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

i: sqrt(-1)

j: sqrt(-1)

pi: 3.1416...

For example,

```
>>pi
ans =
   3.1416
>>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
```

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( ).

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

Most common functions are available to Matlab

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

6.3. Complex Number Functions

conj( ) - Complex conjugate.

Imag( ) - Complex imaginary part.

Real( ) - Complex real part.

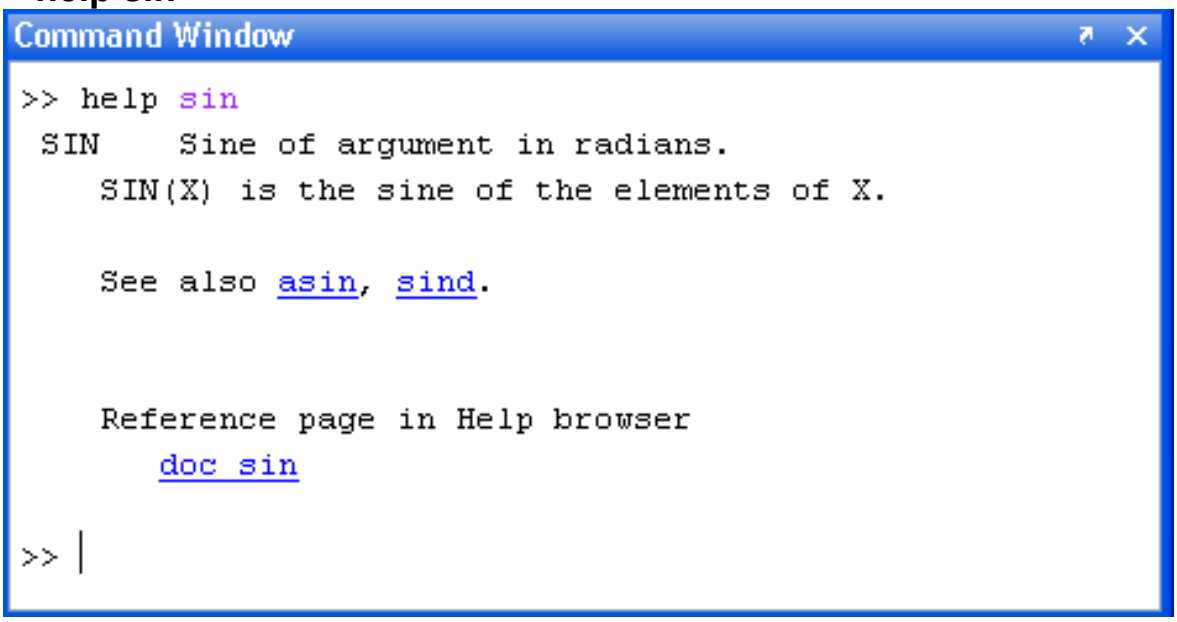
```
>>A=2+4*i, B=conj(A), C=Imag(A), D=real(A)
A =
   2.0000 + 4.0000i
B =
   2.0000 - 4.0000i
C =
   4
D =
   2
```

7. Help

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

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

>> help sin



8. Closing Matlab

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

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

```
>> quit
```

or

```
>> exit
```

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

9. Common commands

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

clear : clears the memory

clear A : clears variable named A from the memory

clc : clears the command window

clf : clears the graphical window

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

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

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

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

Use clear command to delete variables from computer memory

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

Exercise 1:

Write a program to calculate the vapor pressure of water according to Antoine equation: $P^o=\exp(A-B/(T+C))$

Where T is any given temperature in Kelvin and A, B, and C are 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:

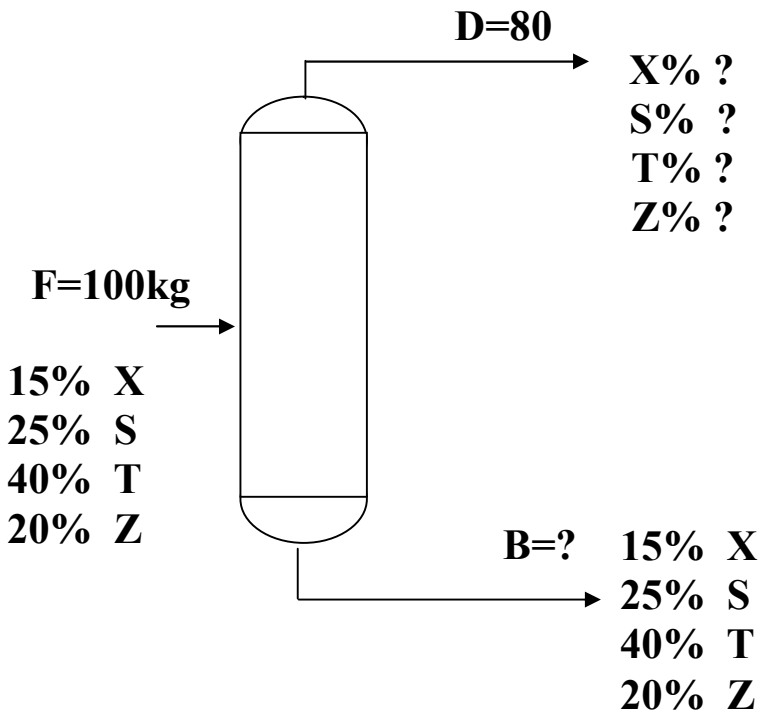
For the following distillation column write a code to find the value of stream B and the compositions of stream D?

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
```



Practice Problems (1)

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

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

2) Solve the following problems in command window

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

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

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

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

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

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

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

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

4) Define the variables a,b and c as: a=5, b=-5, and c=2a+b

Evaluate:

a) $ab + ac - \frac{6a}{b}$

b) $a^{-b/a} + \frac{c^a}{b^{(a+b)/c}}$

5) Compute the reaction rate constant for a first-order reaction given by the Arrhenius law $k=A\, e^{-E/RT}$, at a temperature T=500 K. Here the activation energy is E=20 kcal/mol and the pre-exponential factor is $A=10^{13}$ s^{-1}. The ideal gas constant is R=1.987 cal/mol K.

ALGEBRA

1. Symbolic Toolbox

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

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

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

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

```
>> syms x
```

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

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

To factor x^2+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 solve 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.

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

In expressions with more than one variable, we can solve for one or more of the variables in terms of the others. Here we find the roots of the quadratic 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.

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

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

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

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

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

And you get the following result:

```
x =
 -48/5
 y =
 47/5
>> [x,y]=solve('log(x)+x*y=0', 'x*y+5*y=1')
x =
.8631121967939437
y =
.1705578823046945
```

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

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

And get the solutions

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

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

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

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

In same way if you have more 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('Dy+y=cos(t)')
ans =
1/2*cos(t)+1/2*sin(t)+exp(-t)*C1

>> dsolve('x*Dy-y=1', 'y(1)=5', 'x')
ans =
-1+6*x
```

2.2.2 Second Order Differential 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: $d^2y/dx^2 - 2dy/dx - 3y = x^2$

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

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

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

2.2.3 Higher Order Differential Equations

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

3. Representing Functions

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

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

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

```
>> f(2)
ans =
11
```

The function $g(x,y)=x^2-3xy+2$ 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(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)
```

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)
```

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

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

6. Limits

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

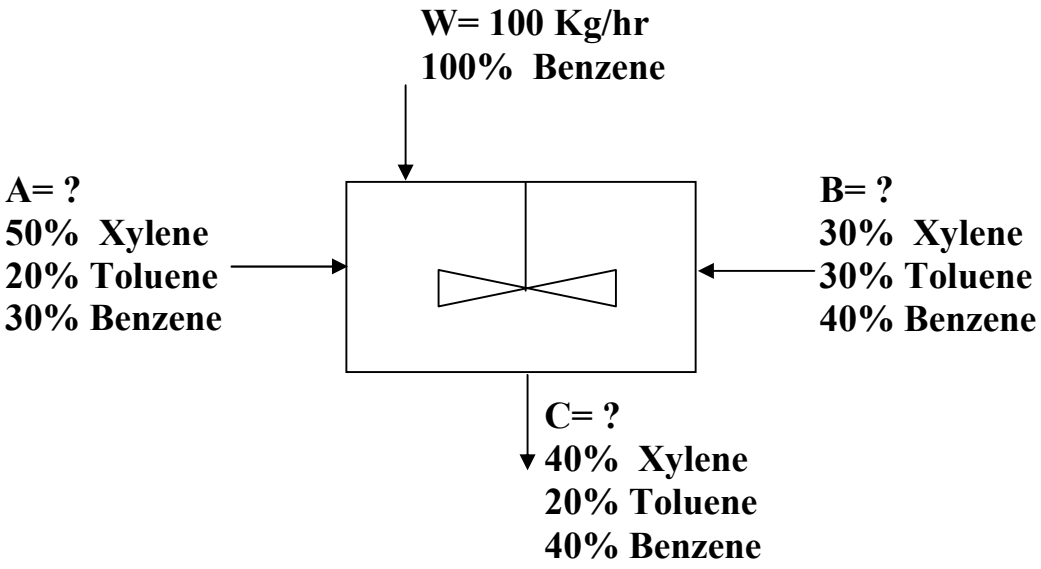
```
>> syms x
>> limit((x^2-4)/(x-2), x, 2)
ans =
4
```

Limits at infinity:

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

Exercise 1:

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



Solution:

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

Type the following command:

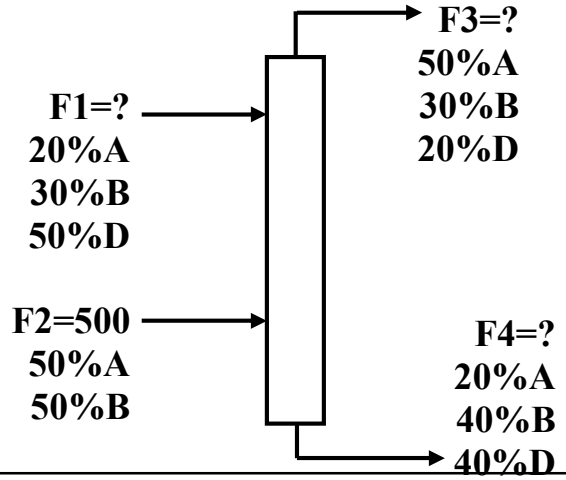
```
[A,B,C]=solve('.5*A+.3*B=.4*C','.2*A+.3*B=.2*C','.3*A+.4*B+100=.4*C')
```

The results will be:

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

Exercise 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 K 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; n=1;
A=1.702; B=9.081e-3; C=-2.164e-6; R=8.314;
Cp=R*(A+B*T+C*T^2);
Q=single(int(Cp,T1,T2))
```

The results will be:

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

Exercise 4:

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
```

Practice Problems (2)

1. Write required code to solve each of the following:-
   a) Factor $x^3+3x^2y+3xy^2+y^3$.
   b) Simplify $(x^3-8)/(x-2)$.
   c) Expand $(x^2+1)(x-5)(2x+3)$.
   d) Solve $\sin x = 2-x$ for x.
   e) Solve $5x+2y+4z = 8$, $-3x+y+2z = -7$, $2x+y+z = 3$ for x, y and z.
   f) Solve $y^2-5xy-y+6x^2+x = 2$ for x.
   g) Find the first derivative of the function $(sinx/(ln(x^2+1))-e^x$ and evaluate it at x=3.
   h) Find the 12^{th} derivative of the function $(x/2+1)^{65}$
   i) Find the first and second partial derivatives for x of the function $e^{x2}sinxy$

2. Obtain the first and second derivatives of the following functions using MATLAB's symbolic mathematics.
   a) $F(x) = x^5 - 8x^4 + 5x^3 - 7x^2 + 11x - 9$
   b) $F(x) = (x^3 + 3x - 8)(x^2 + 21)$
   c) $F(x) = (3x^3 - 8x^2 + 5x + 9)/(x + 2)$
   d) $F(x) = (x^5 - 3x^4 + 5x^3 + 8x^2 - 13)^2$
   e) $F(x) = (x^2 + 8x - 11)/(x^7 - 7x^6 + 5x^3 + 9x - 17)$

3. Chemical engineer, as well as most other engineers, uses 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):
   $C_p = 0.994 + 1.617\times10^{-4}T + 9.7215\times10^{-8}T^2 - 9.5838 \times 10^{-11} T^3 + 1.9520 \times 10^{-14} T^4$
   Determine the temperature that corresponds to a specific heat of 1.2 KJ/(Kg K).

4. Evaluate the triple integral:
   $$\int_0^6 \int_0^6 \int_{-4}^4 (x^3 - 2yz)dx \cdot dy \cdot dz$$

5. 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):
   $C_p = 0.99403 + 1.617\times10^{-4}T + 9.7215\times10^{-8}T^2 - 9.5838 \times 10^{-11} T^3 + 1.9520 \times 10^{-14} T^4$

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
```

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:

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=0.1:0.1:1;
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
```

Exercise 3:

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

$\rho = 1200.92 – 1.0056\ TKo + 0.001084 * (TKo)2$

The condactivity

$K= 0.34 + 9.278 * 10\text{-}4\ .TKo$

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
```

Practice Problems (3)

1. 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.

2. 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".

3. 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

4. Use the **linspace** commands to create 4 vectors a, b, c, d with the elements:
   a) 2, 4, 6, 8… to 10 elements
   b) 10, 8, 6, 4, 2, 0, -2, -4…to 15 elements
   c) 1, 1/2, 1/3, 1/4, 1/5… to 10 elements
   d) 0, 1/2, 2/3, 3/4, 4/5… to 10 elements

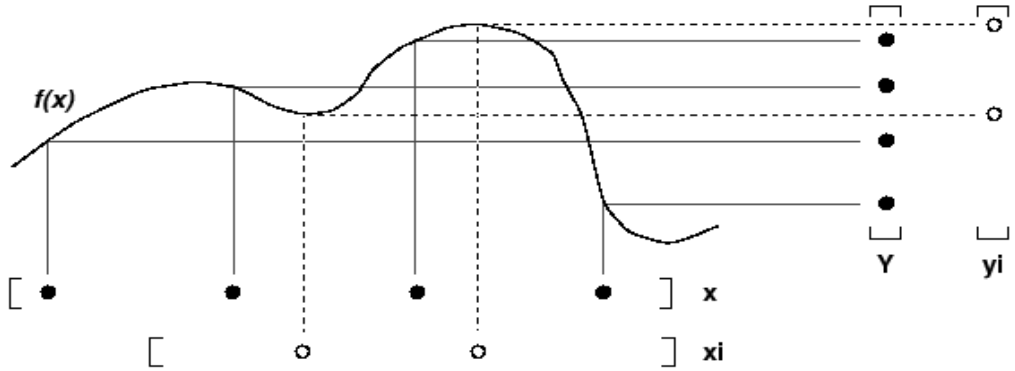
5. Compute the average molecular weight of a mixture of <u>equal</u> fractions of CH_4, H_2, O_2, CO, CO_2, H_2O, CH_3OH, C_2H_6.

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

Interpolation

1. One-Dimensional 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 1:

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 2:

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

T (^oC)	20	50	80	110	140	170	200	230
Cpj/mol.^oC	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 ^oC.

Solution:

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

The results will be

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

Exercise 3:

Chemical engineer, as well as most other engineers, uses 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):

$C_p = 0.994 + 1.617\times10^{-4}T + 9.7215\times10^{-8}T^2 - 9.5838 \times 10^{-11} T^3 + 1.9520 \times 10^{-14} T^4$

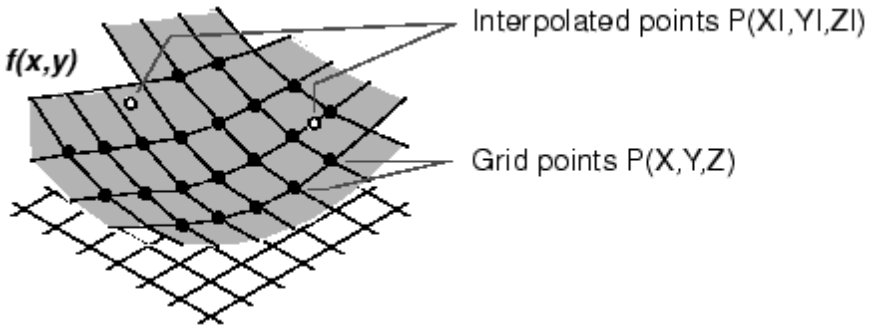
Determine the temperature that corresponds to a specific heat of 1.2 KJ/(Kg K).

Solution:

```
T=10:10:10000;
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)
```

2.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 4:

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 =
   28
```

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

```
z12 = interp2(x,y,z,[1.3,1.5],[1.5,2.3])
z12 =
   28   54
```

Practice Problems (4)

1) 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;

2) 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.

3) 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?

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

$$P^o_{Toluene} = \exp\left(9.0 - \frac{3500}{T+1\ 0}\right)$$ Where: P^o in atm and T in K.

4) The friction factor f depends on the Reynolds number Re for turbulent flow in smooth pipe according to the following relationship.

$$\frac{1}{\sqrt{f}} = -0.40 + \sqrt{3}\ln(\mathrm{Re}\sqrt{f})$$

Write required code to compute **f** for Re = 25200, using ***interp1*** command.

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

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

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]
```

1. 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
```

2. 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
```

3. 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 1:

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 2:

Fit the following data describing the accumulation of species A over time to a second order polynomial, and 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:

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

Now fit the data using polyfit

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

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

Therefore to calculate the mass A at 15 hours

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

Exercise 3:

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 4:

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

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]
Cp=[8.371, 8.472, 8.514, 9.035, 9.824, 10.606, 11.347]
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 (5)

1) Find the 3rd order polynomial that satisfies the data of water for saturation temperature and pressure. By 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) A vapor pressure vs. temperature thermodynamic process has the following data.Write a MATLAB program 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) 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**.

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.

6) 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 $W = \int p dV$. 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.

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
```

Exercise 2:

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:

```
Xe=0.4;
Xw=0.2;
Xb=0.4;
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;
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 (5)

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

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.

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

3) Given the arrays x = [1 3 5], y = [2 4 6] and A = [3 1 5 ; 5 9 7], Calculate;

a) x + y
b) x' + y'
c) A - [x ; y]
d) [x ; y].*A
e) A – 3

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 this 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 explain and demonstrate 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.

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots\ldots\ldots\ldots a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots\ldots\ldots\ldots a_{2n}x_n = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots\ldots\ldots\ldots a_{3n}x_n = b_3$$
$$\vdots \qquad \vdots \qquad \vdots \qquad\qquad \vdots \qquad \vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots\ldots\ldots\ldots a_{nn}x_n = b_n$$

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.

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

$$AX = b$$

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:

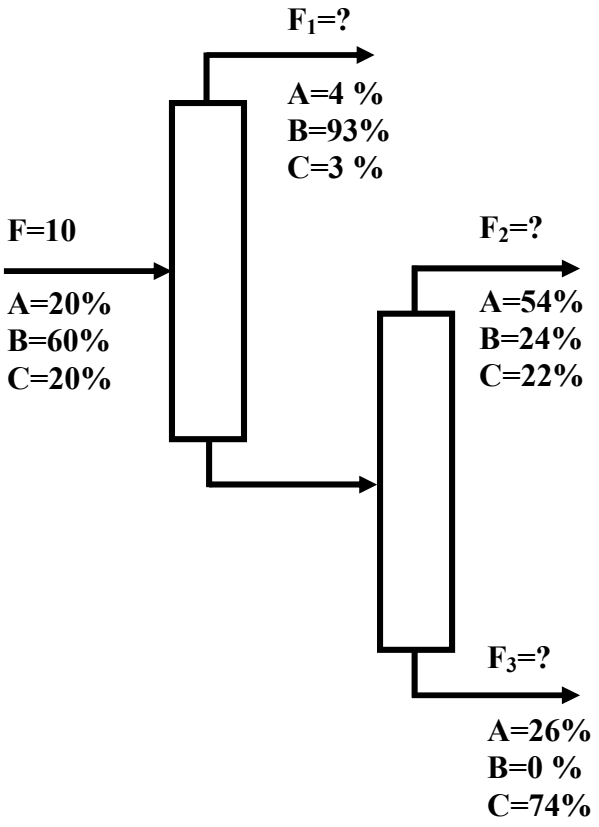
```
x1=x(1)
x2=x(2)
x3=x(3)
```

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 (F_1, F_2 and F_3). We want to calculate the unknown mass flow rates of each outlet stream.



Solution:

Set up the mass balances for

1. the total mass flow rate

$F_1 + F_2 + F_3 = 10$

2. the mass balance on species A

$0.04F_1+0.54F_2+0.26F_3=0.2*10$

3. the mass balance on species B

$0.93F_1+0.24F_2 = 0.6*10$

These three equations can be written in matrix form

$$\begin{bmatrix} 1 & 1 & 1 \\ 0.04 & 0.54 & 0.26 \\ 0.93 & 0.24 & 0 \end{bmatrix} \begin{bmatrix} F1 \\ F2 \\ F3 \end{bmatrix} = \begin{bmatrix} 10 \\ 2 \\ 6 \end{bmatrix}$$

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)
```

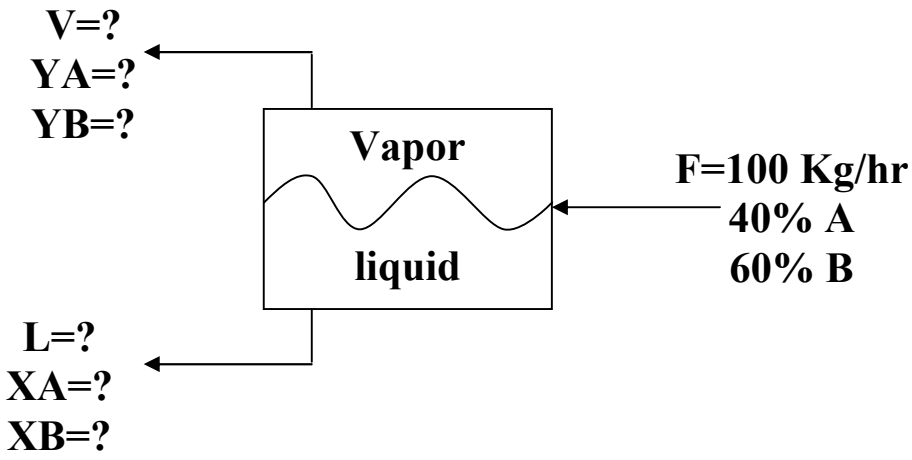
The results will be:

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

Exercise 2:

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:

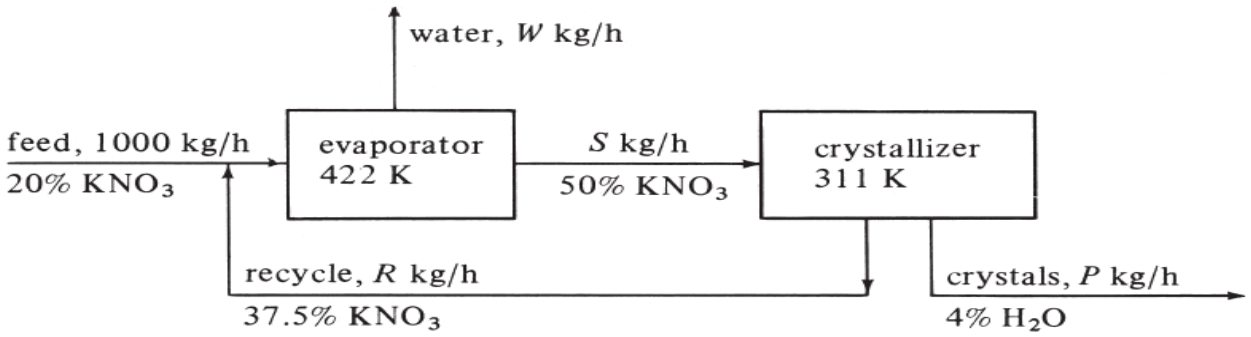
```
A=[1,1,0,0;0,0,1,1;-1.9,0,1,0;0,-0.6,0,1];
B=[1;1;0;0];
X=A\B;
xa=X(1)
xb=X(2)
ya=X(3)
yb=X(4)
a=[xa,ya;xb,yb];
b=[.4*100;.6*100];
x=a\b;
L=x(1),
V=x(2)
```

Gives the results

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

Exercise 3:

In a process producing KNO_3 salt, 1000 kg/h of a feed solution containing 20 wt % KNO_3 is fed to an evaporator, which evaporates some water at 422 K to produce a 50 wt % KNO_3 solution. This is then fed to a crystallizer at 311 K, where crystals containing 96 wt % KNO_3 is removed. The saturated solution containing 37.5 wt % KNO_3 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.



Solution:

Set up the total mass balances for KNO_3 and water:

0.2*1000 = 0.96*P

0.8*1000 = 0.04*P +W

Mass balances on crystallizer for KNO_3 and water:

0.5*S=0.375*R+0.96*P

0.5*S=0.625*R+0.04*P

To find the values of unknown W, S, R and P write the code:

```
A=[0.96, 0, 0, 0;-0.04,1,0,0;-0.96, 0, 0.5,-0.375;-0.04, 0,0.5,-0.625];
B=[200; 800;0;0];
X=A\B;
P=X(1),W=X(2),S=X(3), R=X(4),
```

The results will be:

```
P =
   208.33
W =
   808.3333
S =
   975
R =
   766.6667
```

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

$CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O$

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

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

Practice Problems (6)

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

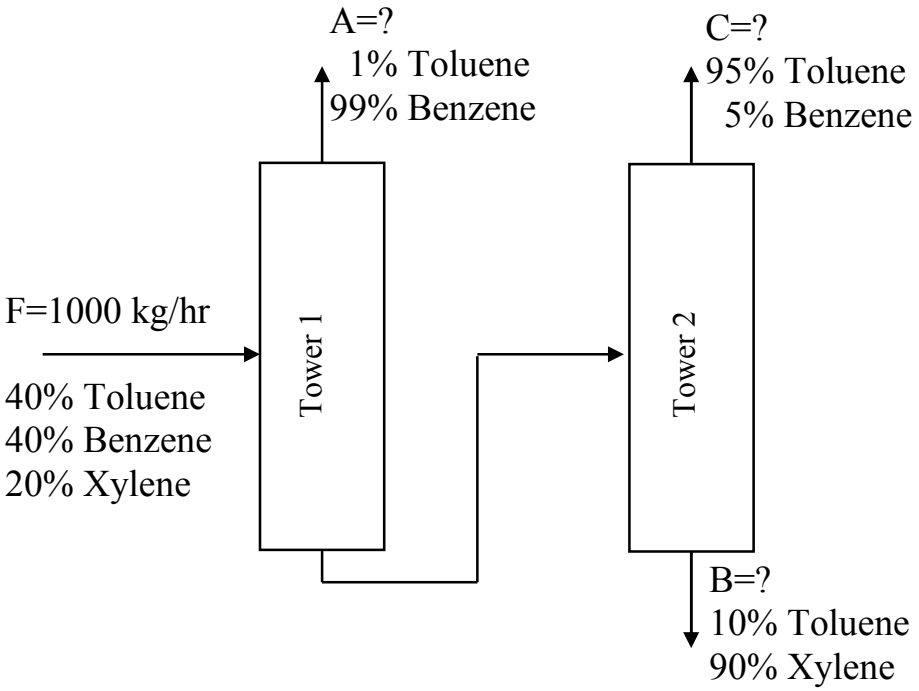
A) $r + s + t + w = 4$
$2r - s + w = 2$
$3r + s - t - w = 2$
$r - 2s - 3t + w = -3$

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

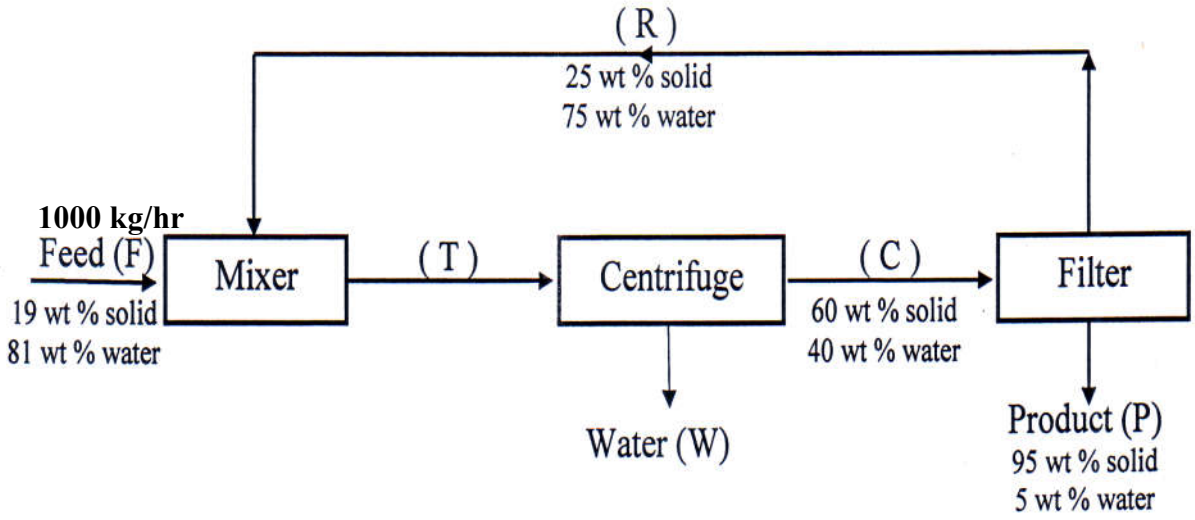
C) $5x_1+3x_2+7x_3-4=0$
$3x_1+26x_2-2x_3-9=0$
$7x_1+2x_2+10x_3-5=0$

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) Write a program to calculate the flow rates of streams W, P, C, R and T in the following flow diagram using **matrix inverse method**.



4) Balance the following chemical equations using the matrix inverse method:

a) $Pb(NO_3)_2 \rightarrow PbO + NO_2 + O_2$

b) $MnO_2 + HCl \rightarrow MnCl_2 + H_2O + Cl_2$

c) $As + NaOH \rightarrow Na_3AsO_3 + H_2$

d) $C_4H_{10} + O_2 \rightarrow CO_2 + H_2O$

e) $BaCl_2 + Al_2(SO_4)_3 \rightarrow BaSO_4 + AlCl_3$