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
```

<u>Note:</u> 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

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