Optimization Techniques

Lab report

NAME :Koduri Gokul

REG.NO : 19BCD7006

LINEAR PROGRAMMING PROBLEM

1 SIMPLEX METHOD

CODE:

```
clc
clear all
NoofVar=2;
MaxZ=[10 12];
A=[4 6;3 8];
B=[240;300];
s=eye(size(A,1));
X=[A s B];
cost=[10 12 0 0 0];
BV=NoofVar+1:1:size(X,2)-1;
zjCj=cost(BV)*X-cost;
zcj=[zjCj;X];
SimTable=array2table(zcj);
SimTable.Properties.VariableNames(1:size(zcj,2))={'x1','x2','s1','s2','sol'}
RUN=true;
while RUN
if any(zjCj<0);
fprintf('Current solution is not optimal \n')
fprintf('Next Iteration: \n')
disp('Previous basic variables: ')
disp(BV)
zEntVar=zjCj(1:end-1);
[entrnCol,pivCol]=min(zEntVar);
fprintf('The minimum DelJ is %d \n',entrnCol)
fprintf('The Pivot column is %d \n',pivCol)
fprintf('Entering variable is %d \n',pivCol)
Xb=X(:,end);
Col=X(:,pivCol);
if all(Col<0)
error('Lpp is unbounded as all entries less than zero')
else
for i=1:size(Col,1)
if Col(i)>0
MinRat(i)=Xb(i)./Col(i);
else
MinRat(i)=inf;
end
```

```
end
[minRatio,pvtRow]=min(MinRat)
fprintf('pivot row corresponding to Minimum ratio is %d \n',pvtRow)
fprintf('Leaving variable is %d \n',BV(pvtRow))
end
BV(pvtRow)=pivCol;
disp('New Basic Variables:');
disp(BV);
pvtKey=X(pvtRow,pivCol);
X(pvtRow,:)=X(pvtRow,:)./pvtKey;
for i=1:size(X,1)
if i~=pvtRow
X(i,:)=X(i,:)-X(i,pivCol).*X(pvtRow,:);
end
end
zjCj=zjCj-zjCj(pivCol).*X(pvtRow,:);
zcj=[zjCj;X];
SimTable=array2table(zcj);
SimTable.Properties.VariableNames(1:size(zcj,2))={'x1','x2','s1','s2','sol'}
BFS=zeros(1,size(X,2));
BFS(BV)=X(:,end);
BFS(end)=sum(BFS.*cost);
CurrentBFS=array2table(BFS);
CurrentBFS.Properties.VariableNames(1:size(CurrentBFS,2))={'x1','x2','s1','s2','sol'
}
else
RUN=false;
fprintf('Current Basic feasible sol is optimal \n')
end
end
```

OUTPUT:

```
SimTable =
  3×5 table
    x1     x2     s1    s2    sol
    ___    ___    __    __    ___
    -10    -12    0     0      0
     4      6     1     0     240
     3      8     0     1     300
Current solution is not optimal
Next Iteration:
Previous basic variables:
     3     4
The minimum DelJ is -12
The Pivot column is 2
```

```
Entering variable is 2
minRatio =
   37.5000
pvtRow =
     2
pivot row corresponding to Minimum ratio is 2
Leaving variable is 4
New Basic Variables:
     3     2
SimTable =
  3×5 table
     x1      x2    s1     s2      sol
    _____    __    __    _____   ____
     -5.5    0     0      1.5    450
     1.75    0     1     -0.75    15
    0.375    1     0     0.125    37.5
CurrentBFS =
  1×5 table
    x1     x2     s1    s2    sol
    __    ____    __    __    ___
    0     37.5    15    0     450
Current solution is not optimal
Next Iteration:
Previous basic variables:
     3     2
The minimum DelJ is -5.500000e+00
The Pivot column is 1
Entering variable is 1
minRatio =
    8.5714
pvtRow =
     1
pivot row corresponding to Minimum ratio is 1
Leaving variable is 3
New Basic Variables:
     1     2
SimTable =
  3×5 table
    x1    x2       s1          s2         sol
    __    __    _________    _________    ______


    0     0       3.1429     -0.85714    497.14
    1     0      0.57143     -0.42857     8.5714
    0     1     -0.21429      0.28571     34.286
```

```
CurrentBFS =
  1×5 table
     x1       x2      s1    s2     sol
   ______   ______   __    __   ______
   8.5714   34.286    0     0    497.14
Current solution is not optimal
Next Iteration:
Previous basic variables:
     1     2
The minimum DelJ is -8.571429e-01
The Pivot column is 4
Entering variable is 4
minRatio =
   120
pvtRow =
     2
pivot row corresponding to Minimum ratio is 2
Leaving variable is 2
New Basic Variables:
     1     4
SimTable =
  3×5 table
    x1    x2      s1      s2    sol
    __   ___   _____    __   ___
    0     3     2.5     0    600
    1    1.5    0.25     0     60
    0    3.5   -0.75     1    120
CurrentBFS =
  1×5 table
    x1    x2    s1    s2     sol
    __   __    __   ___    ___
    60   0     0    120    600
Current Basic feasible sol is optimal
```

2 GRAPHICAL SOLUTION METHOD:

z=-7*x1-5*x2

S.t.c

4*x1+3*x2=2

402*x1+1*x2

=100

x1>0,x2>0

Find maximum value of the above function.

CODE:

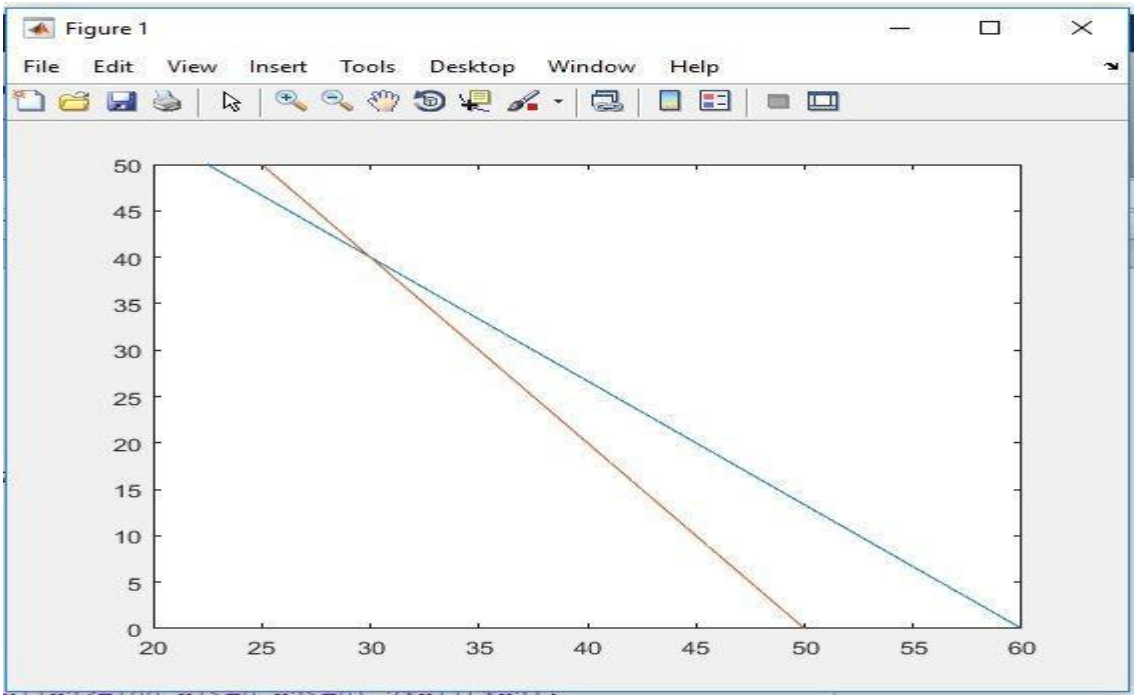
```
syms x1 x2 k f=[-7,-5];
  A=[4,3;2,
  1];
  B=[240,1
  00];
  lb=zeros(2,1);
  Aeq=[];
  Beq=[];
[x,fval]=linprog(f,A,B,Aeq,Be
q,lb); x
  fval
  x2=0:5
  0;
  x1=(240-3*x2)/4;
  plot(x1,x2)
  hold on
  x1=(100-x2
  )/2;
  plot(x1,x2)
  k :=[{4*x1+3*x2<=240,2*x1+x2<=100,x1>=0,x2>=0},7*x1+5*x2]:
  g
  :=linpot::plot_data(k,[x1,
  x2]): plot(g)
  Command

  window(output): Q1

  Optimal solution

  found.

  x =


30.0000
```

40.0000

fval =

-410

OUTPUT:



3 BIG-M METHOD

CODE:

```
format short
clc
clear all
variables = {'x_1','x_2','s_1','s_2','A_1','A_2','sol'};
M=1000;
cost=[-2 -3 0 0 -M -M 0];
A= [0.5 0.25 1 0 0 0 4; 1 3 0 -1 1 0 20; 1 1 0 0 0 1 10];
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;
SimpTable;
RUN = true;
while RUN
ZC = ZjCj(:,1:end-1);
if any(ZC<0)
fprintf('The current BFS is not optimal solution\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('solution is 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);
BV(pvt_row)=pvt_col;
```

```
B = A(:,BV);
A=inv(B)*A;
ZjCj = cost(BV)*A-cost;
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;
OptimalBFS;
```

OUTPUT:

```
The current BFS is not optimal solution
Entering column = 2
Leaving row = 2
The current BFS is not optimal solution
Entering column = 1
Leaving row = 3
```

4 DUAL SIMPLEX METHOD

CODE:

```
clc
clear all
close all
prob = optimproblem('ObjectiveSense','min');
x = optimvar('x',2,1,'LowerBound',0);
prob.Objective = 3*x(1) + x(2);
cons1 = 10*x(1)+2*x(2)>=84
cons2 = 8*x(1)+4*x(2) >= 120;

prob.Constraints.cons1 = cons1;
prob.Constraints.cons2 = cons2;

show(prob)
```

```
sol = solve(prob);
sol.x
x1=sol.x(1)
x2=sol.x(2)
subs(3*x1 + x2)
```

OUTPUT:

```
cons1 =


  Linear OptimizationInequality


    10*x(1) + 2*x(2) >= 84
  OptimizationProblem :
          Solve for:
       x
          minimize :
       3*x(1) + x(2)
          subject to cons1:
       10*x(1) + 2*x(2) >= 84
          subject to cons2:
       8*x(1) + 4*x(2) >= 120
          variable bounds:
       0 <= x(1)
       0 <= x(2)
Solving problem using linprog.
Optimal solution found.

ans =

    4.0000
   22.0000


x1 =

    4.0000


x2 =

   22.0000


ans =

34
```

TRANSPORTATION PROBLEM

5 NORTH WEST CORNER METHOD

CODE:

```
clear all
close all
A=[6 0 10;7 11 11;4 3 12];
dem=[200 100 300];
sup=[150 175 275];


sum=0;
i=1;j=1;
while i<=3 && j<=3
    Min=min(dem(j) & sup(i));
    sum=sum+A(i,j)*Min;
    dem(j)=dem(j)-Min;
    sup(i)=sup(i)-Min;
    if dem(j)==0
      j=j+1;
    end
    if sup(i)==0
     i=i+1;
   end
end
disp("Minimum Cost = "+ num2str(sum))
```

OUTPUT:

Minimum Cost = 5925

VAM

CODE:

```
clc

clearall

a1=[16, 20, 12 ; 14, 8, 18 ; 26, 24, 16];
```

```
b1=[180 120 150];

c1=[200; 160; 90];

[vam,answ]=VAMsm(a1,c1,b1)

vam

answ

function [ Solution, OverallCost ] = VAM(CostsMtx, resources_col, demands_row)

C_start = CostsMtx;

C = C_start;

m =

size(C,1);

n = size(C,2);

a =

resources_col; b

= demands_row;

X = zeros(m,n);

stop = 0;

while stop == 0

for i = 1:m

for j = 1:n

if a(i,1) == 0

C(i,j) = max(C(:,j));

end

if b(1,j) == 0

C(i,j) =

max(C(i,:)); end

end

end
```

```
C_sort_col =
sort(C,1);

C_sort_row =
sort(C,2);

Diff_customer = abs(C_sort_col(1,:) -

C_sort_col(2,:)); Diff_supplier =

abs(C_sort_row(:,1) - C_sort_row(:,2)); for i = 1:m

if a(i,1) == 0

Diff_supplier(i,1) =

0; end

end

for j = 1:n

if b(1,j) == 0

Diff_customer(1,j) = 0;

end


end

Max_Diff_customer = max(Diff_customer);

Max_Diff_supplier = max(Diff_supplier);

Customer_nr = find(Diff_customer==max(Max_Diff_customer,Max_Diff_supplier));

Supplier_nr = find(Diff_supplier==max(Max_Diff_customer,Max_Diff_supplier));

if isempty(Customer_nr) == 0

Supplier_nr_ = find(C(:,Customer_nr(1)) == min(C(:,Customer_nr(1))));

X(Supplier_nr_(1),Customer_nr(1)) = min(a(Supplier_nr_(1),1),b(1,Customer_nr(1)));

a(Supplier_nr_(1),1) = a(Supplier_nr_(1),1) - X(Supplier_nr_(1),Customer_nr(1));

b(1,Customer_nr(1)) = b(1,Customer_nr(1)) - X(Supplier_nr_(1),Customer_nr(1));

Supplier_nr = [];

end

if isempty(Supplier_nr) == 0
```

```
Customer_nr_ = find(C(Supplier_nr(1),:) == min(C(Supplier_nr(1),:)));

X(Supplier_nr(1),Customer_nr_(1)) = min(a(Supplier_nr(1),1),b(1,Customer_nr_(1)));

a(Supplier_nr(1),1) = a(Supplier_nr(1),1) - X(Supplier_nr(1),Customer_nr_(1));

b(1,Customer_nr_(1)) = b(1,Customer_nr_(1)) - X(Supplier_nr(1),Customer_nr_(1));

end

%Stop condition:

a1 = a > 0;

b1 = b > 0;

if sum(a1) == 1

stop = 1;

for j = 1:n

if b(j) > 0;

X(a1 == 1,j) = b(j);

en

d

en

d

en

d

if sum(b1) == 1

stop = 1;

for i = 1:m

if a(i) > 0;

X(i,b1 == 1) = a(i);

en

d

en
```

```
d

en

d

en

d

Solution = X;

OverallCost = sum(sum(C_start .* X));

 end
```

OUTPUT:

```
 vam =


  140    0   60
   40  120    0
    0    0 90


answ =

      5920
```

6 MODI METHOD

CODE:

```
clc
clear all
C=input('Enter Transportation Matrix:-\n')
S=input('Enter Supply column:-\n')
D=input('Enter Demand row:=\n')
A=[C S;D]
[m,n]=size(A);
rsum=sum(A(1:m-1,n))
csum=sum(A(m,1:n-1));
if(rsum==csum)
    for i=1:m-1
```

```
        for j=1:n-1
          X(i,j)=min(A(i,n),A(m,j));
         A(i,n)=A(i,n)-X(i,j);
        A(m,j)=A(m,j)-X(i,j);
     end
   end
else
    display('Unbalanced problem')
end
X
size(A)
TIC=0;
for i=1:m-1
  for j=1:n-1
   TIC=TIC+A(i,j)*X(i,j);
   end
end
TIC
a=1;
b=0;
u=zeros(1,m-1);
v=zeros(n-1,1);
u(1)=0;
for i=1:m-1
   for j=1:n-1
     if (X(i,j)==0)
       continue
       else
         if (j==b+1)
           v(j)=A(i,j)-u(i)
             b=j;
           else
            u(i)=A(i,j)-v(j)
         end
      end
   end
end
u
v
del=zeros(m,n);
for i=1:m-1
    for j=1:n-1
        if (X(i,j)==0);
            del(i,j)=A(i,j)-(u(i)+v(j));
end
end
```

```
end
del
TFC=0;
for i=1:m-1
for j=1:n-1
TFC = TFC+A(i,j)*X(i,j);
end
end
```

OUTPUT:

```
Enter Transportation Matrix:-
 [10 7 8;15 29 9;7 8 12]
C =

    10     7     8
    15    29     9
     7     8    12
Enter Supply column:-
[45;15;40]

S =

    45
    15
    40

Enter Demand row:=
[25 55 20 0]

D =

    25    55    20     0


A =

    10     7     8    45
    15    29     9    15
     7     8    12    40
    25    55    20     0


rsum =

   100


X =
```

```
    25    20     0
     0    15     0
     0    20    20


ans =

     4     4


TIC =

        1225


v =

    10
     0
     0


v =

    10
     7
     0


u =

     0    22     0


u =

     0    22     1


v =

    10
     7
    11


u =

     0    22     1


v =
```

```
   10
    7
   11


del =

    0    0   -3    0
  -17    0  -24    0
   -4    0    0    0
    0    0    0    0
```

7 ASSIGNMENT PROBLEM

Hungarian method:

CODE:

```
clc
clear all
C = [5 7 11 6;8 5 9 6;4 7 10 7;10 4 8 3];
disp("Cost Matrix : ")
disp(C)
B=C;
for i=1:size(C,1)
    r_min=min(C(i,:));
    C(i,:)=C(i,:)-r_min;
end
disp("Resultant matrix after subtracting the row minima and column minima : " );
disp(C)
RUN=true;
while RUN
    temp=C;
    lines=0;
    while RUN
        minz=inf;
        for i = 1:size(temp,1)
            count=size(find(temp(i,:)==0),2);
            disp("Count in row wise : "+ count)

            if(count>0 && count<minz)
                minz=count;
                d=1;
                y=find(temp(i,:)==0,1);
                disp("y1 is : "+ y)
```

```
            end
        end
        for i=1:size(temp,2)
            count=size(find(temp(:,1)==0),1);
            disp("Count in col wise : "+ count)

            if(count>0 && count<minz)
                minz=count;
                d=0;
                y=find(temp(:,1)==0,1);
                disp("y2 is : "+y)

            end
            disp("y is : "+y)

        end
        if minz==inf
            break;
        end
    if d==1
        temp(:,y)=inf;
    else
        temp(y,:)=inf;
    end
    lines=lines+1;
    disp("Lines is : "+lines)

    end
sub=min(min(temp));
if(lines~=4)
    for i=1:size(C,1)
        for j=1:size(C,2)
            if(temp(i,j)~=inf)
                C(i,j)=C(i,j)-sub;
            elseif(size(find(temp(i,:)==inf),2)==4)&&(size(find(temp(:,j)==inf),1==4))
                C(i,j)=C(i,j)+sub;
            end
        end
    end
end
    if(lines==4)
        break;
    end
end
disp('Modified cost Matrix : ');
```

```
disp(C)
total_cost=0;

for i=1:size(C,1)
    for j=1:size(C,2)
       if(C(i,j)==0)
          total_cost=total_cost+B(i,j);
          for k=j+1:size(C,2)
             if(C(i,k)==0)
                C(i,k)=inf;
             end
          end
          for k=i+1:size(C,1)
             if(C(k,j)==0)
                C(k,j)=inf;
             end
          end
       end
    end
 end
disp("Total Cost : "+total_cost)
```

OUTPUT:

```
Cost Matrix :
    5    7   11    6
    8    5    9    6
    4    7   10    7
   10    4    8    3

Resultant matrix after subtracting the row minima and column minima :
    0    2    6    1
    3    0    4    1
    0    3    6    3
    7    1    5    0

Count in row wise : 1
y1 is : 1
Count in row wise : 1
Count in row wise : 1
Count in row wise : 1
Count in col wise : 2
y is : 1
Count in col wise : 2
y is : 1
Count in col wise : 2
y is : 1
Count in col wise : 2
```

y is : 1
Lines is : 1
Count in row wise : 0
Count in row wise : 1
y1 is : 2
Count in row wise : 0
Count in row wise : 1
Count in col wise : 0
y is : 2
Count in col wise : 0
y is : 2
Count in col wise : 0
y is : 2
Count in col wise : 0
y is : 2
Lines is : 2
Count in row wise : 0
Count in row wise : 0
Count in row wise : 0
Count in row wise : 1
y1 is : 4
Count in col wise : 0
y is : 4
Count in col wise : 0
y is : 4
Count in col wise : 0
y is : 4
Count in col wise : 0
y is : 4
Lines is : 3
Count in row wise : 0
Count in row wise : 0
Count in row wise : 0
Count in row wise : 0
Count in col wise : 0
y is : 4
Count in col wise : 0
y is : 4
Count in col wise : 0
y is : 4
Count in col wise : 0
y is : 4
Count in row wise : 1
y1 is : 1
Count in row wise : 2
Count in row wise : 1
Count in row wise : 1
Count in col wise : 2
y is : 1
Count in col wise : 2
y is : 1
Count in col wise : 2

y is : 1
Count in col wise : 2
y is : 1
Lines is : 1
Count in row wise : 0
Count in row wise : 2
y1 is : 2
Count in row wise : 0
Count in row wise : 1
y1 is : 4
Count in col wise : 0
y is : 4
Count in col wise : 0
y is : 4
Count in col wise : 0
y is : 4
Count in col wise : 0
y is : 4
Lines is : 2
Count in row wise : 0
Count in row wise : 2
y1 is : 2
Count in row wise : 0
Count in row wise : 0
Count in col wise : 0
y is : 2
Count in col wise : 0
y is : 2
Count in col wise : 0
y is : 2
Count in col wise : 0
y is : 2
Lines is : 3
Count in row wise : 0
Count in row wise : 1
y1 is : 3
Count in row wise : 0
Count in row wise : 0
Count in col wise : 0
y is : 3
Count in col wise : 0
y is : 3
Count in col wise : 0
y is : 3
Count in col wise : 0
y is : 3
Lines is : 4
Count in row wise : 0
Count in row wise : 0
Count in row wise : 0
Count in row wise : 0
Count in col wise : 0

```
y is : 3
Count in col wise : 0
y is : 3
Count in col wise : 0
y is : 3
Count in col wise : 0
y is : 3
Modified cost Matrix :
    0    2    2    1
    3    0    0    1
    0    3    2    3
    7    1    1    0

Total Cost : 13
```

8 FIBONACCI SEARCH METHOD

CODE:

```
clc
clear
sc = inputdlg('Type an expression that is a function of x ' );
s = sc{:};
f = str2func(['@(x) ' s])
a = input ('Enter lower boundary point: ');
b = input ('Enter upper boundary point: ');
n = input ('Enter the desired number of function evaluation (greater than 2): ');
k = 2;
L = b - a;
y1 = feval(f,(a));
y2 = feval(f,(b));
while k <= n
x_1 = n-k+1;
f_x_1 = fibonn(n-k+1);
f_x_2 = fibonn(n+1);
Lk_star = (f_x_1/f_x_2)* L
x1 = a + Lk_star;
x2 = b - Lk_star;
if mod (k,2) == 0
y1 = feval(f,(x1));
else
y2 = feval(f,(x2));
end
if y1 > y2
a = x1;
else
b = x2;
```

```
end
c = 0.5*(a+b);
k = k + 1;
end
a
b
fprintf ('The minimum value is: %f \n',c)
function f_x = fibonn(x)
if x == 0
   f_x = 1;
   elseif x == 1
   f_x = 1;
   else
   a = 1;
   b = 1;
   for i = 2:x
   f_x = a + b;
   a = b;
   b = f_x;
   end
end
end
```

OUTPUT:

```
f =
  function_handle with value:

    @(x)x+x^2

Enter lower boundary point:
-1
Enter upper boundary point:
1
Enter the desired number of function evaluation (greater than 2):
5

Lk_star =

    0.7692

Lk_star =

    0.4615

Lk_star =

    0.3077

Lk_star =

    0.1538
```

a =

-0.2308

b =

0.0769

The minimum value is: -0.076923

9 LAGRANGE METHOD

CODE:

```
close all
clear all
clc
syms x1 x2 lam1 lam2 h1 F z1
F = 10*x1+4*x2-2*x1*x1-x2*x2+lam1*(2*x1+x2+z1*z1-5);
h1=2*x1+x2+z1*z1-5;
grad1=diff(F,x1);
grad2=diff(F,x2);
grad3=diff(F,z1);
[lams1,xs1,xs2,z]=solve(grad1,grad2,grad3,h1);
double([lams1 xs1 xs2 z]);
n=length(xs1)
for i=1:n
x1=xs1(i);
x2=xs2(i);
f=10*x1+4*x2-2*x1*x1-x2*x2;
x1x2f=[x1 x2 f]
end
```

OUTPUT:

n =

3

x1x2f =

[11/6, 4/3, 91/6]

x1x2f =

[5/2, 2, 33/2]

```
x1x2f =
[5/2, 2, 33/2]
```

10 DIJIKSTRA'S ALGORITHM

CODE:

```
clear All
clc
map = [0 100 30 0 0;
       0 0 20 0 0;
       0 0 0 10 60;
       0 15 0 0 50;
       0 0 0 0 0];
[e,l]=dijkstra(map,1,5);
disp(e)
disp(l)
function [e,L] = dijkstra(A,s,d)
if s==d
   e=0;
   L=[s];
else
A = setupgraph(A,inf,1);
if d==1
   d=s;
end
A=exchangenode(A,1,s);
lengthA=size(A,1);
W=zeros(lengthA);
for i=2 : lengthA
   W(1,i)=i;
   W(2,i)=A(1,i);
end
for i=1 : lengthA
   D(i,1)=A(1,i);
   D(i,2)=i;
end
D2=D(2:length(D),:);
L=2;
while L<=(size(W,1)-1)
   L=L+1;
   D2=sortrows(D2,1);
   k=D2(1,2);
   W(L,1)=k;
   D2(1,:)=[];
   for i=1 : size(D2,1)
      if D(D2(i,2),1)>(D(k,1)+A(k,D2(i,2)))
         D(D2(i,2),1) = D(k,1)+A(k,D2(i,2));
         D2(i,1) = D(D2(i,2),1);
      end
```

```
    end

    for i=2 : length(A)
        W(L,i)=D(i,1);
    end
end
if d==s
    L=[1];
else
    L=[d];
end
e=W(size(W,1),d);
L = listdijkstra(L,W,s,d);
end
end


function G = exchangenode(G,a,b)
buffer=G(:,a);
G(:,a)=G(:,b);
G(:,b)=buffer;
buffer=G(a,:);
G(a,:)=G(b,:);
G(b,:)=buffer;
end


function L = listdijkstra(L,W,s,d)
index=size(W,1);
while index>0
if W(2,d)==W(size(W,1),d)
L=[L s];
index=0;
else
index2=size(W,1);
while index2>0
if W(index2,d)<W(index2-1,d)
L=[L W(index2,1)];
L=listdijkstra(L,W,s,W(index2,1));
index2=0;
else
index2=index2-1;
end
index=0;
end
end
end
end


function G = setupgraph(G,b,s)
if s==1
for i=1 : size(G,1)
for j=1 :size(G,1)
if G(i,j)==0
```

```
G(i,j)=b;
end
end
end
end
if s==2
for i=1 : size(G,1)
for j=1 : size(G,1)
if G(i,j)==b
G(i,j)=0;
end
end
end
end
end
```

OUTPUT:

```
 90

 5     3     1
```

PROBLEMS

Problem 1:

Consider a small plant which makes 2 types of automobile parts say A and B. It buys castings that are machined, bored and polished. The capacity of machining is 25 per hour for A and 40 hours for B, capacity of boring is 28 per hours for A and 35 per hour for B, and the capacity of polishing is 35 per hour A and 25 hours of B. Casting for port A costs Rs. 2 each and for part B they cost Rs. 3 each. They sell for Rs. 5 and Rs. 6 respectively. The three machines have running costs of Rs. 20, Rs. 14 and Rs. 17.50 per hour. Assuming that any combination of parts A and B can be sold, what product mix maximizes profit?

CODE:

OUTPUT:

Problem 2:

Convert the transportation problem shown in Table 5.11 into a balanced problem. Find the initial feasible solution of the transportation problem.

Table 5.11: Demand Exceeding Supply

Source	Destination				Supply
	1	2	3	4	
1	10	16	9	12	200
2	12	12	13	5	300
3	14	8	13	4	300
Demand	100	200	450	250	1000/800

CODE:

```
clc
clear all
Cost = [10 16 9 12;12 12 13 5;14 8 13 4]
A = [200 300 300]
B = [100 200 450 250]
if sum(A) == sum(B)
fprintf('Given Transportation Problem is Balanced\n')
else
fprintf('Given Transportation Problem is UnBalanced\n')
if sum(A)<sum(B)
Cost(end+1,:)=zeros(1,size(A,3));
A(end+1) = sum(B)-sum(A);
elseif sum(B)<sum(A)
Cost(:,end+1) = zeros(1,size(A,3));
B(end+1) = sum(A)-sum(B);
end
end
X = zeros(size(Cost));
[m,n] = size(Cost);
BFS = m+n-1;
i =1;
j =1;
l = 0;
while(l<BFS)
if A(i)<=B(j)
X(i,j) = A(i);
B(j) = B(j)-A(i);
i = i+1;
l = l+1;
elseif B(j)<=A(i)
X(i,j) = B(j);
A(i) = A(i) - B(j);
j = j+1;
l = l+1;
else
break
end
end
fprintf('Intial BFS = \n')
IB = array2table(X);
disp(IB);
TotalBFS = length(nonzeros(X));
if TotalBFS == BFS
fprintf('Intial BFS is NON-Degenerate \n');
else
fprintf('Initial BFS is Degenerate \n');
end
InitialCost = sum(sum(Cost.*X));
fprintf('Initial BFS Cost = %d\n',InitialCost);
```

OUTPUT:

```
Cost =

    10    16     9    12
    12    12    13     5

    14     8    13     4


A =

   200   300   300


B =

   100   200   450   250

Given Transportation Problem is UnBalanced
Intial BFS =
    X1     X2     X3     X4
    ___    ___    ___    ___

    100    100      0      0
      0    100    200      0
      0      0    250     50
      0      0      0    200

Intial BFS is NON-Degenerate
Initial BFS Cost = 9850
```

Problem 3:

Consider the following transportation problem (cost in rupees). Find the optimum solution.

Factory	D	E	F	G	Capacity
A	4	6	8	6	700
B	3	5	2	5	400
C	3	9	6	5	600
Requirement	400	450	350	500	1700

CODE:

```
clear
clc
Cost = [4 6 8 6;3 5 2 5;3 9 6 5];
A = [700 400 600];
B = [400 450 350 500];
if sum(A) == sum(B)
fprintf('Given Transportation Problem is Balanced\n');
else
fprintf('Given Transportation Problem is UnBalanced\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
X = zeros(size(Cost));
[m,n] = size(Cost);
BFS = m+n-1;
i =1;
j =1;
l = 0;
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('Intial BFS = \n');
IB = array2table(X);
disp(IB);
TotalBFS = length(nonzeros(X));
if TotalBFS == BFS
fprintf('Intial BFS is NON-Degenerate \n');
else
fprintf('Initial BFS is Degenerate \n');
end
InitialCost = sum(sum(Cost.*X));
fprintf('Initial BFS Cost = %d\n',InitialCost);
```

OUTPUT:

```
Given Transportation Problem is Balanced
Intial BFS =
    X1      X2      X3      X4
    ___     ___     ___     ___

    400     300       0       0
      0     150     250       0
      0       0     100     500
```

```
Intial BFS is NON-Degenerate
Initial BFS Cost = 7750
```

Problem 4:

A project consists of 4 major jobs for which 4 contractors have submitted their tenders. The tender amount quoted

in lakhs of rupees are given in the matrix below:

Jobs		A	B	C	D
Contractors	1	10	24	30	15
	2	16	22	28	12
	3	12	20	32	10
	4	9	26	34	16

Find the assignment which minimizes the total project cost.

CODE:

```
clc;
clear all;
C = [10 24 30 15;16 22 28 12;12 20 32 10;9 26 34 16];
disp('cost matrix');
disp(C);
total_cost=0;
B = C;
for i = 1:size(C,1)
r_min = min(C(i,:));
C(i,:)-C(i,:)-r_min;
end
for i = 1:size(C,2)
c_min = min(C(:,i));
C(:,i) = C(:,i)-c_min;
end
disp('Resultant matrix after subtracting the row minima and');
disp(C);
RUN=true;
while RUN
temp = C;
lines = 0;
while RUN
minz =inf;
for i=1:size(temp,1)
count = size(find(temp(i,:)==0),2);
disp('count in row wise: ')
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('coutn in colwise: ')
disp(count)
if(count>0 && count<minz)
minz = count;
d=0;
y = find(temp(:,i)==0,1);
disp('y2 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~=4)
for i = 1:size(C,1)
for j = 1:size(C,2)
if(temp(i,j)~=inf)
C(i,j)=C(i,j)-sub;;
elseif(size(find(temp(i,:)==inf),2)==4)&&(size(find(temp(:,j)==inf),1)==4)
C(i,j)=C(i,j)+sub;
end
end
end
end
if(lines==4)
break;
end
end
for i =1:size(C,1)
for j = 1:size(C,2)
if(C(i,j)==0)
total_cost =total_cost+B(i,j);
for k=j+1:size(C,2)
if(C(i,k)==0)
C(i,k) = inf;
end
end
for k=i+1:size(C,1)
if(C(k,j)==0)
C(k,j)=inf;
end
end
end
end
end
disp('Total cost')
disp(total_cost)
```

OUTPUT:

```
cost matrix
    10    24    30    15
    16    22    28    12
    12    20    32    10
     9    26    34    16

Resultant matrix after subtracting the row minima and
     1     4     2     5
     7     2     0     2
     3     0     4     0
     0     6     6     6

count in row wise:
     0

count in row wise:
     1
```

```
y1 is:
     3

count in row wise:
     2

count in row wise:
     1

coutn in colwise:
     1

y is:
     3

coutn in colwise:
     1

y is:
     3

coutn in colwise:
     1

y is:
     3

coutn in colwise:
     1

y is:
     3

lines is:
     1

count in row wise:
     0

count in row wise:
     0

count in row wise:
     2

y1 is:
     2

count in row wise:
     1

y1 is:
     1

coutn in colwise:
     1

y is:
     1

coutn in colwise:
```

```
     1

y is:
     1

coutn in colwise:
     0

y is:
     1

coutn in colwise:
     1

y is:
     1

lines is:
     2

count in row wise:
     0

count in row wise:
     0

count in row wise:
     2

y1 is:
     2

count in row wise:
     0

coutn in colwise:
     0

y is:
     2

coutn in colwise:
     1

y2 is:
     3

y is:
     3

coutn in colwise:
     0

y is:
     3

coutn in colwise:
     1

y is:
     3
```

```
lines is:
     3

count in row wise:
     0

count in row wise:
     0

count in row wise:
     0

count in row wise:
     0

coutn in colwise:
     0

y is:
     3

coutn in colwise:
     0

y is:
     3

coutn in colwise:
     0

y is:
     3

coutn in colwise:
     0

y is:
     3

count in row wise:
     0

count in row wise:
     3

y1 is:
     2

count in row wise:
     2

y1 is:
     2

count in row wise:
     1

y1 is:
     1
```

```
coutn in colwise:
     1

y is:
     1

coutn in colwise:
     2

y is:
     1

coutn in colwise:
     1

y is:
     1

coutn in colwise:
     2

y is:
     1

lines is:
     1

count in row wise:
     0

count in row wise:
     3

y1 is:
     2

count in row wise:
     2

y1 is:
     2

count in row wise:
     0

coutn in colwise:
     0

y is:
     2

coutn in colwise:
     2

y is:
     2

coutn in colwise:
     1

y2 is:
```

```
     2

y is:
     2

coutn in colwise:
     2

y is:
     2

lines is:
     2

count in row wise:
     0

count in row wise:
     0

count in row wise:
     2

y1 is:
     2

count in row wise:
     0

coutn in colwise:
     0

y is:
     2

coutn in colwise:
     1

y2 is:
     3

y is:
     3

coutn in colwise:
     0

y is:
     3

coutn in colwise:
     1

y is:
     3

lines is:
     3

count in row wise:
     0
```

```
count in row wise:
     0

count in row wise:
     0

count in row wise:
     0

coutn in colwise:
     0

y is:
     3

coutn in colwise:
     0

y is:
     3

coutn in colwise:
     0

y is:
     3

coutn in colwise:
     0

y is:
     3

count in row wise:
     2

y1 is:
     2

count in row wise:
     3

count in row wise:
     2

count in row wise:
     1

y1 is:
     1

coutn in colwise:
     1

y is:
     1

coutn in colwise:
     3
```

```
y is:
     1

coutn in colwise:
     2

y is:
     1

coutn in colwise:
     2

y is:
     1

lines is:
     1

count in row wise:
     2

y1 is:
     2

count in row wise:
     3

count in row wise:
     2

count in row wise:
     0

coutn in colwise:
     0

y is:
     2

coutn in colwise:
     3

y is:
     2

coutn in colwise:
     2

y is:
     2

coutn in colwise:
     2

y is:
     2

lines is:
     2

count in row wise:
```

```
     1

y1 is:
     3

count in row wise:
     2

count in row wise:
     1

count in row wise:
     0

coutn in colwise:
     0

y is:
     3

coutn in colwise:
     0

y is:
     3

coutn in colwise:
     2

y is:
     3

coutn in colwise:
     2

y is:
     3

lines is:
     3

count in row wise:
     0

count in row wise:
     1

y1 is:
     4

count in row wise:
     1

count in row wise:
     0

coutn in colwise:
     0

y is:
     4
```

```
coutn in colwise:
     0

y is:
     4

coutn in colwise:
     0

y is:
     4

coutn in colwise:
     2

y is:
     4

lines is:
     4

count in row wise:
     0

count in row wise:
     0

count in row wise:
     0

count in row wise:
     0

coutn in colwise:
     0

y is:
     4

coutn in colwise:
     0

y is:
     4

coutn in colwise:
     0

y is:
     4

coutn in colwise:
     0

y is:
     4

Total cost
    71
```

The End