MATLAB REPORT

Course code:Mat2003

Slot L7

VIT-AP UNIVERSITY

Guided by:-

Prof.Yada NanduKumar

Name : Valiveti Manikanta Bhuvanesh

Rg No :19BCD7088

Batch :L7

Index

S.NO	Topic
1	Simplex Method
2	Big M Method
3	Dual Simplex Method
4	Least Cost Method
5	Hungarian Method
6	Golden Section Search Method
7	Newton Raphson Method
8	Game Theory(Saddel Point)
9	Lagrange’s Multipliers

SIMPLEX METHOD

Code:-

```
%VALIVETI MANIKANTA BHUVANESH
%19BCD7088
%Simplex method
%input the parameters
clc;
clear all;
Noofvariables=3;
C=[5 2 8]; %cost matrix in row
M=[2 -4 1;2 3 -1;6 -1 3]; %coefficient m atrix
b=[42;42;42]; % right hand side matrix

s=eye(size(M,1));
A=[M s b]

[m,n]=size(A);

cost=zeros(1,n); % zero matrix


cost(1:Noofvariables)=C

%return;
%%%Constraint BV (indices of BV)
BV=Noofvariables+1:1:n-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';'s_1';'s_2';'s_3';'sol'}

%%Simplex Table strating

 RUN=true;

 while RUN

if any(zjcj<0) %%check if  any negative value 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 zj-cj 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
%---------------------------------upto here-----------------------
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,:);
   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';'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';'s_1';'s_2';'s_3';'sol'}
else

   RUN=false;
   fprintf('======********====== \n')
   fprintf('The current BFS is optimal \n')
    fprintf('======********====== \n')
 end
end
```

OUTPUT:-

```
A =

    2   -4    1    1    0    0   42
    2    3   -1    0    1    0   42
    6   -1    3    0    0    1   42


cost =

    5    2    8    0    0    0    0


BV =

    4    5    6


zjcj =

   -5   -2   -8    0    0    0    0


SimpleTable =

  4×7 table

    x_1   x_2   x_3   s_1   s_2   s_3   sol
    ___   ___   ___   ___   ___   ___   ___
```

-5	-2	-8	0	0	0	0
2	-4	1	1	0	0	42
2	3	-1	0	1	0	42
6	-1	3	0	0	1	42

The current BFS is not optimal

The next iteration

Old Basic variable (BV)

4 5 6

the most negative element in zj-cj row is -8 corresponding column 3

Entering variable is 3

Minimum Ratio corresponding to pivot row is 3

Leaving variable is 6

New basic variable (BV)=

4 5 3

TABLE =

4×7 table

x_1	x_2	x_3	s_1	s_2	s_3	sol
11	-4.6667	0	0	0	2.6667	112
0	-3.6667	0	1	0	-0.33333	28
4	2.6667	0	0	1	0.33333	56
2	-0.33333	1	0	0	0.33333	14

Current_BFS =

1×7 table

x_1	x_2	x_3	s_1	s_2	s_3	sol
0	0	14	28	56	0	112

The current BFS is not optimal

The next iteration

Old Basic variable (BV)

4 5 3

the most negative element in zj-cj row is -4.666667e+00 corresponding column 2

Entering variable is 2

Minimum Ratio corresponding to pivot row is 2

Leaving variable is 5

New basic variable (BV)=

4 2 3

TABLE =

4×7 table

x_1	x_2	x_3	s_1	s_2	s_3	sol

18	0	0	0	1.75	3.25	210
5.5	0	0	1	1.375	0.125	105
1.5	1	0	0	0.375	0.125	21
2.5	0	1	0	0.125	0.375	21

Current_BFS =

1×7 table

x_1	x_2	x_3	s_1	s_2	s_3	sol
0	21	21	105	0	0	210

======********======

The current BFS is optimal

======********======

Big M Method

Code:-

```
%VALIVETI MANIKANTA BHUVANESH
%19BCD7088
%Big M method
%input the parameters
clc;
clear all;
Noofvariables=3;
j=1000000;
C=[5 10 8]; %cost matrix in row
M=[3 5 2 0;4 4 4 -1;2 4 5 0] %coefficient m atrix
b=[60;72;100]; % right hand side matrix
p=[0 0 -j -j 0];
s=eye(size(M,1));
A=[M s b]

[m,n]=size(A);

cost=[C p] % zero matrix

%return;
%%%Constraint BV (indices of BV)
BV=Noofvariables+1:1:n-2

%
% %% 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';'s_1';'s_2';'a_1';'a_2';'sol'}

%%Simplex Table strating

 RUN=true;

 while RUN
```

```
if any(zjcj<0) %%check if  any negative value 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 zj-cj 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
%-----------------------------------upto here-------------
----------
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,:);
    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';'s_2';'s_1';'a_1';'a_2';'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';'s_1';'s_2';'a_1';'a_2';'sol'}
else

    RUN=false;
    fprintf('======********======= \n')
    fprintf('The current BFS is optimal \n')
     fprintf('======********======= \n')
 end
end
```

Output:-

M =

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

A =

```
    3     5     2     0     1     0     0    60
    4     4     4    -1     0     1     0    72
    2     4     5     0     0     0     1   100


cost =

         5        10         8         0         0   -1000000   -1000000         0


BV =

    4     5     6


SimpleTable =

  4×8 table

    x_1      x_2      x_3     s_1    s_2     a_1     a_2     sol
   ______   ______   ______   ___    ___    _____    ___    ______

   -2e+06   -4e+06   -5e+06    0      0    1e+06     0    -1e+08
       3        5        2    0      1        0     0        60
       4        4        4   -1      0        1     0        72
       2        4        5    0      0        0     1       100
```

The current BFS is not optimal

The next iteration

Old Basic variable (BV)

4 5 6

the most negative element in zj-cj row is -5000008 corresponding column 3

Entering variable is 3

Minimum Ratio corresponding to pivot row is 2

Leaving variable is 5

New basic variable (BV)=

4 3 6

TABLE =

4×8 table

x_1	x_2	x_3	s_2	s_1	a_1	a_2	sol
3e+06	1e+06	0	-1.25e+06	0	2.25e+06	0	-9.9999e+06
1	3	0	0.5	1	-0.5	0	24
1	1	1	-0.25	0	0.25	0	18
-3	-1	0	1.25	0	-1.25	1	10

Current_BFS =

1×8 table

x_1	x_2	x_3	s_1	s_2	a_1	a_2	sol
0	0	18	24	0	10	0	-9.9999e+06

The current BFS is not optimal

The next iteration

Old Basic variable (BV)

4 3 6

the most negative element in zj-cj row is -1250002 corresponding column 4

Entering variable is 4

Minimum Ratio corresponding to pivot row is 3

Leaving variable is 6

New basic variable (BV)=

4 3 4

TABLE =

4×8 table

x_1	x_2	x_3	s_2	s_1	a_1	a_2	sol
-1.8	-3.6	0	0	0	1e+06	1e+06	160
2.2	3.4	0	0	1	0	-0.4	20
0.4	0.8	1	0	0	0	0.2	20
-2.4	-0.8	0	1	0	-1	0.8	8

Current_BFS =

1×8 table

x_1	x_2	x_3	s_1	s_2	a_1	a_2	sol
0	0	20	8	0	0	0	160

The current BFS is not optimal

The next iteration

Old Basic variable (BV)

4 3 4

the most negative element in zj-cj row is -3.600000e+00 corresponding column 2

Entering variable is 2

Minimum Ratio corresponding to pivot row is 1

Leaving variable is 4

New basic variable (BV)=

2 3 4

TABLE =

4×8 table

x_1	x_2	x_3	s_2	s_1	a_1	a_2	sol
0.52941	0	0	0	1.0588	1e+06	1e+06	181.18
0.64706	1	0	0	0.29412	0	-0.11765	5.8824
-0.11765	0	1	0	-0.23529	0	0.29412	15.294
-1.8824	0	0	1	0.23529	-1	0.70588	12.706

Current_BFS =

1×8 table

x_1	x_2	x_3	s_1	s_2	a_1	a_2	sol
0	5.8824	15.294	12.706	0	0	0	181.18

======********======

The current BFS is optimal

======********======

Dual Simplex Method

Code:-

```
%VALIVETI MANIKANTA BHUVANESH
%19BCD7088
%Dual simplex method
clc
clear all
V={'x1','x2','s1','s2','Sol'};
C = [3 9 0 0 0];
Coeff = [2 1 ; 1 2];
b=[8 ;8];
s=eye(size(Coeff,1));
A=[Coeff s b]
BV=[];
for j=1:size(s,2)
    for i=1:size(A,2)
        if A(:,i)==s(:,j)
            BV=[BV i];
        end
    end
end
fprintf('Basic Variables');
disp(V(BV));
ZjCj=C(BV)*A-C;
ZCj=[ZjCj;A];
Dual= array2table(ZCj);
Dual.Properties.VariableNames(1:size(ZCj,2))=V
m=true;
while m
    ZC=ZjCj(:,1:end-1);
    if any(ZC<0)
        fprintf('The current Basic Feasible solution is
not optimal\n');
        [Entval,pvt_col]=min(ZC);
        fprintf('Entering Column = %d\n',pvt_col);
        sol=A(:,end);
        Column=A(:,pvt_col);
        if all(Column<=0)
            fprintf('UNBOUNDED! ');
        else
            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);
        end
        BV(pvt_row)=pvt_col;
        B=A(:,BV);
        A=inv(B)*A;
        ZjCj=C(BV)*A-C;
        ZCj=[ZjCj;A];
        Dual=array2table(ZCj);
        Dual.Properties.VariableNames(1:size(ZCj,2))=V
    else
        m=false;
        fprintf('The Optimal solution is reached ');
        Final_BFS = zeros(1,size(A,2));
        Final_BFS(BV) =A(:,end);
        Final_BFS(end)=sum(Final_BFS.*C);
        OptimalBFS=array2table(Final_BFS);

OptimalBFS.Properties.VariableNames(1:size(OptimalBFS,2))
=V
    end
end
```

OutPut:-

A =

2	1	1	0	8
1	2	0	1	8

Basic Variables {'s1'} {'s2'}

Dual =

3×5 table

x1	x2	s1	s2	Sol
-3	-9	0	0	0
2	1	1	0	8
1	2	0	1	8

The current Basic Feasible solution is not optimal

Entering Column = 2

Leaving Row = 2

Dual =

3×5 table

x1	x2	s1	s2	Sol
1.5	0	0	4.5	36
1.5	0	1	-0.5	4
0.5	1	0	0.5	4

The Optimal solution is reached

OptimalBFS =

1×5 table

x1	x2	s1	s2	Sol
0	4	4	0	36

Least cost method

Code:-

```
%VALIVETI MANIKANTA BHUVANESH
%19BCD7088
%LeastCostMethod
clear all;
clc;
A=[50 40 60]; %row
B=[20 95 35];  %column
C=[5 8 4; 6 6 3 ; 3 9 6]; %cost matrix

%Checked Balanced/Unbalanced Problem
if sum(A)==sum(B)
   fprintf('Given TP is balanced\n');
else
   fprintf('Given TP is unbalanced\n');

if sum(A)<sum(B)
 C(end+1,:)=zeros(1,size(B,2));

 A(end+1)=sum(B)-sum(A);

elseif sum(B)<sum(A)
    C(:,end+1)= zeros(1,size(A,2));
 B(end+1)=sum(A)-sum(B);
end
end

%Least cost method

[m,n]=size(C);
Cost=C;
X=zeros(size(C));
i=1;
j=1;
BFS=m+n-1;

TotalBFS=length(nonzeros(X));
if TotalBFS==BFS
    fprintf('Initial BFS is Non-degenerate \n');
else
fprintf('Initial BFS is Degenerate \n');
end

for i=1:size(C,1)
```

```
    for j=1:size(C,2)
      M=min(min(C));
[p q]=find(M==C);
x1=min(A(p),B(q));
[value,index]=max(x1);
ii=p(index);
jj=q(index);
y=min(A(ii),B(jj));
X(ii,jj)=y;
A(ii)=A(ii)-y;
B(jj)=B(jj)-y;
C(ii,jj)=inf;
    end
end

fprintf('Initial BFS = \n');
IB=array2table(X);
disp(IB)
InitialCost=sum(sum(Cost.*X))
fprintf('Initial BFS cost=%d\n', InitialCost)

% Finding the Optimal solution
u=zeros(1,m); %rows
v=zeros(1,n); %columns
u(1)=0;

b=0;

for i=1:m
    for j=1:n
        if (X(i,j)==0)% Un allocated cells
            continue
        else
            if(j==b+1)
                 v(j)=Cost(i,j)-u(i);
                 b=j;
            else
                 u(i)=Cost(i,j)-v(j);
            end
        end
    end
end
u
v
% Calculating net evaluations
Net=zeros(m,n);
for i=1:m
```

```
    for j=1:n
        if (X(i,j)==0)%Un occupied cells
            Net(i,j)=Cost(i,j)-(u(i)+v(j));
        end
    end
end

Net
%return;
if any(Net <  0,'all')
    fprintf("BFS is not Optimal")
else
    fprintf("BFS is Optimal")
end
```

Output :-

```
Given TP is balanced
Initial BFS is Degenerate
Initial BFS =
    X1    X2    X3
    __    __    __

     0    50     0
     0     5    35
    20    40     0


InitialCost =

   955

Initial BFS cost=955

u =

     8     3     0


v =

     3     9     0


Net =

    -6     0    -4
     0     0     0
     0     0     6

BFS is not Optimal
```

Hungarian Method

Code:-

```
%VALIVETI MANIKANTA BHUVANESH
%19BCD7088
%hungarian method
clc;
clear all;
arr = [140 110 155 170 180;115 100 110 140 155;120 90 135
150 165;30 30 60 60 90;35 15 50 60 85];
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
minimium');
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);

            if(count>0 && count<minz)
                minz=count;
                d = 1;
                y = find(temp(i,:)==0,1);

            end
        end
        for i = 1:size(temp,2)
            count = size(find(temp(:,i)==0),1);

            if (count>0 && count < minz)
```

```
                minz = count;
                d = 0;
                y=find(temp(:,i)==0,1);

            end
        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~=5)
        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)==4)
&& (size(find(temp(:,j)==inf),1)==4))
                    arr(i,j) = arr(i,j)+sub;
                end
            end
        end
    end
    if(lines == 5)
        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

140	110	155	170	180
115	100	110	140	155
120	90	135	150	165
30	30	60	60	90
35	15	50	60	85

after subtracting row minimum and column minimium

30	0	35	30	15
15	0	0	10	0
30	0	35	30	20
0	0	20	0	5
20	0	25	15	15

lines is :

1

lines is :

2

lines is :

3

lines is :

1

lines is :

2

lines is :

3

lines is :

4

lines is :

5

Modified cost matrix

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

Total cost

310

Golden section search method

Code:-

```
%VALIVETI MANIKANTA BHUVANESH
%GOLDEN SECTION SEARCH METHOD
clear
clear all
syms x x1 x2 y
f(x)=(x^2)+(54/x)
a=0;
b=5;
e=0.1;
y=(x*(b-a))+a
f(x)=f(y)
a=0;
b=1;
interval=[a,b]
diff=abs(a-b);
while(diff>e)
    x2=interval(1)+(0.618*diff);
    x1=interval(2)-(0.618*diff);
    p=vpa(f(x1));
    q=vpa(f(x2));
    if (p>q);
    interval=[x1,interval(2)];
    else(p<q);
    interval=[interval(1),x2];
    end
    diff=abs(interval(1)-interval(2));
end
fprintf("Minimum lies in between")
disp(interval)
```

Output:-

f(x) =

54/(5*x) + 25*x^2

interval =

0 1

Minimum lies in between 0.5279 0.6180

Newton Raphson Method

Code:-

```
%VALIVETI MANIKANTA BHUVANESH
%19BCD7088
%NewtonRaphsonMethod
clc;
clear all;
syms x;
f=(x^2)+(54/x)
g=diff(f)
r=diff(f,2)
e=0.0005;
x0=1;
RUN=true;
while(RUN)
    f0=subs(g,x,x0);
    f1=subs(r,x,x0);
    x1=x0-(f0/f1);
    o=abs(subs(g,x,x1));
    if(o<e)
        RUN=false;
    end
    x0=x1;
end
disp(vpa(x1))
```

Output:-

f =

54/x + x^2

g =

2*x - 54/x^2

r =

108/x^3 + 2

2.9999999313284725991391151826 93

Game Theory(Saddle point)

Code:-

```
%VALIVETI MANIKANTA BHUVANESH
%19BCD7088
%Game theroy
clc
clear all
A=[1 3 1;0 -4 -3;1 5 -1]
maxi=max(A)
maxmin=min(maxi)
mini=min(A,[],2)
minmax=max(mini)
if(minmax==maxmin)
    r=find(mini==minmax)
    c=find(maxi==maxmin)
    fprintf("Saddel point is available")
    fprintf("A is giving " )
    disp(r)
    fprintf("B is giving ")
    disp(c)
    fprintf("Value of thr game is " + minmax)
end
```

Output:-

```
A =

   1   3   1
   0  -4  -3
   1   5  -1


maxi =

   1   5   1
```

```
maxmin =

     1


mini =

     1
    -4
    -1


minmax =

     1


r =

     1


c =

     1     3

Saddel point is availableA is giving      1

B is giving      1     3

Value of thr game is 1
```

Lagrange's Multipliers

Code:

```
%VALIVETI MANIKANTA BHUVANESH
%19BCD7088
%LagrangesMethod
clc
clear all
syms x y lam
a=x*y
b=x^3+y^4-1
L=a + lam *b
dx=diff(L,x)==0
dy=diff(L,y)==0
dlam=diff(L,lam)==0
[x,y,lam]=solve(dx,dy,dlam,x,y,lam,'Real',true);
vpa(x)
vpa(y)
vpa(lam)
```

Output:

a =

x*y

b =

x^3 + y^4 - 1

L =

x*y + lam*(x^3 + y^4 - 1)

```
dx =

3*lam*x^2 + y == 0


dy =

4*lam*y^3 + x == 0


dlam =

x^3 + y^4 - 1 == 0


ans =

0.82982653336624344100099149695016
0.82982653336624344100099149695016


ans =

 0.80910671157022121428995304861 62
-0.80910671157022121428995304861 62


ans =

-0.39166062692497864811319120565742
 0.39166062692497864811319120565742
```