FINAL LAB REPORT

Optimization Techniques

NAME: M Gyanada Chowdary

REG.NO: 21BCE7727

LAB SLOT: L41,42

PROF: Shalini Thakur

LIST OF EXPERIMENTS:

- Fundamentals of Matlab
- Linear Programming Problems
- Evaluating Maximum profit of planning products(Simplex Method)
- Evaluating Maximum profit of planning products(Simplex Method)
- Manufacturing Problems(Big-M Method)
- Transportation Problems(NWCM, MMM,VAM)
- Assignment Problems (Hungerian Method)
- Nonlinear Programming Problem(Golden Search Method)
- OPTIMISATION CODES.

Introduction to Optimization Techniques:

Optimization,also known as mathematical programming,collection of mathematical principles and methods used for solving quantitative problems in many disciplines,including physics,biology,engineering,economics and business.
The subject grew from a realization that quantitative problems in manifestly different disciplines have important mathematical elements in common.Because of this commonality,many problems can be formulated and solved by using the unified set of ideas and methods that make up the field of optimization.

Introduction to MATLAB:

MATLAB stands for Matrix Laboratory. It is a high-performance language that is used for technical computing.It was developed by Cleve Molar of the company MathWorks. Inc in the year 1984. It is written in C,C++,Java.
It allows matrix manipulations,plotting of functions,implementations of algorithms and creation of user interfaces.

Getting started with MATLAB:

It is both a programming language as well as a programming environment.

Command Window:

In this window one must type and immediately execute the statements,as it requires quick prototyping.

Editor [Script]:

In this window, one can execute larger programs with multiple statements,and complex functions. The files can be saved with the extension '.m'.

Workspace:

In this window the values of the variables that are created in the course of the program are displayed.
It displays the exact location of the program file being created

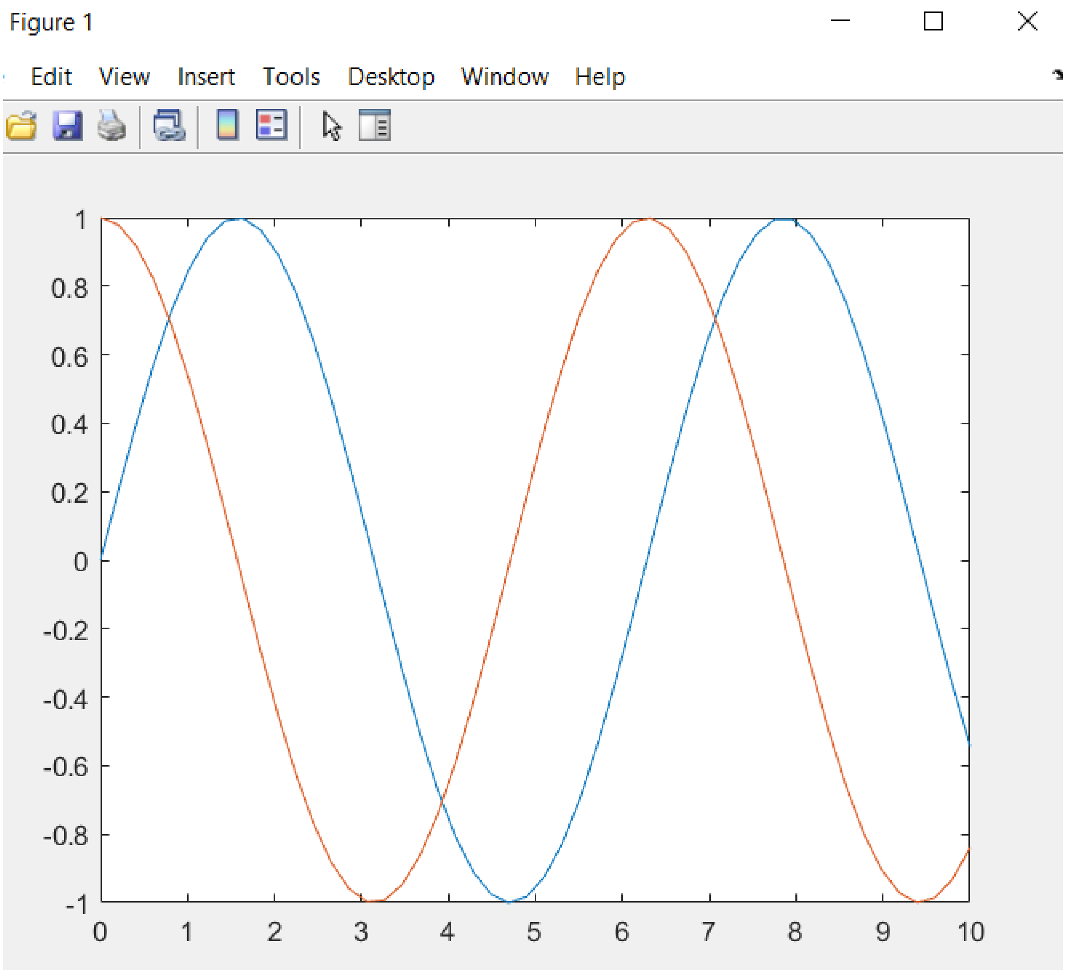
1)Fundamentls of matlab


```
1)
x=2;
y=sqrt(x^2+5*x+2);

2)
r=sin(x);
plot(r,x)

3)
x=linspace(0,10,50)
y1=sin(x);
plot(x,y1);
hold on
y2=cos(x);
plot(x,y2);
hold off
```

4)
y = atan(x)+exp(x)+2==0;




```
x =

  Columns 1 through 18

         0    0.2041    0.4082    0.6122    0.8163    1.0204    1.2245    1.4286    1.6327    1.8367    2.0408    2.2449    2.4490    2.6531    2.8571    3.0612    3.2653    3.4694

  Columns 19 through 36

    3.6735    3.8776    4.0816    4.2857    4.4898    4.6939    4.8980    5.1020    5.3061    5.5102    5.7143    5.9184    6.1224    6.3265    6.5306    6.7347    6.9388    7.1429

  Columns 37 through 50

    7.3469    7.5510    7.7551    7.9592    8.1633    8.3673    8.5714    8.7755    8.9796    9.1837    9.3878    9.5918    9.7959   10.0000
```

2)

Linear Programming Problems:

Question:1

The Two Mines Company own two different mines that produce an ore which, after being crushed, is graded into three classes: high, medium and low-grade. The company has contracted to provide a smelting plant with 12 tons of high-grade, 8 tons of medium-grade and 24 tons of low-grade ore per week. The two mines have different operating characteristics as detailed below.

Mine	Cost per day (£'000)	Production (tons/day)		
		High	Medium	Low
X	180	6	3	4
Y	160	1	1	6

Consider that mines cannot be operated in the weekend. How many days per week should each mine be operated to fulfill the smelting plant contract?

Step 1: Define the decision variables

Let x1 be the number of days Mine X is operated in a week

Let x2 be the number of days Mine Y is operated in a week

Step 2: Define the objective function

The objective is to minimize the operating cost of the mines, which is given by:

Cost = 180x1 + 160x2

Step 3: Define the constraints

The constraints are as follows:

6x1 + x2 >= 12 (high grade)

3x1 + x2 >= 8 (medium grade)

4x1 + 6x2 >= 24 (low grade)

x1, x2 >= 0 (non-negative)

Matlab code:

```
% Define the decision variables

% x1 = number of days Mine X is operated per week

% x2 = number of days Mine Y is operated per week

f = [180; 160]; % objective function coefficients

A = [-6 -1; -3 -1; -4 -6]; % constraint coefficients
```

b = [-12; -8; -24]; % constraint RHS

lb = [0; 0]; % lower bounds

ub = [inf; inf]; % upper bounds

x = linprog(f, A, b, [], [], lb, ub); % solve the LP

disp(x);

Output:

```
>> %21BCE7727
%M Gyanada Chowdary
% Define the decision variables
% x1 = number of days Mine X is operated per week
% x2 = number of days Mine Y is operated per week
f = [180; 160]; % objective function coefficients
A = [-6 -1; -3 -1; -4 -6]; % constraint coefficients
b = [-12; -8; -24]; % constraint RHS
lb = [0; 0]; % lower bounds
ub = [inf; inf]; % upper bounds
x = linprog(f, A, b, [], [], lb, ub); % solve the LP
disp(x);

Optimal solution found.

    1.7143
    2.8571
```

Question: 2

Giapetto's wooden soldiers and trains. Each soldier sells for $27, uses $10 of raw materials and takes $14 of labor & overhead costs. Each train sells for $21, uses $9 of raw materials, and takes $10 of overhead costs. Each soldier needs 2 hours finishing and 1 hour carpentry; each train needs 1 hour finishing and 1 hour carpentry. Raw materials are unlimited, but only 100 hours of finishing and 80 hours of carpentry are available each week. Demand for trains is unlimited; but at most 40 soldiers can be sold each week. How many of each toy should be made each week to maximize profits?

Let x be the number of soldiers produced each week and y be the number of trains produced each week.

We want to maximize the profit, which is given by:

Profit = 27x + 21y - 10x - 9y - 14x - 10y

Profit = 3x + 2y

subject to the following constraints:

2x + y <= 100 (Finishing time constraint)

x + y <= 80 (Carpentry time constraint)

x <= 40 (Soldier demand constraint)

where x, y >= 0.

Matlab Code:

```
Command Window
>> %21BCE7727
%M Gyanada Chowdary
f = [-3; -2];
A = [2 1; 1 1; 1 0];
b = [100; 80; 40];
lb = [0; 0];
ub = [];
[x, fval, exitflag] = linprog(f, A, b, [], [], lb, ub);
disp(x);

Optimal solution found.

    20
    60
```

3) Evaluating Maximum profit of planning products(Simplex Method):

Question:

A sports factory prepares cricket bats and hockey sticks. A cricket nat bat takes 2 hours of machine time and 3 hours of craftsman's time. A hockey stick take 3 hours of machine time and 2 hours of craftsman's time. The factory has 90 hours of machine time and 85 hours of craftsman's time. What number of bats and sticks must be made if the factory is to work at full capacity ? If the profit on a bat is Rs. 3 and on a stick it is Rs. 4, find the maximum profit.

Matlab Code:

```
%21BCE7727
%M Gyanada Chowdary
% Define the problem data
A = [2 3; 3 2];
b = [90; 85];
c = [3; 4];

% Initialize the simplex table
T = [A eye(2) b; c' zeros(1, 3)];

% Perform the simplex iterations
for iter = 1:3
   % Determine the pivot column
   [~, col] = max(T(end, 1:end-1));

   % Determine the pivot row
   ratios = T(1:end-1, end) ./ T(1:end-1, col);
   ratios(ratios <= 0) = inf;
   [~, row] = min(ratios);
```

```
    % Perform the pivot operation
    T(row, :) = T(row, :) / T(row, col);
    for i = 1:size(T, 1)
        if i ~= row
            T(i, :) = T(i, :) - T(i, col) * T(row, :);
        end
    end

    % Display the simplex table
    fprintf('Iteration %d:\n', iter)
    disp(T)
end

% Extract the optimal solution and objective value
x = T(1:2, end);
obj = T(end, end);

% Display the results
fprintf('The optimal solution is x = [%d; %d]\n', x)
fprintf('The maximum profit is Rs. %.2f\n', obj)
```

```
>> %21BCE7727
%M Gyanada Chowdary
% Define the problem data
A = [2 3; 3 2];
b = [90; 85];
c = [3; 4];

% Initialize the simplex table
T = [A eye(2) b; c' zeros(1, 3)];

% Perform the simplex iterations
for iter = 1:3
    % Determine the pivot column
    [~, col] = max(T(end, 1:end-1));

    % Determine the pivot row
    ratios = T(1:end-1, end) ./ T(1:end-1, col);
    ratios(ratios <= 0) = inf;
    [~, row] = min(ratios);

    % Perform the pivot operation
    T(row, :) = T(row, :) / T(row, col);
    for i = 1:size(T, 1)
        if i ~= row
            T(i, :) = T(i, :) - T(i, col) * T(row, :);
        end
    end

    % Display the simplex table
    fprintf('Iteration %d:\n', iter)
    disp(T)
end
```

```
% Extract the optimal solution and objective value
x = T(1:2, end);
obj = T(end, end);

% Display the results
fprintf('The optimal solution is x = [%d; %d]\n', x)
fprintf('The maximum profit is Rs. %.2f\n', obj)
Iteration 1:
    0.6667    1.0000    0.3333         0   30.0000
    1.6667         0   -0.6667    1.0000   25.0000
    0.3333         0   -1.3333         0 -120.0000

Iteration 2:
         0    1.0000    0.6000   -0.4000   20.0000
    1.0000         0   -0.4000    0.6000   15.0000
         0         0   -1.2000   -0.2000 -125.0000

Iteration 3:
         0    1.0000    0.6000   -0.4000   20.0000
    1.0000         0   -0.4000    0.6000   15.0000
         0         0   -1.2000   -0.2000 -125.0000

The optimal solution is x = [20; 15]
The maximum profit is Rs. -125.00
```

4) Evaluating Maximum profit of planning products(Simplex Method)

Question:

A foundry is faced with a problem of scheduling production and subcontracting for three products, each requiring casting, machining and assembly operations. Casting operation for product 1 and 2 could be subcontracted but the castings for product 3 require special equipment and hence cannot be subcontracted. In foundry each unit of product 1 requires 6 minutes of casting time, product 2 requires 10 minutes and product 3 requires 8 minutes of casting time. Machining times per unit of products 1, 2 and 3 are 6, 3 and 8 minutes while assembly times are 3, 2 and 2 minutes respectively. The time available per week for casting, machining and assembly is 8,000, 12,000 and 10,000 minutes respectively. The overall profits obtained per unit of product 1, 2 and 3 are ₹ 7, ₹ 10 and ₹ 11 respectively with castings produced in foundry. With castings obtained from subcontractor the profit per unit for product 1 and 2 are ₹ 5 and ₹ 9 respectively. How should the foundry schedule its production and subcontracting so as to maximize the profit ? Formulate mathematical model only. [*P.U.B.E. (Prod.), 2001*]

LPP FORMULATION:

Let x1, x2, and x3 be the number of units of products 1, 2, and 3 produced in-house

Using the foundry's own casting equipment, respectively.

Let y1, y2, and y3 be the number of units of products 1, 2, and 3 subcontracted

using the casting services of a subcontractor, respectively.

OBJECTIVE FUNCTION:

Z = 7x1 + 10x2 + 11x3 + 5y1 + 9y2

Subject to constraints:

$$6x1 + 10x2 + 8x3 + 6y1 + 10y2 <= 8000 \ (Casting\ constraint)$$

$$6x1 + 3x2 + 8x3 <= 12000 \ (Machining\ constraint)$$

$$3x1 + 2x2 + 2x3 <= 10000 \ (Assembly\ constraint)$$

$$x1, x2, x3, y1, y2 >= 0 \ (Non-negative\ constraint)$$

MATLAB CODE:

```
Noofvariables=5;
C=[7 10 11 5 9];
Coeff=[6 10 8 6 10;6 3 8 0 0;3 2 2 0 0];
b=[8000;12000;10000];
s=eye(size(Coeff,1));
A=[Coeff s b];
cost=zeros(1,size(A,2));
cost(1:Noofvariables)=C;
% % %%Constraint BV
BV=Noofvariables+1:1:size(A,2)-1;
%%% Calculate zj-cj row
zjcj=cost(BV)*A-cost;
%
%%% for print the table
zcj=[zjcj;A];
SimpleTable=array2table(zcj);
SimpleTable.Properties.VariableNames(1:size(zcj,2))
```

```
={ 'x_1' ; 'x_2' ;'x_3';'x_4';'x5';'s_1' ; 's_2' ; 's_3';'sol' };
%Simplex method ierations strating
RUN=true;
while RUN
if any(zjcj<0) %%check if any negativevalue there
fprintf( 'The current BFS is not optimal\n' )
fprintf( '\n The next iteration\n' )
disp( 'Old Basic variable (BV)' );
disp(BV);
%%find the entering variable
zc=zjcj(1:end-1);
[EnterCol,pvt_col]=min(zc);
fprintf( 'the most negative element in zjcj row is %d corresponding
column%d \n' ,EnterCol, pvt_col);
fprintf( 'Entering variable is %d \n' ,pvt_col);
%% Finding the leaving variable
if all(A(:,pvt_col)<=0)
error( 'LPP is unbounded all enteries <= in column %d' , pvt_col);
else
sol=A(:,end);
column=A(:,pvt_col);
for i=1:size(A,1)
if column(i)>0
ratio(i) =sol(i)./column(i);
else
ratio(i)=inf;
end
end
%% finding minimum
[MinRatio,pvt_row]=min(ratio);
fprintf( 'Minimum Ratio corresponding to pivot row is %d \n' , pvt_row);
fprintf( 'Leaving variable is %d \n' ,BV(pvt_row));
end
```

```
BV(pvt_row)=pvt_col;
disp( 'New basic variable (BV)=' );
disp(BV);
%%pivot key
pvt_key=A(pvt_row,pvt_col);
%update the table for next iteration
A(pvt_row,:)=A(pvt_row,:)./pvt_key;
for i=1:size(A,1)
if i~=pvt_row
A(i,:)=A(i,:)-A(i,pvt_col).*A(pvt_row,:);
disp(SimpleTable);
end
end
zjcj=zjcj-zjcj(pvt_col).*A(pvt_row,:);
%%for printing purpose
zcj=[zjcj;A];
TABLE=array2table(zcj);
TABLE.Properties.VariableNames(1:size(zcj,2))
={ 'x_1' ; 'x_2' ; 'x_3';'x_4';'x5';'s_1' ;'s_2' ; 's_3'; 'sol' };
BFS=zeros(1,size(A,2));
BFS(BV)=A(:,end);
BFS(end)=sum(BFS.*cost);
Current_BFS=array2table(BFS);
Current_BFS.Properties.VariableNames(1:size(Current_BFS,2))
={ 'x_1' ; 'x_2' ;'x_3';'x_4';'x5';'s_1' ; 's_2' ; 's_3';'sol' };
else
RUN=false;
fprintf( '======****====== \n' )
fprintf( 'The current BFS is optimal \n' )
fprintf( '======****====== \n' )
disp(TABLE);
end
end
```

```
>> Noofvariables=5;
C=[7 10 11 5 9];
Coeff=[6 10 8 6 10;6 3 8 0 0;3 2 2 0 0];
b=[8000;12000;10000];
s=eye(size(Coeff,1));
A=[Coeff s b];
cost=zeros(1,size(A,2));
cost(1:Noofvariables)=C;
% % %%Constraint BV
BV=Noofvariables+1:1:size(A,2)-1;
%%% Calculate zj-cj row
zjcj=cost(BV)*A-cost;
%
%%% for print the table
zcj=[zjcj;A];
SimpleTable=array2table(zcj);
SimpleTable.Properties.VariableNames(1:size(zcj,2))={ 'x_1' ; 'x_2' ;'x_3';'x_4';'x5';'s_1' ; 's_2' ; 's_3';'sol' };
%Simplex method ierations strating
RUN=true;
while RUN
if any(zjcj<0) %%check if any negativevalue there
fprintf( 'The current BFS is not optimal\n' )
fprintf( '\n The next iteration\n' )
disp( 'Old Basic variable (BV)' );
disp(BV);
%%find the entering variable
zc=zjcj(1:end-1);
[EnterCol,pvt_col]=min(zc);
fprintf( 'the most negative element in zjcj row is %d corresponding column;%d \n' ,EnterCol, pvt_col);
fprintf( 'Entering variable is %d \n' ,pvt_col);
```

Command Window

```
%% Finding the leaving variable
if all(A(:,pvt_col)<=0)
error( 'LPP is unbounded all enteries <= in column %d' , pvt_col);
else
sol=A(:,end);
column=A(:,pvt_col);
for i=1:size(A,1)
if column(i)>0
ratio(i) =sol(i)./column(i);
else
ratio(i)=inf;
end
end
%% finding minimum
[MinRatio,pvt_row]=min(ratio);
fprintf( 'Minimum Ratio corresponding to pivot row is %d \n' , pvt_row);
fprintf( 'Leaving variable is %d \n' ,BV(pvt_row));
end
BV(pvt_row)=pvt_col;
disp( 'New basic variable (BV)=' );
disp(BV);
%%pivot key
pvt_key=A(pvt_row,pvt_col);
%update the table for next iteration
A(pvt_row,:)=A(pvt_row,:)./pvt_key;
for i=1:size(A,1)
if i~=pvt_row
A(i,:)=A(i,:)-A(i,pvt_col).*A(pvt_row,:);
disp(SimpleTable);
end
end
zjcj=zjcj-zjcj(pvt_col).*A(pvt_row,:);
```

```
%%for printing purpose
zcj=[zjcj;A];
TABLE=array2table(zcj);
TABLE.Properties.VariableNames(1:size(zcj,2))={ 'x_1' ; 'x_2' ; 'x_3';'x_4';'x5';'s_1' ;'s_2' ; 's_3'; 'sol' };
BFS=zeros(1,size(A,2));
BFS(BV)=A(:,end);
BFS(end)=sum(BFS.*cost);
Current_BFS=array2table(BFS);
Current_BFS.Properties.VariableNames(1:size(Current_BFS,2))={ 'x_1' ; 'x_2' ;'x_3';'x_4';'x5';'s_1' ; 's_2' ; 's_3';'sol' };
else
RUN=false;
fprintf( '======*********====== \n' )
fprintf( 'The current BFS is optimal \n' )
fprintf( '======*********====== \n' )
disp(TABLE);
end
end
```

Output:

```
The current BFS is not optimal

 The next iteration
Old Basic variable (BV)
     6     7     8

the most negative element in zjcj row is -11 corresponding column;3
Entering variable is 3
Minimum Ratio corresponding to pivot row is 1
Leaving variable is 6
New basic variable (BV)=
     3     7     8
```

x_1	x_2	x_3	x_4	x5	s_1	s_2	s_3	sol
-7	-10	-11	-5	-9	0	0	0	0
6	10	8	6	10	1	0	0	8000
6	3	8	0	0	0	1	0	12000
3	2	2	0	0	0	0	1	10000

x_1	x_2	x_3	x_4	x5	s_1	s_2	s_3	sol
-7	-10	-11	-5	-9	0	0	0	0
6	10	8	6	10	1	0	0	8000
6	3	8	0	0	0	1	0	12000
3	2	2	0	0	0	0	1	10000

```
======****======
The current BFS is optimal
======****======
```

x_1	x_2	x_3	x_4	x5	s_1	s_2	s_3	sol
1.25	3.75	0	3.25	4.75	1.375	0	0	11000
0.75	1.25	1	0.75	1.25	0.125	0	0	1000
0	-7	0	-6	-10	-1	1	0	4000
1.5	-0.5	0	-1.5	-2.5	-0.25	0	1	8000

5)
Manufacturing Problems(Big-M Method)

Q.

A person requires 10, 12, and 12 units chemicals A, B and C respectively for his garden. A liquid product contains 5, 2 and 1 units of A,B and C respectively per jar. A dry product contains 1,2 and 4 units of A,B and C per carton.
If the liquid product sells for Rs.3 per jar and the dry product sells for Rs.2 per carton, how many of each should be purchased, in order to minimize the cost and meet the requirements?

LPP Formulation:

Objective function:

Min Z = 3x1 + 2x2

Subject to Constraints:

5x1 + x2 + s1-a1 = 10
2x1 + 2x2 + s2-a2 = 12
x1 + 4x2 +s3-a3 = 12

x1, x2 ≥ 0

MATLAB CODE:

```
Variables = {'x_1','x_2','s_1','s_2','s_3','A_1','A_2','A_3','Sol'};
M = 1000;
Cost = [-3 -2 0 0 0 -M -M -M 0];
A = [5 1 -1 0 0 1 0 0 10;
2 2 0 -1 0 0 1 0 12;
1 4 0 0 -1 0 0 1 12];
s = eye(size(A,1));
%FINDING STARTING BFS
BV = [];
for j=1:size(s,2)
for i=1:size(A,2)
if A(:,i)==s(:,j)
BV = [BV i];
end
end
end

%Compute Value of Table
B = A(:,BV);
A = inv(B)*A;
ZjCj = Cost(BV)*A-Cost;
%For Print Table
ZCj = [ZjCj;A];
SimpTable = array2table(ZCj);
SimpTable.Properties.VariableNames(1:size(ZCj,2)) = Variables

%SIMPLEX METHOD START -
RUN = true;
while RUN
%% FINDING THE ENTERING VARIABLE
ZC = ZjCj(:,1:end-1);
if any(ZC<0);
fprintf(' The Current BFS is NOT Optimal \n ');
[Entval, pvt_col]= min(ZC);
fprintf('Entering Column = %d \n',pvt_col);
%% FINDING THE LEAVING VARIABLE
sol = A(:,end);
Column = A(:,pvt_col);
if all(Column)<=0
fprintf('Solution is UNBOUNDED \n')
else
Har = find(Column > 0);
ratio = inf.*ones(1,length(sol));
ratio(Har)=sol(Har)./Column(Har);
for i=1:size(Column,1)
```

```
if Column(i)>0
ratio(i)=sol(i)./Column(i);
else
ratio(i)=inf;
end
end
[minR,pvt_row] = min(ratio);
fprintf('Leaving Row = %d \n',pvt_row);

%% UPDATE THE BV & TABLE
BV(pvt_row) = pvt_col;
B = A(:,BV);
A = inv(B)*A;
ZjCj = Cost(BV)*A - Cost;

%% For Print Table
ZCj = [ZjCj;A];
TABLE = array2table(ZCj);
TABLE.Properties.VariableNames(1:size(ZCj,2)) = Variables
end
else
RUN = false;
fprintf(' ==== CURRENT BFS IS OPTIMAL ====\n');
end
end
%FINAL OPTIMAL SOLUTION PRINT
FINAL_BFS = zeros(1,size(A,2));
FINAL_BFS(BV) = A(:,end);
FINAL_BFS(end) = sum(FINAL_BFS.*Cost);
OptimalBFS = array2table(FINAL_BFS);
OptimalBFS.Properties.VariableNames(1:size(OptimalBFS,2)) = Variables
```

Code:

```
>> Variables = {'x_1','x_2','s_1','s_2','s_3','A_1','A_2','A_3','Sol'};
M = 1000;
Cost = [-3 -2 0 0 0 -M -M -M 0];
A = [5 1 -1 0 0 1 0 0 10;
2 2 0 -1 0 0 1 0 12;
1 4 0 0 -1 0 0 1 12];
s = eye(size(A,1));
%FINDING STARTING BFS
BV = [];
for j=1:size(s,2)
for i=1:size(A,2)
if A(:,i)==s(:,j)
BV = [BV i];
end
end
end

%Compute Value of Table
B = A(:,BV);
A = inv(B)*A;
ZjCj = Cost(BV)*A-Cost;
%For Print Table
ZCj = [ZjCj;A];
SimpTable = array2table(ZCj);
SimpTable.Properties.VariableNames(1:size(ZCj,2)) = Variables
```

```
%SIMPLEX METHOD START -
RUN = true;
while RUN
%% FINDING THE ENTERING VARIABLE
ZC = ZjCj(:,1:end-1);
if any(ZC<0);
fprintf(' The Current BFS is NOT Optimal \n ');
[Entval, pvt_col]= min(ZC);
fprintf('Entering Column = %d \n',pvt_col);
%% FINDING THE LEAVING VARIABLE
sol = A(:,end);
Column = A(:,pvt_col);
if all(Column)<=0
fprintf('Solution is UNBOUNDED \n')
else
Har = find(Column > 0);
ratio = inf.*ones(1,length(sol));
ratio(Har)=sol(Har)./Column(Har);
for i=1:size(Column,1)
if Column(i)>0
ratio(i)=sol(i)./Column(i);
else
ratio(i)=inf;
end
end
[minR,pvt_row] = min(ratio);
fprintf('Leaving Row = %d \n',pvt_row);
```

```
%% UPDATE THE BV & TABLE
BV(pvt_row) = pvt_col;
B = A(:,BV);
A = inv(B)*A;
ZjCj = Cost(BV)*A - Cost;

%% For Print Table
ZCj = [ZjCj;A];
TABLE = array2table(ZCj);
TABLE.Properties.VariableNames(1:size(ZCj,2)) = Variables
end
else
RUN = false;
fprintf(' ==== CURRENT BFS IS OPTIMAL ====\n');
end
end
%FINAL OPTIMAL SOLUTION PRINT
FINAL_BFS = zeros(1,size(A,2));
FINAL_BFS(BV) = A(:,end);
FINAL_BFS(end) = sum(FINAL_BFS.*Cost);
OptimalBFS = array2table(FINAL_BFS);
OptimalBFS.Properties.VariableNames(1:size(OptimalBFS,2)) = Variables
```

Output:

SimpTable =

4×9 table

x_1	x_2	s_1	s_2	s_3	A_1	A_2	A_3	Sol
-7997	-6998	1000	1000	1000	0	0	0	-34000
5	1	-1	0	0	1	0	0	10
2	2	0	-1	0	0	1	0	12
1	4	0	0	-1	0	0	1	12

The Current BFS is NOT Optimal
Entering Column = 1
Leaving Row = 1

TABLE =

4×9 table

x_1	x_2	s_1	s_2	s_3	A_1	A_2	A_3	Sol
0	-5398.6	-599.4	1000	1000	1599.4	0	0	-18006
1	0.2	-0.2	0	0	0.2	0	0	2
0	1.6	0.4	-1	0	-0.4	1	0	8
0	3.8	0.2	0	-1	-0.2	0	1	10

The Current BFS is NOT Optimal
Entering Column = 2
Leaving Row = 3

TABLE =

4×9 table

x_1	x_2	s_1	s_2	s_3	A_1	A_2	A_3	Sol
0	-2.5224e-13	-315.26	1000	-420.68	1315.3	0	1420.7	-3799.2
1	3.1554e-17	-0.21053	0	0.052632	0.21053	0	-0.052632	1.4737
0	2.5243e-16	0.31579	-1	0.42105	-0.31579	1	-0.42105	3.7895
0	1	0.052632	0	-0.26316	-0.052632	0	0.26316	2.6316

The Current BFS is NOT Optimal
Entering Column = 5
Leaving Row = 2

TABLE =

4×9 table

x_1	x_2	s_1	s_2	s_3	A_1	A_2	A_3	Sol
0	0	0.25	0.875	0	999.75	999.12	1000	-13
1	0	-0.25	0.125	0	0.25	-0.125	0	1
0	-1.3066e-31	0.75	-2.375	1	-0.75	2.375	-1	9
0	1	0.25	-0.625	0	-0.25	0.625	0	5

fx ==== CURRENT BFS IS OPTIMAL ====

```
OptimalBFS =

  1×9 table

    x_1    x_2    s_1    s_2    s_3    A_1    A_2    A_3    Sol
    ___    ___    ___    ___    ___    ___    ___    ___    ___

     1      5      0      0      9      0      0      0     -13
```

6) Transportation Problems (NWCM, MMM, VAM):

A manufacturer has distribution centres located at Agra, Allahabad and Kolkata. These centres have available 40, 20 and 40 units of his product. His retail outlets at A, B, C, D and E requires 25, 10, 20, 30 and 15 units of the product, respectively. The shipping cost per unit (in rupees) between each centre and outlet is given in the following table.

Distribution Centre	*Retail Outlets*				
	A	*B*	*C*	*D*	*E*
Agra	55	30	40	50	40
Allahabad	35	30	100	45	60
Kolkata	40	60	95	35	30

Find the Initial basic feasible solution by NWCM, VAM.

MATLAB CODE:

#NWCM(Northwest corner method):

```
%21BCE7727
%M Gyanada Chowdary
%Transportation problem of Northwest corner
%initialisation
clc
clear all
cost = [55 30 40 50 40;35 30 100 45 60;40 60 95 35 30];
A = [40 20 40];
B = [25 10 20 30 15];
```

```
if sum(A) == sum(B)
fprintf('Given transportation problem is balance\n')
else
fprintf('Given transportation problem is unbalanced\n')
end
if sum(A)<sum(B)
cost(end+1, :) = sum(B)-sum(A);
A(end+1)=sum(B)-sum(A);
elseif sum(B)<sum(A);
cost(:,end+1)=sum(A)-sum(B);
B(end+1)=sum(A)-sum(B);
end
X = zeros(size(cost));
[m,n] = size(cost);
for i=1:m
for j=1:n
x11 = min(A(i),B(j));
X(i,j) = x11;
A(i) = A(i)-x11;
B(j) = B(j)-x11;
end
end
fprintf('Initial BFS=\n');
IB = array2table(X);
disp(IB);
TotalBFS = length(nonzeros(X));
InitialCost = sum(sum(cost.*X));
fprintf('Initial BFS cost=%d\n',InitialCost)
```

```
>> %21BCE7727
%M Gyanada Chowdary
%Transportation problem of Northwest corner
%initialisation
clc
clear all
cost = [55 30 40 50 40;35 30 100 45 60;40 60 95 35 30];
A = [40 20 40];
B = [25 10 20 30 15];
if sum(A) == sum(B)
fprintf('Given transportation problem is balance\n')
else
fprintf('Given transportation problem is unbalanced\n')
end
if sum(A)<sum(B)
cost(end+1, :) = sum(B)-sum(A);
A(end+1)=sum(B)-sum(A);
elseif sum(B)<sum(A);
cost(:,end+1)=sum(A)-sum(B);
B(end+1)=sum(A)-sum(B);
end
```

```
X = zeros(size(cost));
[m,n] = size(cost);
for i=1:m
for j=1:n
x11 = min(A(i),B(j));
X(i,j) = x11;
A(i) = A(i)-x11;
B(j) = B(j)-x11;
end
end
fprintf('Initial BFS=\n');
IB = array2table(X);
disp(IB);
TotalBFS = length(nonzeros(X));
InitialCost = sum(sum(cost.*X));
fprintf('Initial BFS cost=%d\n',InitialCost)
```

Output:

```
Given transportation problem is balance
Initial BFS=
    X1    X2    X3    X4    X5
    __    __    __    __    __

    25    10     5     0     0
     0     0    15     5     0
     0     0     0    25    15

Initial BFS cost=4925
```

#VAM(Vogel's Approximation method):

MATLAB CODE:

```
%21BCE7727
%M Gyanada Chowdary
%VAM
clc
clear
Cost=[ 55 30 40 50 40;35 30 100 45 60;40 60 95 35 30];
IB1=array2table(Cost);
disp(IB1); A=[40 20 40];
B=[25 10 20 30 15];
if sum(A)==sum(B)
fprintf("It is balanced\n");
else
fprintf("It is not balanced\n");
if sum(A)<sum(B)
Cost(end+1,:)=zeros(1,size(A,2));
```

```
A(end+1)=sum(B)-sum(A);
elseif sum(B)<sum(A)
Cost(:,end+1)=zeros(1,size(A,2));
B(end+1)=sum(A)-sum(B);
end
end
ICost = Cost;
X = zeros(size(Cost));
[m,n] = size(Cost);
BFS = m+n-1;
for i=1:m*n
Col = sort(Cost,1);
Row = sort(Cost,2);
pRow = Row(:,2)-Row(:,1);
pCol = Col(2,:)-Col(1,:);
R = max(pRow);
C = max(pCol);
Rmax = find(pRow == max(R,C));
Cmax = find(pCol == max(R,C));
Cr = Cost(Rmax,:);
Cc = Cost(:,Cmax);
if max(pRow) ~= max(pCol)
if max(pRow)>max(pCol)
[rowind,colind] = find(min(min(Cr))==Cost(Rmax,:));
row1 = Rmax(rowind);
col1 = colind;
else
[rowind,colind]=find(min(min(Cc))==Cost(:,Cmax));
row1 = rowind;
col1 = Cmax(colind);
end
x11 = min(A(row1),B(col1));
[val,ind] = max(x11);
ii = row1(ind);
jj = col1(ind);
else
[rowind1,colind1] = find(min(min(Cr))==Cost(Rmax,:));
row1 = Rmax(rowind1);
col1 = colind1;
C1 = Cost(row1,col1);
[rowind2,colind2]=find(min(min(Cc))==Cost(:,Cmax));
row2 = rowind2;
col2 = Cmax(colind2);
C2 = Cost(row2,col2);
if C1<C2
x11 = min(A(row1),B(col1));
[val,ind]=max(x11);
ii = row1(ind);
jj = col1(ind);
```

```
else
x11 = min(A(row2),B(col2));
[val,ind]=max(x11);
ii = row2(ind);
jj = col2(ind);
end
end
y11 = min(A(ii),B(jj));
X(ii,jj) = y11;
A(ii) = A(ii)-y11;
B(jj) = B(jj)-y11;
Cost(ii,jj) = Inf;
end
fprintf('BFS Table: \n' );
IB=array2table(X);
disp(IB);
InitialCost = sum(sum(ICost.*X));
fprintf('Initial BFS Cost= %d\n',InitialCost);
```

```
Initial BFS Cost= 5050
>> %21BCE7727
%M Gyanada Chowdary
%VAM
clc
clear
Cost=[ 55 30 40 50 40;35 30 100 45 60;40 60 95 35 30];
IB1=array2table(Cost);
disp(IB1); A=[40 20 40];
B=[25 10 20 30 15];
if sum(A)==sum(B)
fprintf("It is balanced\n");
else
fprintf("It is not balanced\n");
if sum(A)<sum(B)
Cost(end+1,:)=zeros(1,size(A,2));
A(end+1)=sum(B)-sum(A);
elseif sum(B)<sum(A)
Cost(:,end+1)=zeros(1,size(A,2));
B(end+1)=sum(A)-sum(B);
end
end
```

```
ICost = Cost;
X = zeros(size(Cost));
[m,n] = size(Cost);
BFS = m+n-1;
for i=1:m*n
Col = sort(Cost,1);
Row = sort(Cost,2);
pRow = Row(:,2)-Row(:,1);
pCol = Col(2,:)-Col(1,:);
R = max(pRow);
C = max(pCol);
Rmax = find(pRow == max(R,C));
Cmax = find(pCol == max(R,C));
Cr = Cost(Rmax,:);
Cc = Cost(:,Cmax);
if max(pRow) ~= max(pCol)
if max(pRow)>max(pCol)
[rowind,colind] = find(min(min(Cr))==Cost(Rmax,:));
row1 = Rmax(rowind);
col1 = colind;
else
[rowind,colind]=find(min(min(Cc))==Cost(:,Cmax));
row1 = rowind;
col1 = Cmax(colind);
end
x11 = min(A(row1),B(col1));
[val,ind] = max(x11);
ii = row1(ind);
jj = col1(ind);
```

```
else
[rowind1,colind1] = find(min(min(Cr))==Cost(Rmax,:));
row1 = Rmax(rowind1);
col1 = colind1;
C1 = Cost(row1,col1);
[rowind2,colind2]=find(min(min(Cc))==Cost(:,Cmax));
row2 = rowind2;
col2 = Cmax(colind2);
C2 = Cost(row2,col2);
if C1<C2
x11 = min(A(row1),B(col1));
[val,ind]=max(x11);
ii = row1(ind);
jj = col1(ind);
else
x11 = min(A(row2),B(col2));
[val,ind]=max(x11);
ii = row2(ind);
jj = col2(ind);
end
end
y11 = min(A(ii),B(jj));
X(ii,jj) = y11;
A(ii) = A(ii)-y11;
B(jj) = B(jj)-y11;
Cost(ii,jj) = Inf;
end
fprintf('BFS Table: \n' );
IB=array2table(X);
disp(IB);
InitialCost = sum(sum(ICost.*X));
fprintf('Initial BFS Cost= %d\n',InitialCost);
```

Output:

```
    Cost1    Cost2    Cost3    Cost4    Cost5
    _____    _____    _____    _____    _____

     55       30        40      50       40
     35       30       100      45       60
     40       60        95      35       30

It is balanced
BFS Table:
    X1    X2    X3    X4    X5
    __    __    __    __    __

     5    10    20     5     0
    20     0     0     0     0
     0     0     0    25    15

Initial BFS Cost= 3650
```

```
    Cost1      Cost2      Cost3      Cost4      Cost5
    _____      _____      _____      _____      _____

     55         30          40         50         40
     35         30         100         45         60
     40         60          95         35         30

It is balanced
BFS Table:
    X1     X2     X3     X4     X5
    __     __     __     __     __

     5     10     20      5      0
    20      0      0      0      0
     0      0      0     25     15

Initial BFS Cost= 3650
```

7) Assignment Problems (Hungerian Method):

Q.

The marketing director of a multi-unit company is faced with a problem of assigning 5 senior managers to six zones. From past experience he knows that the efficiency percentage judged by sales, operating costs, etc., depends on the manager-zone combination. The efficiency of different managers is given below:

		Zones					
		I	*II*	*III*	*IV*	*V*	*VI*
Manager	*A*	73	91	87	82	78	80
	B	81	85	69	76	74	85
	C	75	72	83	84	78	91
	D	93	96	86	91	83	82
	E	90	91	79	89	69	76

Find out which zone should be managed by a junior manager due to the non-availability of a senior manager.

Code:

```
clc;
clear all;
arr=[73,91,87,82,78,80;
    81,85,69,76,74,85;
    75,72,83,84,78,91;
    93,96,86,91,83,82;
    90,91,79,89,69,76;
    0,0,0,0,0,0];
disp('cost matrix');
disp(arr);
b=arr;
for i=1:size(arr,1)
    sub=min(arr(i,:));
    arr(i,:) = arr(i,:)-sub;
end
for i=1:size(arr,2)
    sub=min(arr(:,i));
    arr(:,i) = arr(:,i)-sub;
end
disp('after subtracting row minimum and column minimum');
disp(arr);
while true
    temp=arr;
    lines = 0;
    while true
        minZ=inf;
        for i=1:size(temp,1)
            count=size(find(temp(i,:)==0),2);
            disp('count in row is:');
            disp(count);
```

```
    if(count>0 && count < minZ)
        minZ=count;
        d=1;
        y=find(temp(i,:)==0,1);
        disp('y1 is:');
        disp(y);
    end
end
for i=1:size(temp,2)
    count=size(find(temp(:,i)==0),1);
    disp('count in col is:');
    disp(count);
    if(count>0 && count < minZ)
        minZ=count;
        d=0;
        y=find(temp(:,i)==0,1);
        disp('y2s is:');
        disp(y);
    end
    disp('y is:');
        disp(y);
end
if minZ==inf
    break;
end
if d==1
    temp(:,y)=inf;
else
    temp(y,:)=inf;
end
lines = lines + 1;
disp('lines is:');
disp(lines);
```

```
    end
    sub = min(min(temp));
    if(lines~=6)
        for i=1:size(arr,1)
            for j=1:size(arr,2)
                if(temp(i,j)~=inf)
                    arr(i,j) = arr(i,j)-sub;
                elseif((size(find(temp(i,:)==inf),2)==6) && (size(find(temp(:,j)==inf),1)==6))
                    arr(i,j) = arr(i,j)+sub;
                end
            end
        end
    end
    if(lines==6)
        break;
    end
end
disp('Modified cost matrix');
disp(arr);
totalc=0;

for i=1:size(arr,1)
    for j=1:size(arr,2)
        if(arr(i,j)==0)
            totalc=totalc+b(i,j);
            for k=j+1:size(arr,2)
                if(arr(i,k)==0)
                    arr(i,k)=inf;
                end
            end
            for k=i+1:size(arr,1)
                if(arr(k,j)==0)
                    arr(k,j)=inf;
```

```
            end
          end
        end
      end
    end

disp('Total Cost');
disp(totalc);
```

```
>> %21BCE7727
 %M Gyanada Chowdary
clc;
clear all;
arr=[73,91,87,82,78,80;
    81,85,69,76,74,85;
    75,72,83,84,78,91;
    93,96,86,91,83,82;
    90,91,79,89,69,76;
    0,0,0,0,0,0];
disp('cost matrix');
disp(arr);
b=arr;
for i=1:size(arr,1)
    sub=min(arr(i,:));
    arr(i,:) = arr(i,:)-sub;
end
for i=1:size(arr,2)
    sub=min(arr(:,i));
    arr(:,i) = arr(:,i)-sub;
end
```

```
disp('after subtracting row minimum and column minimum');
disp(arr);
while true
    temp=arr;
    lines = 0;
    while true
        minZ=inf;
        for i=1:size(temp,1)
            count=size(find(temp(i,:)==0),2);
            disp('count in row is:');
            disp(count);
            if(count>0 && count < minZ)
                minZ=count;
                d=1;
                y=find(temp(i,:)==0,1);
                disp('y1 is:');
                disp(y);
            end
        end
        for i=1:size(temp,2)
            count=size(find(temp(:,i)==0),1);
            disp('count in col is:');
            disp(count);
            if(count>0 && count < minZ)
                minZ=count;
                d=0;
                y=find(temp(:,i)==0,1);
                disp('y2s is:');
                disp(y);
            end
```

```
                disp('y is:');
                    disp(y);
        end
        if minZ==inf
            break;
        end
        if d==1
            temp(:,y)=inf;
        else
            temp(y,:)=inf;
        end
        lines = lines + 1;
        disp('lines is:');
        disp(lines);
    end
    sub = min(min(temp));
    if(lines~=6)
        for i=1:size(arr,1)
            for j=1:size(arr,2)
                if(temp(i,j)~=inf)
                    arr(i,j) = arr(i,j)-sub;
                elseif((size(find(temp(i,:)==inf),2)==6) && (size(find(temp(:,j)==inf),1)==6))
                    arr(i,j) = arr(i,j)+sub;
                end
            end
        end
    end
    if(lines==6)
        break;
    end
end ...
```

```
disp('Modified cost matrix');
disp(arr);
totalc=0;

for i=1:size(arr,1)
    for j=1:size(arr,2)
        if(arr(i,j)==0)
            totalc=totalc+b(i,j);
            for k=j+1:size(arr,2)
                if(arr(i,k)==0)
                    arr(i,k)=inf;
                end
            end
            for k=i+1:size(arr,1)
                if(arr(k,j)==0)
                    arr(k,j)=inf;
                end
            end
        end
    end
end

disp('Total Cost');
disp(totalc);
```

Output:

```
cost matrix
    73    91    87    82    78    80
    81    85    69    76    74    85
    75    72    83    84    78    91
    93    96    86    91    83    82
    90    91    79    89    69    76
     0     0     0     0     0     0

after subtracting row minimum and column minimum
     0    18    14     9     5     7
    12    16     0     7     5    16
     3     0    11    12     6    19
    11    14     4     9     1     0
    21    22    10    20     0     7
     0     0     0     0     0     0
```

```
count in row is:
     1

y1 is:
     1

count in row is:
     1

count in row is:
     1

count in row is:
     1

count in row is:
     1

count in row is:
     6

count in col is:
     2

y is:
     1

count in col is:
     2
```

```
y is:
     1

count in col is:
     2

y is:
     1

count in col is:
     2

y is:
     1

count in col is:
     1

y is:
     1

count in col is:
     2

y is:
     1
```

```
y is:
     1

count in col is:
     2

y is:
     1

lines is:
     1

count in row is:
     0

count in row is:
     1

y1 is:
     3

count in row is:
     1

count in row is:
     1

count in row is:
     1
```

```
count in row is:
     5

count in col is:
     0

y is:
     3

count in col is:
     2

y is:
     3

count in col is:
     2

y is:
     3

count in col is:
     1

y is:
     3

count in col is:
     2
```

```
y is:
     3

count in col is:
     2

y is:
     3

lines is:
     2

count in row is:
     0

count in row is:
     0

count in row is:
     1

y1 is:
     2

count in row is:
     1

count in row is:
     1
```

```
y1 is:
     2

count in row is:
     1

count in row is:
     1

count in row is:
     4

count in col is:
     0

y is:
     2

count in col is:
     2

y is:
     2

count in col is:
     0

y is:
     2
```

```
y is:
     2

count in col is:
     1

y is:
     2

count in col is:
     2

y is:
     2

count in col is:
     2

y is:
     2

lines is:
     3

count in row is:
     0

count in row is:
     0

count in row is:
```

```
count in row is:
     1

y1 is:
     6

count in row is:
     1

count in row is:
     3

count in col is:
     0

y is:
     6

count in col is:
     0

y is:
     6

count in col is:
     0

y is:
     6
```

```
count in col is:
     1

y is:
     6

count in col is:
     2

y is:
     6

count in col is:
     2

y is:
     6

lines is:
     4

count in row is:
     0

count in row is:
     0

count in row is:
     0
```

```
count in row is:
     1

y1 is:
     5

count in row is:
     2

count in col is:
     0

y is:
     5

count in col is:
     0

y is:
     5

count in col is:
     0

y is:
     5

count in col is:
     1
```

```
y is:
     5

count in col is:
     2

y is:
     5

count in col is:
     0

y is:
     5

lines is:
     5

count in row is:
     0

count in row is:
     0

count in row is:
     0

count in row is:
     0
```

```
count in row is:
     0

count in row is:
     1

y1 is:
     4

count in col is:
     0

y is:
     4

count in col is:
     0

y is:
     4

count in col is:
     0

y is:
     4

count in col is:
     1
```

```
y is:
     4

count in col is:
     0

y is:
     4

count in col is:
     0

y is:
     4

lines is:
     6

count in row is:
     0

count in row is:
     0

count in row is:
     0

count in row is:
     0
```

```
count in row is:
     0

count in row is:
     0

count in col is:
     0

y is:
     4

count in col is:
     0

y is:
     4

count in col is:
     0

y is:
     4

count in col is:
     0

y is:
     4
```

```
count in col is:
     0

y is:
     4

count in col is:
     0

y is:
     4

Modified cost matrix
     0    18    14     9     5     7
    12    16     0     7     5    16
     3     0    11    12     6    19
    11    14     4     9     1     0
    21    22    10    20     0     7
     0     0     0     0     0     0

Total Cost
   365
```

Q.

Use three iterations of Golden section search method and find the interval of uncertainty in which the volume $V(x) = \left(\frac{\pi}{3}\right)(9 - x^2)(x + 3), -3 \leq x \leq 3$ of the cone maximizes. **(10 Marks)**

Q.

Use three iterations of Fibonacci search method and find the uncertainty of the intervals in which the area $A(x) = x(8 - 2x), 0 \leq x \leq 5$; of the rectangular box maximizes. **(10 Marks)**

Q.

The volume of the solid is given by $V(x) = \exp(2x) - x^3$. Use four iteration of Newton's method and find the minimum volume with initial guess $x_1 = 1, k = 1$. **(12M)**

8) Nonlinear Programming Problem(Golden Search Method)

Golden Search Method:

MATLAB CODE:

```
%21BCE7727
%M Gyanada chowdary
V = @(x) (9-x.^2).*(x+3);
a = -3;
b = 3;
phi = (1 + sqrt(5))/2;

c = b - (b-a)/phi;
d = a + (b-a)/phi;
if V(c) < V(d)
```

```
    a = c;
else
    b = d;
end
fprintf('Iteration 1: [%f, %f]\n', a, b);

c = b - (b-a)/phi;
d = a + (b-a)/phi;
if V(c) < V(d)
    a = c;
else
    b = d;
end
fprintf('Iteration 2: [%f, %f]\n', a, b);

c = b - (b-a)/phi;
d = a + (b-a)/phi;
if V(c) < V(d)
    a = c;
else
    b = d;
end
fprintf('Iteration 3: [%f, %f]\n', a, b);
```

```
%21BCE7727
%M Gyanada chowdary
V = @(x) (9-x.^2).*(x+3);
a = -3;
b = 3;
phi = (1 + sqrt(5))/2;

c = b - (b-a)/phi;
d = a + (b-a)/phi;
if V(c) < V(d)
    a = c;
else
    b = d;
end
fprintf('Iteration 1: [%f, %f]\n', a, b);

c = b - (b-a)/phi;
d = a + (b-a)/phi;
if V(c) < V(d)
    a = c;
else
    b = d;
end
fprintf('Iteration 2: [%f, %f]\n', a, b);

c = b - (b-a)/phi;
d = a + (b-a)/phi;
if V(c) < V(d)
    a = c;
else
    b = d;
end
fprintf('Iteration 3: [%f, %f]\n', a, b);
```

Output:

```
Iteration 1: [-0.708204, 3.000000]
Iteration 2: [-0.708204, 1.583592]
Iteration 3: [0.167184, 1.583592]
fx >>
```

Fibanocci Search Method:

MATLAB CODE:

```
A = @(x) x.*(8-2*x);
a = 0;
b = 5;
tol = 1e-6;
fib = [1 1];
while (b-a)/fib(end) > tol
    fib = [fib fib(end)+fib(end-1)];
end
for i = length(fib)-2:-1:length(fib)-4
    x1 = a + (b-a)*fib(i)/fib(i+2);
    x2 = a + (b-a)*fib(i+1)/fib(i+2);
    if A(x1) > A(x2)
        b = x2;
    else
        a = x1;
    end
    x = (a + b)/2;
    max_area = A(x);
    fprintf('The maximum area is %f at x = %f\n', max_area, x);
end
```

```
A = @(x) x.*(8-2*x);
a = 0;
b = 5;
tol = 1e-6;
fib = [1 1];
while (b-a)/fib(end) > tol
    fib = [fib fib(end)+fib(end-1)];
end
for i = length(fib)-2:-1:length(fib)-4
    x1 = a + (b-a)*fib(i)/fib(i+2);
    x2 = a + (b-a)*fib(i+1)/fib(i+2);
    if A(x1) > A(x2)
        b = x2;
    else
        a = x1;
    end
    x = (a + b)/2;
    max_area = A(x);
    fprintf('The maximum area is %f at x = %f\n', max_area, x);
end
```

Output:

```
The maximum area is 7.586105 at x = 1.545085
The maximum area is 7.963412 at x = 2.135255
The maximum area is 7.894669 at x = 1.770510
```

Newton's Method:

MATLAB CODE:

```
%21BCE7727
%M Gyanada Chowdary
V = @(x) exp(2*x) - x^3;
dV = @(x) 2*exp(2*x) - 3*x^2;
x0 = 1;
for i = 1:4
   x = x0 - V(x0)/dV(x0);
   fprintf('Iteration %d: x = %f, V(x) = %f\n', i, x, V(x));
   x0 = x;
end
fprintf('The minimum volume is %f.\n', exp(2*x0));
```

```
%21BCE7727
%M Gyanada Chowdary
V = @(x) exp(2*x) - x^3;
dV = @(x) 2*exp(2*x) - 3*x^2;
x0 = 1;
for i = 1:4
    x = x0 - V(x0)/dV(x0);
    fprintf('Iteration %d: x = %f, V(x) = %f\n', i, x, V(x));
    x0 = x;
end
fprintf('The minimum volume is %f.\n', exp(2*x0));
```

Output:

```
Iteration 1: x = 0.457548, V(x) = 2.401229
Iteration 2: x = -0.092438, V(x) = 0.831998
Iteration 3: x = -0.600751, V(x) = 0.517554
Iteration 4: x = 0.474754, V(x) = 2.477430
The minimum volume is 2.584435.
```

9) Other Problems(LAB9)

Q1)

MATLAB CODE:

```
%21BCE7727

%M Gyanada Chowdary

tic

Var = {'x1', 'x2', 's1', 's2', 'Sol'};

C = [6 3 0 0 0];

A = [-3 -5 ; -4 -3];

b = [-50;-60]

m = eye(size(A, 1))

W = [A m b];

BV = [];

for j = 1:size(m, 2)

for i = 1:size(W, 2)

if W(:, i) == m(:, j)
```

```
BV = [BV i];
end
end
end
disp(BV);
Q = W(:, BV);
W = inv(Q) * W;
ZjCj = C(BV) * W - C;
Zcj = [ZjCj; W]
Dual = array2table(Zcj);
Dual.Properties.VariableNames(1:size(Zcj, 2)) = Var
RUN = true;
while RUN
SOL = W(:, end);
if any(SOL < 0);
fprintf('The current BFS is not FEASIBLE \n')
[LeaVal, pvt_row] = min(SOL);
fprintf('Leaving Row = %d \n', pvt_row);
ROW = W(pvt_row, 1:end - 1);
ZJ = ZjCj(:, 1:end - 1);
for i = 1:size(ROW, 2)
if ROW(i) < 0
ratio(i) = abs(ZJ(i) ./ ROW(i));
else
ratio(i) = inf;
end
end
[minVAL, pvt_col] = min(ratio);
fprintf('Entering Variable = %d \n', pvt_col);
%BV
fprintf('Basic Variables (BV) =')
BV(pvt_row) = pvt_col;
disp(Var(BV));
```

pvt_key = W(pvt_row, pvt_col);

W(pvt_row, :) = W(pvt_row, :) ./ pvt_key;

for i = 1:size(W, 1)

if i ~= pvt_row

W(i, :) = W(i, :) - W(i, pvt_col) .* W(pvt_row, :);

end

end

ZjCj = C(BV) * W - C;

Zcj = [ZjCj; W]

Dual = array2table(Zcj);

Dual.Properties.VariableNames(1:size(Zcj, 2)) = Var

else

RUN = false;

fprintf('The current BFS is FEASIBLE and OPTIMAL\n')

end

end

```
>> %21BCE7727
%M Gyanada Chowdary
tic
Var = {'x1', 'x2', 's1', 's2', 'Sol'};
C = [6 3 0 0 0];
A = [-3 -5 ; -4 -3];
b = [-50;-60]
m = eye(size(A, 1))
W = [A m b];
BV = [];
for j = 1:size(m, 2)
for i = 1:size(W, 2)
if W(:, i) == m(:, j)
BV = [BV i];
end
end
end
disp(BV);
Q = W(:, BV);
W = inv(Q) * W;
ZjCj = C(BV) * W - C;
Zcj = [ZjCj; W]
Dual = array2table(Zcj);
Dual.Properties.VariableNames(1:size(Zcj, 2)) = Var
RUN = true;
while RUN
SOL = W(:, end);
if any(SOL < 0);
fprintf('The current BFS is not FEASIBLE \n')
```

```
[LeaVal, pvt_row] = min(SOL);
fprintf('Leaving Row = %d \n', pvt_row);
ROW = W(pvt_row, 1:end - 1);
ZJ = ZjCj(:, 1:end - 1);
for i = 1:size(ROW, 2)
if ROW(i) < 0
ratio(i) = abs(ZJ(i) ./ ROW(i));
else
ratio(i) = inf;
end
end
[minVAL, pvt_col] = min(ratio);
fprintf('Entering Variable = %d \n', pvt_col);
%BV
fprintf('Basic Variables (BV) =')
BV(pvt_row) = pvt_col;
disp(Var(BV));
pvt_key = W(pvt_row, pvt_col);
W(pvt_row, :) = W(pvt_row, :) ./ pvt_key;
for i = 1:size(W, 1)
if i ~= pvt_row
W(i, :) = W(i, :) - W(i, pvt_col) .* W(pvt_row, :);
end
end
ZjCj = C(BV) * W - C;
Zcj = [ZjCj; W]
Dual = array2table(Zcj);
Dual.Properties.VariableNames(1:size(Zcj, 2)) = Var
else
RUN = false;
fprintf('The current BFS is FEASIBLE and OPTIMAL\n')
end
```

```
end
F_BFS = zeros(1, size(W, 2));
F_BFS(BV) = W(:, end);
F_BFS(end) = sum(F_BFS .* C);
OptimalBFS = array2table(F_BFS);
OptimalBFS.Properties.VariableNames(1:size(OptimalBFS, 2)) = Var
Elapsed_time = toc
```

Output:

```
b =

   -50
   -60


m =

     1     0
     0     1

     3     4


Zcj =

    -6    -3     0     0     0
    -3    -5     1     0   -50
    -4    -3     0     1   -60
```

```
Dual =

  3×5 table
```

x1	x2	s1	s2	Sol
-6	-3	0	0	0
-3	-5	1	0	-50
-4	-3	0	1	-60

```
The current BFS is not FEASIBLE
Leaving Row = 2
Entering Variable = 2
Basic Variables (BV) =    {'s1'}    {'x2'}


Zcj =

   -2.0000         0         0   -1.0000   60.0000
    3.6667         0    1.0000   -1.6667   50.0000
    1.3333    1.0000         0   -0.3333   20.0000
```

```
Dual =

  3×5 table
```

x1	x2	s1	s2	Sol
-2	0	0	-1	60
3.6667	0	1	-1.6667	50
1.3333	1	0	-0.33333	20

```
The current BFS is FEASIBLE and OPTIMAL
```

```
OptimalBFS =

  1×5 table

    x1    x2    s1    s2    Sol
    __    __    __    __    ___

    0     20    50    0     60


Elapsed_time =

    0.1415
```

Q2)

MATLAB CODE:

```
%21BCE7727
%M Gyanada Chowdary
Variables = {'x','y','z','s_1','s_2','A_1','A_2','Sol'};
M = 5000;
Cost = [2 3 4 0 0 -M -M 0];
A = [3 1 4 1 0 0 0 600;
2 4 2 0 -1 1 0 480;
2 3 3 0 0 0 1 540];
s = eye(size(A,1));
BV = [];
for j=1:size(s,2)
for i=1:size(A,2)
if A(:,i)==s(:,j)
BV = [BV i];
end
end
end
B = A(:,BV);
A = inv(B)*A;
ZjCj = Cost(BV)*A-Cost;
ZCj = [ZjCj;A];
SimpTable = array2table(ZCj);
SimpTable.Properties.VariableNames(1:size(ZCj,2)) = Variables
```

```
RUN = true;
while RUN
%% FINDING THE ENTERING VARIABLE
ZC = ZjCj(:,1:end-1);
if any(ZC<0);
fprintf(' The Current BFS is NOT Optimal \n ');
[Entval, pvt_col]= min(ZC);
fprintf('Entering Column = %d \n',pvt_col);
%% FINDING THE LEAVING VARIABLE
sol = A(:,end);
Column = A(:,pvt_col);
if all(Column)<=0
fprintf('Solution is UNBOUNDED \n')
else
Har = find(Column > 0);
ratio = inf.*ones(1,length(sol));
ratio(Har)=sol(Har)./Column(Har);
for i=1:size(Column,1)
if Column(i)>0
ratio(i)=sol(i)./Column(i);
else
ratio(i)=inf;
end
end
[minR,pvt_row] = min(ratio);
fprintf('Leaving Row = %d \n',pvt_row);
%% UPDATE THE BV & TABLE
BV(pvt_row) = pvt_col;
B = A(:,BV);
A = inv(B)*A;
ZjCj = Cost(BV)*A - Cost;
%% For Print Table
ZCj = [ZjCj;A];
```

```
TABLE = array2table(ZCj);
TABLE.Properties.VariableNames(1:size(ZCj,2)) = Variables
end
else
RUN = false;
fprintf(' ==== CURRENT BFS IS OPTIMAL ====\n');
end
end
FINAL_BFS = zeros(1,size(A,2));
FINAL_BFS(BV) = A(:,end);
FINAL_BFS(end) = sum(FINAL_BFS.*Cost);
OptimalBFS = array2table(FINAL_BFS);
OptimalBFS.Properties.VariableNames(1:size(OptimalBFS,2)) = Variables
```

```
>> %21BCE7727
%M Gyanada Chowdary
Variables = {'x','y','z','s_1','s_2','A_1','A_2','Sol'};
M = 5000;
Cost = [2 3 4 0 0 -M -M 0];
A = [3 1 4 1 0 0 0 600;
2 4 2 0 -1 1 0 480;
2 3 3 0 0 0 1 540];
s = eye(size(A,1));
BV = [];
for j=1:size(s,2)
for i=1:size(A,2)
if A(:,i)==s(:,j)
BV = [BV i];
end
end
end
B = A(:,BV);
A = inv(B)*A;
ZjCj = Cost(BV)*A-Cost;
ZCj = [ZjCj;A];
SimpTable = array2table(ZCj);
SimpTable.Properties.VariableNames(1:size(ZCj,2)) = Variables
RUN = true;
while RUN
%% FINDING THE ENTERING VARIABLE
ZC = ZjCj(:,1:end-1);
if any(ZC<0);
fprintf(' The Current BFS is NOT Optimal \n ');
[Entval, pvt_col]= min(ZC);
fprintf('Entering Column = %d \n',pvt_col);
```

```
%% FINDING THE LEAVING VARIABLE
sol = A(:,end);
Column = A(:,pvt_col);
if all(Column)<=0
fprintf('Solution is UNBOUNDED \n')
else
Har = find(Column > 0);
ratio = inf.*ones(1,length(sol));
ratio(Har)=sol(Har)./Column(Har);
for i=1:size(Column,1)
if Column(i)>0
ratio(i)=sol(i)./Column(i);
else
ratio(i)=inf;
end
end
[minR,pvt_row] = min(ratio);
fprintf('Leaving Row = %d \n',pvt_row);
%% UPDATE THE BV & TABLE
BV(pvt_row) = pvt_col;
B = A(:,BV);
A = inv(B)*A;
ZjCj = Cost(BV)*A - Cost;
%% For Print Table
ZCj = [ZjCj;A];
TABLE = array2table(ZCj);
TABLE.Properties.VariableNames(1:size(ZCj,2)) = Variables
end
else
RUN = false;
fprintf(' ==== CURRENT BFS IS OPTIMAL ====\n');
end
```

```
end
FINAL_BFS = zeros(1,size(A,2));
FINAL_BFS(BV) = A(:,end);
FINAL_BFS(end) = sum(FINAL_BFS.*Cost);
OptimalBFS = array2table(FINAL_BFS);
OptimalBFS.Properties.VariableNames(1:size(OptimalBFS,2)) = Variables
```

```
SimpTable =

  4×8 table

      x         y         z       s_1    s_2    A_1    A_2      Sol
    ______    ______    ______    ___    ____    ___    ___    ________

    -20002    -35003    -25004     0     5000     0      0     -5.1e+06
         3         1         4     1        0     0      0          600
         2         4         2     0       -1     1      0          480
         2         3         3     0        0     0      1          540

 The Current BFS is NOT Optimal
 Entering Column = 2
Leaving Row = 2

TABLE =

  4×8 table

      x       y       z       s_1      s_2        A_1      A_2        Sol
    _______   _    _______    ___    _______    ______    ___    ___________

    -2500.5   0    -7502.5     0     -3750.8    8750.8     0     -8.9964e+05
        2.5   0        3.5     1        0.25     -0.25     0             480
        0.5   1        0.5     0       -0.25      0.25     0             120
        0.5   0        1.5     0        0.75     -0.75     1             180

 The Current BFS is NOT Optimal
 Entering Column = 3
Leaving Row = 3
```

```
TABLE =

  4×8 table

       x       y         z          s_1    s_2      A_1        A_2      Sol
    _______    _    ___________     ___    ____    ______    ________    ___

    0.33333    0              0      0      0.5    4999.5      5001.7    660
     1.3333    0    4.4409e-16       1     -1.5       1.5     -2.3333     60
    0.33333    1    2.7756e-17       0     -0.5       0.5    -0.33333     60
    0.33333    0              1      0      0.5      -0.5     0.66667    120

 ==== CURRENT BFS IS OPTIMAL ====

OptimalBFS =

  1×8 table

    x    y      z     s_1    s_2    A_1    A_2    Sol
    _    __    ___    ___    ___    ___    ___    ___

    0    60    120    60      0      0      0     660
```

-----------------------END---