Alexandria University

Faculty of Engineering

Electronics and Communication Department

Signals & Systems
Lab (1) Report

By

Ahmed SamehTaha 21010083
Amir Mohamed Kasseb 21010302

Supervised by:

Eng. Ahmed Mostafa

Eng. Esraa Ragab

Problem 1

Problem1.m

```
%Salary Vector

S=[150 150 150 160];

%New Salary Vector

NewS= (S+10)+0.1*(S+10)
```

A) Printing the New Salary after calculating it .

Command Window

```
>> Problem1

NewS =

   176   176   176   187
```

Problem 2

Problem2.m

```
%Original Vector
V=[2 8 7 3 1 0 8 9];
%New Vector
Newvector=ones(1,length(V));
Newvector(mod(V,2)==0)=-1
```

A) Printing the New Vector after putting -1 in the place of evens and 1 in the place of odds .

```
>> Problem2

Newvector =

    -1    -1     1     1     1    -1    -1     1
```

Problem 3

```
Problem3.m
%Original Vector

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

%Remove Comments For the Results

%A)
V(end-2:end) = V(end-2:end) + 2;
%B)
%V(end-3:end) = V(end:-1:end-3);
%C)
%V(2:2:end) = V(2:2:end) + V(1:2:end-1);
```

A) The First Vector after applying the indexing technique.

```
>> Problem3
>> V

V =

     1
     2
     3
     4
     5
     8
     9
    10
```

B) The Second Vector after applying the indexing technique.

```
Problem3.m
%Original Vector

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

%Remove Comments For the Results

%A)
%V(end-2:end) = V(end-2:end) + 2;
%B)
V(end-3:end) = V(end:-1:end-3);
%C)
%V(2:2:end) = V(2:2:end) + V(1:2:end-1);
```

```
Command Window
>> Problem3
>> V

V =

     1
     2
     3
     4
     8
     7
     6
     5
```

C)The Third Vector after applying the same indexing technique.

```
Command Window
>> Problem3
>> V

V =

     1
     3
     3
     7
     5
    11
     7
    15
```

Problem 4

```
Problem4.m
%Required Vector
A=[1:9 8:-1:1].^2
```

A) The Result of the required Vector

```
>> Problem4

A =

     1     4     9    16    25    36    49    64    81    64    49    36    25    16     9     4     1
```

Problem 5

problem_5.m

```
M=[1 2 3 4; -1 -2 -3 -4; 1 2 3 4; -1 -2 -3 -4;]
M(:,[1:4])=M(:,[4:-1:1])
M([1:4],:)=M([4:-1:1],:)
M(:,[2 3])=M(:,[3 2])
M([1 4],:)=M([4 1],:)
M([1 2 3 4],[1 2 3 4])=M([1 3 4 2], [3 2 4 1])
```

A) Reflect array (M) left-side right

```
M =

     1     2     3     4
    -1    -2    -3    -4
     1     2     3     4
    -1    -2    -3    -4


M =

     4     3     2     1
    -4    -3    -2    -1
     4     3     2     1
    -4    -3    -2    -1
```

B) Reflect array (M) upside down,

```
M =

    -4    -3    -2    -1
     4     3     2     1
    -4    -3    -2    -1
     4     3     2     1
```

C) Swap columns 2 and 3 of array (M)

```
M =

    -4    -2    -3    -1
     4     2     3     1
    -4    -2    -3    -1
     4     2     3     1
```

D) Swap rows 1 and 4 of array (M)

```
M =

     4     2     3     1
     4     2     3     1
    -4    -2    -3    -1
    -4    -2    -3    -1
```

E) Shuffle the rows of (M) from [1 2 3 4] to [1 3 4 2] and shuffle the columns of (M) from [1 2 3 4] to [3 2 4 1].

```
M =

     3     2     1     4
    -3    -2    -1    -4
    -3    -2    -1    -4
     3     2     1     4
```

Problem 6

problem_6.m

```
x=[1:5; zeros(3,5); -1:-1:-5]'
y=x'
z=[x(1:3,:); x(2:-1:1,:)]'
w=[x(:,1)*2 x(:,2:4)+100 x(:,5)/-10]
```

Generate Matrix x:

```
x =

     1     0     0     0    -1
     2     0     0     0    -2
     3     0     0     0    -3
     4     0     0     0    -4
     5     0     0     0    -5
```

Matrix y:

```
y =

     1     2     3     4     5
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
    -1    -2    -3    -4    -5
```

Matrix z:

```
z =

     1     2     3     2     1
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
    -1    -2    -3    -2    -1
```

Matrix w:

```
w =

    2.0000  100.0000  100.0000  100.0000    0.1000
    4.0000  100.0000  100.0000  100.0000    0.2000
    6.0000  100.0000  100.0000  100.0000    0.3000
    8.0000  100.0000  100.0000  100.0000    0.4000
   10.0000  100.0000  100.0000  100.0000    0.5000
```

Problem 7

Part 1

A) Use the 'zeros' function to create an empty 5 -by-5 matrix (A) for the coefficients and an empty 5 -by-1 column vector (B) for the RHSs of the equations.

```
A=zeros(5)
B=zeros(5,1)
```

```
Command Window
>> problem_7

A =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0


B =

     0
     0
     0
     0
     0
```

B) Use the array editor to populate the coefficients into matrix (A).

A
5x5 double

	1	2	3	4	5	6
1	2	3	5	6	21	
2	5	0	3	2	0	
3	6	7	8	9	11	
4	13	0	17	5	6	
5	1	4	0	3	9	
6						
7						
8						

C) Use the command prompt to populate the RHSs into vector (B).

Command Window

```
B =

     0
     0
     0
     0
     0

>> B=[152; 19; 135; 127; 66]

B =

   152
    19
   135
   127
    66
```

D) we can use the 'rank' function to find out whether or not the 5 equations are "independent" enough to assign unique solutions for the 5 unknowns, by Comparing the rank & the number of unknowns.
If the rank is equal to 5, it means that the system of equations has a unique solution. If the rank is less than 5, the equations are dependent, and the system may have infinite solutions or the system is inconsistent.

E) Write down an expression for a variable S that is true if the rank of (A) is equal to the number of variables in the case above and false otherwise.

By using size(A, dim), where A is the matrix and dim specifies the dimension which is 2D Matrix.

```
A=zeros(5)
B=zeros(5,1)
A=[2 3 5 6 21; 5 0 2 2 0; 6 7 8 9 11; 0 13 17 5 6; 1 4 0 3 9]

B=[152; 19; 135; 127; 66]

S = (rank(A) == size(A, 2))
```

Command Window

```
     0    13    17     5     6
     1     4     0     3     9


B =

   152
    19
   135
   127
    66


S =

  logical

   1
```

F) Solve the set of linear equations for x_1 through x_5.
AX=B
X=inverse of A * B

problem_7.m

```
A=zeros(5)
B=zeros(5,1)
A=[2 3 5 6 21; 5 0 2 2 0; 6 7 8 9 11; 0 13 17 5 6; 1 4 0 3 9]

B=[152; 19; 135; 127; 66]

S = (rank(A) == size(A, 2))

X=inv(A)*B
```

Command Window

```
   127
    66


S =

  logical

   1


X =

    1.0000
    2.0000
    3.0000
    4.0000
    5.0000
```

Part (2)

Use the Matlab help system to find the names:

1. The exponential function

problem_7.m

```
A=zeros(5)
B=zeros(5,1)
A=[2 3 5 6 21; 5 0 2 2 0; 6 7 8 9 11; 0 13 17 5 6; 1 4 0 3 9]

B=[152; 19; 135; 127; 66]

S = (rank(A) == size(A, 2))

X=inv(A)*B

help exp
```

Command Window

```
X =

    1.0000
    2.0000
    3.0000
    4.0000
    5.0000

 exp    Exponential.
    exp(X) is the exponential of the elements of X, e to the X.
    For complex Z=X+i*Y, exp(Z) = exp(X)*(COS(Y)+i*SIN(Y)).

    See also expm1, log, log10, expm, expint.

    Reference page for exp
    Other functions named exp
```

2. A function that returns the natural logarithm

problem_7.m

```
B=zeros(5,1)
A=[2 3 5 6 21; 5 0 2 2 0; 6 7 8 9 11; 0 13 17 5 6; 1 4 0 3 9]

B=[152; 19; 135; 127; 66]

S = (rank(A) == size(A, 2))

X=inv(A)*B

help exp
help log
```

ommand Window

```
exp    Exponential.
   exp(X) is the exponential of the elements of X, e to the X.
   For complex Z=X+i*Y, exp(Z) = exp(X)*(COS(Y)+i*SIN(Y)).

   See also expm1, log, log10, expm, expint.

   Reference page for exp
   Other functions named exp

log    Natural logarithm.
   log(X) is the natural logarithm of the elements of X.
   Complex results are produced if X is not positive.

   See also log1p, log2, log10, exp, logm, reallog.

   Reference page for log
   Other functions named log
```

3. Functions that return the base-2 and base-10 logarithms

Command Window

```
log2   Base 2 logarithm and dissect floating point number.
   Y = log2(X) is the base 2 logarithm of the elements of X.

   [F,E] = log2(X) for each element of the real array X, returns an
   array F of real numbers, usually in the range 0.5 <= abs(F) < 1,
   and an array E of integers, so that X = F .* 2.^E.  Any zeros in X
   produce F = 0 and E = 0.  This corresponds to the ANSI C function
   frexp() and the IEEE floating point standard function logb().

   See also log, log10, pow2, nextpow2, realmax, realmin.

   Reference page for log2
   Other functions named log2

log10  Common (base 10) logarithm.
   log10(X) is the base 10 logarithm of the elements of X.
   Complex results are produced if X is not positive.
```

4. A function that gets the square root of a number

```
sqrt   Square root.
   sqrt(X) is the square root of the elements of X. Complex
   results are produced if X is not positive.

   See also sqrtm, realsqrt, hypot.
```

5. “Sound” & “Image” commands

```
sound Play vector as sound.
   sound(Y,FS) sends the signal in vector Y (with sample frequency
   FS) out to the speaker on platforms that support sound. Values in
   Y are assumed to be in the range -1.0 <= y <= 1.0. Values outside
   that range are clipped.  Stereo sounds are played, on platforms
   that support it, when Y is an N-by-2 matrix.

   sound(Y) plays the sound at the default sample rate of 8192 Hz.

   sound(Y,FS,BITS) plays the sound using BITS bits/sample if
   possible.  Most platforms support BITS=8 or 16.

   Example:
     load handel
     sound(y,Fs)
   You should hear a snippet of Handel's Hallelujah Chorus.
```

```
image Display image from array
   image(C) displays the data in array C as an image. Each element of C
   specifies the color for 1 pixel of the image. The resulting image is an
   m-by-n grid of pixels where m is the number of columns and n is the
   number of rows in C. The row and column indices of the elements
   determine the centers of the corresponding pixels.

   When C is a 2-dimensional m-by-n matrix, the elements of C are used as
   indices into the current COLORMAP to determine the color.  The value of
   the image object's CDataMapping property determines the method used to
   select a colormap entry.  For 'direct' CDataMapping (the default),
   values in C are treated as colormap indices (1-based if double, 0-based
   if uint8 or uint16).  For 'scaled' CDataMapping, values in C are first
   scaled according to the axes CLim and then the result is treated as a
   colormap index.  When C is a 3-dimensional m-by-n-by-3 matrix, the
   elements in C(:,:,1) are interpreted as red intensities, in C(:,:,2) as
   green intensities, and in C(:,:,3) as blue intensities, and the
   CDataMapping property of image is ignored.  For matrices containing
   doubles, color intensities are on the range [0.0, 1.0].  For uint8 and
   uint16 matrices, color intensities are on the range [0, 255].

   image(C) places the center of element C(1,1) at (1,1) in the axes, and
   the center of element (M,N) at (M,N) in the axes, and draws each
   rectilinear patch as 1 unit in width and height.  As a result, the
   outer extent of the image occupies [0.5 N+0.5 0.5 M+0.5] of the axes,
   and each pixel center of the image lies at integer coordinates ranging
   between 1 and M or N.

   image(x,y,C) specifies the image location. Use x and y to specify the
   locations of the corners corresponding to C(1,1) and C(m,n). To specify
   both corners, set x and y as two-element vectors. To specify the first
   corner and let image determine the other, set x and y as scalar values
```