Appendix

The numerical study for the various chapters was implemented using *MATLAB*-7.7. Several programmes were written for this purpose, and the important ones are given in this section. For easy understanding of various stages of the programmes, comments have been given wherever necessary. The following programmes are listed in the appendix.

Programme-1: To find the efficiency through aggregate directional distance formulation of data envelopment model (Chapter-2).

Programme-2: To find efficiency through non-radial, non-oriented aggregate directional distance formulation of data envelopment model with integer and real valued variable (Chapter-2).

Programme-3: To find efficiency through improved efficiency measure of directional distance formulation of data envelopment model (Chapter-3).

Programme-4: Modified directional distance formulation of DEA to assess bankruptcy (Chapter-4).

Programme-5: Ranking Efficient DMUs through single virtual DMU of multiplier CRS model (Chapter-5).

Programme-6: Ranking efficient DMUS through changing reference technology (Chapter-5).

Programme-7: Aggregate directional distance formulation of data envelopment model with uncontrollable variables (Chapter-6).

Programme-8: Measuring human development index (Chapter-7).

Programme-1

```
clear all

%%=========================================================
% Aggregate Directional Distance Model
%==========================================================
% Inputs are

iput=xlsread('eg','Sheet1');
oput=xlsread('eg','Sheet2');

% % iput=xlsread('pareto_koopman','Sheet2');
% % oput=xlsread('pareto_koopman','Sheet1');
%
x1=iput';
y1=oput';
%
```

```
  n=size(x1,2); % Number of columns in the input matrix
  m=size(x1,1) ;% Number of rows in the input matrix
  k=size(y1,1) ;% Number of rows in the output matrix
%

maxm=max(y1');
minm=min(x1');
%
  if n ~= size(y1,2);
      disp('Invalid number of DMU at output Matrix')
  end
%
  if size(x1,2)~=size(y1,2);
      disp('Invalid number of DMU entered at either input matrix or
output Matrix')
  end

% Objective function is

    f=(1/(k+m))*[ones(1,k)*-1 ones(1,m)*-1  zeros(1,n)]; % Equal
weight

  for i=1:n
     fprintf('\n DMU number %d\n\n',i)

Rio1=(x1(:,i)-(minm'));

for ip=1:m                          % To avoid 0/0

    if Rio1(ip,1)==0;

    Rio1(ip,1)=0.0001;

end
end

 for t=1:m
      Rio(t,t)=Rio1(t);

  end

     Ryo1=(maxm')-y1(:,i) ;

   for op=1:k                              % To avoid 0/0

    if Ryo1(op,1)==0;

     Ryo1(op,1)=0.0001;

end
end

%  Ryo1=Ry1'

 Ryo2(:,i)=Ryo1;

 xval=[zeros(m,k) Rio x1 ];
```

```
  Ryo=zeros(k,k);

  for p=1:k
      Ryo(p,p)=Ryo1(p);
  end

    yval=[Ryo zeros(k,m) y1*(-1)];


      A=[yval;xval];

      b=[y1(:,i)*-1;x1(:,i)] ;

      Aeq=[zeros(1,k)  zeros(1,m) ones(1,n)] ;

    beq=[1] ;

% Lower bounds on decision variables are

lb=[zeros(m,1);zeros(n,1); zeros(k,1)];          %lower bounds

 ub=[ones(m,1);ones(n,1)];           % Upper bounds


options =
optimset('LargeScale','on','MaxIter',1000000,'Display','iter');

[w,fval,EXITFLAG,options]=linprog(f,A,b,Aeq,beq,lb,ub) ;

 % fprintf('In-Efficiency measure beta = %1.4f\n\n', (fval(1)));

ww(:,i)=w;

   op_proj=y1*w(m+k+1:n+m+k);    % Projected values of output
variables

 op_pro1(:,i)=op_proj;

beta=round((fval)*10000);
beta=(beta/10000)*-1;
bt(i)=beta;
e(i)=1-beta;

beta1=round(w(1)*10000);
beta1=(beta1/10000);
b1(i)=beta1;

beta2=round(w(2)*10000);
beta2=(beta2/10000);
b2(i)=beta2;

beta3=round(w(3)*10000);
beta3=(beta3/10000);
b3(i)=beta3;
```

```
 bby=[beta1 beta2 beta3];


 bbby=bby*Ryo;

 oppro=y1(:,i)+bbby';

 op2(:,i)=oppro;

 beta4=round(w(4)*10000);

 beta4=(beta4/10000);
b4(i)=beta4;

beta5=round(w(5)*10000);
beta5=(beta5/10000);
b5(i)=beta5;

 ip_proj=x1*w(m+k+1:n+m+k);    % Projected values of input variables

  ip_pro1(:,i)=ip_proj;

  bbx=[beta4 beta5];

   bbbx=bbx*Rio;

 ip_pro=x1(:,i)-bbbx';
   ip2(:,i)=ip_pro;


 end

sl=1:n+m+k;

www=[sl' ww];
```

****

Program-2

```
clear all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Program to find efficiency in the case of negative data using
%
% directional distance approach NON_RADIAL Non-oriented ORIENTED
MODEL
% INTEGER and REAL VALUED inputs Outputs
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ip=xlsread('integer','input');
op=xlsread('integer','output');

x1=ip';
```

```
y1=op';%

  n=size(x1,2); % Number of columns in the input matrix
  m=size(x1,1) ;% Number of rows in the input matrix
  k=size(y1,1) ;% Number of rows in the output matrix
  yint=y1(1,:); % Integer Outputs
  yr=y1(2,:);   % Real valued outputs
%

pi=m           ; %Number of integer input
pr=1            ; %Number of integer output

maxm=max(y1');
minm=min(x1');
%
  if n ~= size(y1,2);
      disp('Invalid number of DMU at output Matrix')
  end
%
  if size(x1,2)~=size(y1,2);
      disp('Invalid number of DMU entered at either input matrix or
output Matrix')
  end

% Inputs are

  for i=1:n
     fprintf('\n DMU number %d\n\n',i)

Rio1=(x1(:,i)-(minm'));


for ip=1:m                          % To avoid 0/0

    if Rio1(ip,1)==0;

    Rio1(ip,1)=0.000001;

end
end

Rio=zeros(m,m);
for t=1:m
      Rio(t,t)=Rio1(t);

  end

     Ryo1=(maxm')-y1(:,i) ;


   for op=1:k                              % To avoid 0/0

    if Ryo1(op,1)==0;

     Ryo1(op,1)=0.000001;
end
end
```

```
      Ryo=zeros(k,k);

  for kp=1:k
      Ryo(kp,kp)=Ryo1(kp);
  end


% Objective function is

f=[(-1/(m+k))*ones(1,m+k)  zeros(1,n) zeros(1,pi+pr) zeros(1,pi+pr)]
;
%Number of inputs,   Outputs,      lambdas   integer slacks integer
targets

xval1=[zeros(m,m+k) x1 zeros(m,pi+pr) eye(m,pi)*-1 zeros(m,pr)];


xval2=[Rio zeros(m,k+n) eye(m,m) zeros(m,pr) eye(m,pi) zeros(m,pr)] ;


yval1=[zeros(pi,m+k) yint*-1 zeros(pi,m+pr+pi) eye(pi)];

yval2=[zeros(pi,m) Ryo1(1) zeros(pi,pr+n+pi) eye(pi) zeros(pi)
eye(pi)*-1];

yvalr=[zeros(pr,m+pi) Ryo1(2) yr*-1 zeros(pr,pi) zeros(pr)
zeros(pr,pi) zeros(pr)];

Aeq=[zeros(1,m+k) ones(1,n) zeros(1,2*(pi+pr))] ;

   Beq=[1] ;


     A=[xval1;xval2;yval1;yval2;yvalr];

     b=[zeros(m,1);x1(:,i);zeros(pi,1);yint(:,i)*-1;yr(:,i)*-1]  ;


  Lower bounds on decision variables are

lb= [zeros(m+k,1);zeros(n+2*(pi+pr),1)];


ub=[ones(m+k,1);inf(n+2*(pi+pr),1)];

M=[m+k+n+pi+pr+1:m+k+n+2*(pi+pr)];

e=2^-24;

[x v s]=IP(f,A,b,Aeq,Beq,lb,ub,M,e);

beta1=round ((v)*10000);
beta=(beta1/10000)*-1

efi=1-beta;
```

```
ee(i)=efi;

xx(:,i)=x;


% Input target is

op_proj=y1*x(m+k+1:m+k+n);    % Projected values of Output variables
  obso=y1(:,i);
  op1(:,i)=op_proj;

vv(i)=efi;

xxx=x(m+k+n+pi+pr+1:m+k+n+pi+pr+pi);

Target_Input(:,i)=xxx;

yyy=x(m+k+n+pi+pr+pi+1:m+k+n+pi+pr+pi+pr);

Target_Output(:,i)=yyy;

ss(i)=s;
end
 eee=rankd(ee)   % Ranking the Decision making units
     sl=1:n ;
%
     RO=[sl' ee' eee']
%

% Efficiency of the DMUs


  fprintf('\n Efficiency ')
  vv'

  fprintf('\n status of the programme ')

  ss'
 %  Inputs target

  fprintf('\n The Input targets are ')

  Target_Input'
  %  Outputs target

  fprintf('\n The Output targets are ')

  Target_Output'

clear all
```

Program-3

```
%%=========================================================%%
%       Program to find efficiency through IEM

%%=========================================================%%

iput=xlsread('eg','iput');
oput=xlsread('eg','oput');


x1=iput';
y1=oput';


  n=size(x1,2); % Number of columns in the input matrix
  m=size(x1,1) ;% Number of rows in the input matrix
  k=size(y1,1) ;% Number of rows in the output matrix


maxm=max(y1');
minm=min(x1');

  if n ~= size(y1,2);
      disp('Invalid number of DMU at output Matrix')
  end

  if size(x1,2)~=size(y1,2);
      disp('Invalid number of DMU entered at either input matrix or
output Matrix')
  end


% Inputs are

  for i=1:n

     fprintf('\n DMU number %d\n\n',i)

Rio1=(x1(:,i)-(minm'));

for ip=1:m                          % To avoid 0/0

    if Rio1(ip,1)==0;

    Rio1(ip,1)=0.0001;

end
end

Rio=zeros(m,m);
for t=1:m
      Rio(t,t)=Rio1(t);

  end
```

```
      Ryo1=(maxm')-y1(:,i) ;


    for op=1:k                                  % To avoid 0/0

     if Ryo1(op,1)==0;

      Ryo1(op,1)=0.0001;

end
end

       Ryo=zeros(k,k);

   for p=1:k
       Ryo(p,p)=Ryo1(p);
   end

    % Objective function is

   f=[1 zeros(1,n) ((1/(m))*ones(1,m))*-1 zeros(1,k)] ; % Equal weights


   xval=[x1(:,i)*-1 x1 Rio zeros(m,k)];


     yval=[y1(:,i) y1*-1 zeros(k,m) Ryo];

     betas=[ones(m+k,1)*-1 zeros(m+k,n) eye(m+k,m+k)];

     den=[1 zeros(1,n+m) ones(1,k)*(1/k)] ;

    lambda=[-1 ones(1,n) zeros(1,m+k)];

    A=[yval;xval;betas];

    Aeq=[den;lambda];


    b=[zeros(2*(k+m),1)] ;


     beq=[1;0] ;

% Lower bounds on decision variables are

lb=[zeros(1,1);zeros(n,1); zeros(k+m,1)];          %lower bounds

%    ub=[inf;ones(n+k+m,1)];
options =
optimset('LargeScale','on','MaxIter',1000000,'Display','iter');

[w,fval,EXITFLAG,options]=linprog(f,A,b,Aeq,beq,lb) ;
```

```
% fprintf('In-Efficiency measure beta = %1.4f\n\n', (fval(1)));

  e(i)=fval;

  ww(:,i)=w;

w1=w/w(1);
www(:,i)=w1;

ip_proj=x1*w1(2:n+1);     % Projected values of input variables

  ip1(:,i)=ip_proj;


  op_proj=y1*w1(2:n+1);    % Projected values of output variables

 op1(:,i)=op_proj;

betai=w1(n+2:n+m+1);


 bbx=betai'*Rio;
 ip_pro=x1(:,i)-bbx';

 ip2(:,i)=ip_pro;

 betar=w1(n+m+2:n+m+k+1);
 bby=betar'*Ryo;

 oppro=y1(:,i)+bby';

 op2(:,i)=oppro;

   end
```

Program-4

```
clear all

%%=======================================%%
% Modified Directional Distance Formulation of DEA to Assess
Bankruptcy

%%=======================================%%

data=xlsread('datalimit','second');

iput=data(:,1:3);
oput=data(:,4:10);
% %

x1=iput';
y1=oput';
```

```
  n=size(x1,2); % Number of columns in the input matrix
  m=size(x1,1) ;% Number of rows in the input matrix
  k=size(y1,1) ;% Number of rows in the output matrix
%

minm=min(y1');
maxm=max(x1');
%
  if n ~= size(y1,2);
      disp('Invalid number of DMU at output Matrix')
  end
%
  if size(x1,2)~=size(y1,2);
      disp('Invalid number of DMU entered at either input matrix or
output Matrix')
  end

% Inputs are

   for i=1:n

     fprintf('\n DMU number %d\n\n',i)

Rio1=(maxm'-(x1(:,i)));

%Rio1=(x1(:,i))-minm';
for ip=1:m                          % To avoid 0/0

    if Rio1(ip,1)==0;

    Rio1(ip,1)=0.00001;

end
end

Rio=zeros(m,m);
for t=1:m
      Rio(t,t)=Rio1(t);

end

   Ryo1=y1(:,i)-(minm');

     %  Ryo1=maxm'-y1(:,i);

   for op=1:k                           % To avoid 0/0

    if Ryo1(op,1)==0;

     Ryo1(op,1)=0.0001;

end
end


      Ryo=zeros(k,k);
```

```
 for p=1:k
     Ryo(p,p)=Ryo1(p);
 end


  % Objective function is

f=[1 zeros(1,n) ((-1/(m))*ones(1,m)) zeros(1,k)] ; % Equal weight


xval=[x1(:,i) x1*-1 Rio zeros(m,k)];


   yval=[y1(:,i)*-1 y1 zeros(k,m) Ryo];



   den=[1 zeros(1,n+m) ones(1,k)*(1/(k))] ;

  lambda=[-1 ones(1,n) zeros(1,m+k)];

  A=[yval;xval];

  Aeq=[den;lambda];


  b=[zercs((k+m),1)] ;


   beq=[1;0] ;

% Lower bounds on decision variables are

lb=[zeros(1,1);zeros(n,1); zeros(k+m,1)];          %lower bounds

  ub=[inf;ones(n+k+m,1)];
options =
optimset('LargeScale','on','MaxIter',1000000,'Display','iter');

[w,fval,EXITFLAG,options]=linprog(f,A,b,Aeq,beq,lb,ub) ;


 fprintf('In-Efficiency measure beta = %1.4f\n\n', (fval(1)));

  e(i)=fval;

  ww(:,i)=w;

w1=w/w(1);
www(:,i)=w1;


ip_proj=x1*w1(2:n+1);     % Projected values of input variables

ip1(:,i)=ip_proj;
```

```
op_proj=y1*w1(2:n+1);    % Projected values of output variables

op1(:,i)=op_proj;

 betai=w1(n+2:n+m+1);

 bbx=betai'*Rio;
ip_pro=x1(:,i)+bbx';

ip2(:,i)=ip_pro;

betar=w1(n+m+2:n+m+k+1);
bby=betar'*Ryo;

oppro=y1(:,i)-bby';

op2(:,i)=oppro;


  end

  ee=rankd(e) ; % Ranking the Decision making units
     sl=1:n ;
%
     RO=[sl' e' ee']
%
```

Program-5

```
% Ranking Efficient DMUs through single virtual DMU of multiplier CRS
model

NEWMULTI=fopen('NEWMULTI.txt','w') % Open a output file
fprintf('\n Running program for ranking the efficient DMUs \n')
fprintf(NEWMULTI,'\n Running program for ranking the efficient DMUs
based on multiplier CRS model \n')


fprintf('\n Running program for ranking the efficient   DMUs based on
GRJ CRS Multiplier model \n')
fprintf(NEWMULTI,'\n Running program for ranking the efficient   DMUs
based on GRJ CRS Multiplier model  \n')

ip=xlsread('eg','newinput');
op=xlsread('eg','newoutput');


X1=ip';
Y1=op';


r=3
```

```
n=size(X1,2);
m=size(X1,1);
k=size(Y1,1);


   fprintf('\n To find efficiencies of reference DMUs based on
Inefficient DMU   %d\n')
  fprintf(NEWMULTI,'\n To find efficiencies of reference DMUs based
on Inefficient DMU   %d\n')

  for j=n:n          % Number of inefficient DMUs

  fprintf('\n Inefficient DMU number: %d\n',j)
 fprintf(NEWMULTI,'\n Inefficient DMU number: %d\n',j)

y2=Y1(:,j);              % Consider j th DMU as inefficient and take
output of that DMU

f=-1*[y2' zeros(1,m) 1];              % Put outputs of jth DMU in the
objetive function

x2=X1(:,j);            % Take input of jth DMU

Aeq=[zeros(1,k) x2' 0];            %consider Sum of input of jth DMU =
1 i.e equality constraint AX=b Vector A

 beq=[1];                 %consider Sum of input of jth DMU = 1 i.e
equality constraint AX=b,b

 for i=1:r                      % r is the number of efficient DMUs
   % fprintf('DMU number %d\n',i)
   % fprintf(NEWMULTI,'DMU number %d\n',i)

    a=[Y1;X1*-1];                   %Inequality constraint AX < b
matrix A

    a11=a';

    a1=[zeros(1,m+k);a11];       %Adding one rows to the matrix A

    a2=a1(1:i,:);                %Inorder to delete the reference DMU
from the constraint take first r-1 rows
    a3=a1(i+2:n+1,:);            % Consider last r+1 to n rows

    a4=[a2;a3];
     A1=a4(2:n,:) ;               % Delete the first row, inorder to
take n-1 constraints in the AX < b matrix A
   A=[A1 ones(n-1,1)];
     b= zeros(n-1,1);            %column vector b  AX < b
    lb=[zeros(1,m+k)];           % Lower bound of the variables


    [w,fval] =linprog(f,A,b,Aeq,beq,lb); %Linear program function

 fprintf('Results of the computations are \n\n')
```

```
 fprintf(NEWMULTI,'Results of the computations are \n\n')

 fprintf('Efficiency measure Theta = %1.4f\n\n',abs(fval(1)));
 fprintf(NEWMULTI,'Efficiency measure Theta =
%1.4f\n\n',abs(fval(1))) ;  % Efficiency

 for l=1:k
     fprintf('output weight is=%8.10f\n',  w(l));
     fprintf(NEWMULTI,'output weight is=%8.10f\n',  w(l));    %Outputs
weight
 end


 for p=k+1:k+m
  fprintf('input weight is=%8.10f\n', w(p));
     fprintf(NEWMULTI,'input weight is=%8.10f\n',  w(p));
Inputs weight
 end

         efi=round(-1*(fval)*10000)/10000  % Rounding the efficiency
to four digit

         e1(i)=efi;


 end

  ee=e1;
 end



 ae=ee;
eee=rankd(ae) ;   % Ranking the Decision making units
    sl=1:r;
   RO=[sl' ae' eee'];
   fprintf('Ranking of decision making units \n\n')


      fprintf(NEWMULTI,'Ranking of decision making units\n\n')

      fprintf('DMU No\tEfficiency\tRank \n \n')

      fprintf(NEWMULTI, 'DMU No\tEfficiency\tRank  \n')

      fprintf('%3.0f  \t%1.4f   \t%2.0f\n',RO')
         fprintf(NEWMULTI,'%3.0f   \t%1.4f     \t%2.0f\n',RO')


    fclose(NEWMULTI);
        fprintf('\n\n Output written to NEWMULTI.txt');
```

Program-6

```
clear all

%%=========================================================================%%
%  Ranking efficient DMUS  through changing reference technology
%%=========================================================================%%

 r=9
%
ip=xlsread('eg','iprank');
op=xlsread('eg','oprank');

x1=ip';
y1=op';

%
maxm=max(y1')   % Find the maximum value of output variable
minm=min(x1')   % Find the minimum value of the input variable

 n=size(x1,2); % Number of columns in the input matrix
  m=size(x1,1) ; % Number of rows in the input matrix
  k=size(y1,1) ; % Number of rows in the output matrix
  if n ~= size(y1,2);
      disp('Invalid number of DMU at output Matrix')
  end
%
  if size(x1,2)~=size(y1,2);
      disp('Invalid number of DMU entered at either input matrix or
output Matrix')
  end

% Objective function is


  for j=r+1:n
      % Objective function is

f=[1/k*ones(1,k)*-1 zeros(1,n-1)]            % Equal weightf=[-1
zeros(1,n-1)] ;

      fprintf('\n Inefficient DMU number: %d\n',j)

  Rio1=(x1(:,j)-(minm'))

for ip=1:m

    if Rio1(ip,1)==0

    Rio1(ip,1)=0.0000001
end
end

     Ryo1=(maxm')-y1(:,j)

   for op=1:k
```

```
    if Ryo1(op,1)==0

     Ryo1(op,1)=0.0000001

end
end

 xval=[zeros(m,k) x1]

  Ryo=zeros(k,k)

  for p=1:k
      Ryo(p,p)=Ryo1(p)
  end

    yval=[Ryo y1*(-1)]

      A1=[xval;yval]

for i=1:r
    fprintf('\n The referenced DMU number: %d\n',i)

     % Find range of input directional vector

          %  Find range of output directional vector

     A2=A1(:,1:i+k-1)          % Take first column of A

     A3=A1(:,i+k+1:n+k)          % Take third column of A

     A=[A2 A3]

      b=[x1(:,j);y1(:,j)*-1]
      Aeq=[zeros(1,k) ones(1,n-1)]
    beq=[1]
```

Program-7

```
clear all

%==========================================================
%  ADDM with uncontrollable variables
%==========================================================

iput=xlsread('health','iput');
 oput=xlsread('health','oput');
%%

x1=iput';
y1=oput';

  n=size(x1,2); % Number of columns in the input matrix
  m=size(x1,1) ;% Number of rows in the input matrix
  k=size(y1,1) ;% Number of rows in the output matrix
%
l=1; % Controllable input
```

```
h=3; % Uncontrolable inputs

maxm=max(y1');
minm=min(x1');
%
  if n ~= size(y1,2);
      disp('Invalid number of DMU at output Matrix')
  end
%
  if size(x1,2)~=size(y1,2);
      disp('Invalid number of DMU entered at either input matrix or
output Matrix')
  end

% Inputs are

  for i=1:n

     fprintf('\n DMU number %d\n\n',i)

Rio1=(x1(:,i)-(minm'));

for ip=1:m                          % To avoid 0/0

    if Rio1(ip,1)==0;

    Rio1(ip,1)=0.01;

end
end

Rio=zeros(m,m);
for t=1:m
      Rio(t,t)=Rio1(t);

  end

     Ryo1=(maxm')-y1(:,i) ;


   for op=1:k                              % To avoid 0/0

    if Ryo1(op,1)==0;

     Ryo1(op,1)=0.01;

end
end
      Ryo=zeros(k,k);

  for p=1:k
      Ryo(p,p)=Ryo1(p);
  end


   % Objective function is
```

```
 f=[1 zeros(1,n) ((1/(l))*ones(1,l))*-1 zeros(1,h+k)] ; % Equal
weight


 xval=[x1(1:l,i)*-1 x1(1:l,:) Rio(1:l,:) zeros(l,k)];

   xucn=[x1(l+1:m,i)*-1 x1(l+1:m,:) zeros(h,l+h+k)]   ;

    yval=[y1(:,i) y1*-1 zeros(k,m) Ryo];

%     betas=[ones(m+k,1)*-1 zeros(m+k,n) eye(m+k,m+k)];

    den=[1 zeros(1,n+m) ones(1,k)*(1/(k))] ;

   lambda=[-1 ones(1,n) zeros(1,m+k)];

   A=[yval;xval;xucn];

   Aeq=[den;lambda];


   b=[zeros((l+h+k),1)] ;

    beq=[1;0] ;

% Lower bounds on decision variables are

lb=[zeros(1,1);zeros(n,1); zeros(l+h+k,1)];          %lower bounds

%   ub=[inf;ones(n+k+m,1)];
options =
optimset('LargeScale','on','MaxIter',1000000,'Display','iter');

[w,fval,EXITFLAG,options]=linprog(f,A,b,Aeq,beq,lb) ;


% fprintf('In-Efficiency measure beta = %1.4f\n\n', (fval(1)));

  e(i)=fval;

  ww(:,i)=w;

wl=w/w(1);
www(:,i)=wl;

 ip_proj=x1(1,:)*wl(2:n+1);     % Projected values of input variables

ip1(:,i)=ip_proj;

op_proj=y1*wl(2:n+1);    % Projected values of output variables

op1(:,i)=op_proj;

 betai=wl(n+2:n+m+1);

bbx=betai'*Rio;
```

```
ip_pro=x1(:,i)-bbx';

ip2(:,i)=ip_pro;

betar=w1(n+m+2:n+m+k+1);
bby=betar'*Ryo;

oppro=y1(:,i)+bby';

op2(:,i)=oppro;

  end
```

Program-8

```
clear all

%=========================================%
    Program to find HDI
%=========================================%

iput=xlsread('ha ','iput');
 oput=xlsread('ha ','oput')


x1=iput';
y1=oput';

  n=size(x1,2);   Number of columns in the input matrix
  m=size(x1,1) ;  Number of rows in the input matrix
  k=size(y1,1) ;  Number of rows in the output matrix


 maxm=max(y1');
 minm=min(x1');

 maxm=[ 73.9000   90.9000  100.0000];
 minm=[-4.0122];

  if n ~= size(y1,2);
      disp('Invalid number of DMU at output Matrix')
  end

  if size(x1,2)~=size(y1,2);
      disp('Invalid number of DMU entered at either input matrix or
output Matrix')
  end



% Inputs are

  for i=1:n

     fprintf('\n DMU number %d\n\n',i)

Rio1=(x1(:,i)-(minm'));
```

```
for ip=1:m                          % To avoid 0/0

    if Rio1(ip,1)==0;

    Rio1(ip,1)=0.01;

end
end

Rio=zeros(m,m);
for t=1:m
      Rio(t,t)=Rio1(t);

  end

     Ryo1=(maxm')-y1(:,i) ;


   for op=1:k                              % To avoid 0/0

    if Ryo1(op,1)==0;

     Ryo1(op,1)=0.01;

end
end

      Ryo=zeros(k,k);

  for p=1:k
      Ryo(p,p)=Ryo1(p);
  end

    % Objective function is

% f=[-1 zeros(1,n) zeros(1,m) ((1/(k))*ones(1,k))*-1] ; % Equal
weight

 f=[-1 zeros(1,n) zeros(1,m) -.3333 -.2222 -.1112] ; % Unequal
weightage

 xval=[x1(:,i)*-1 x1 Rio zeros(m,k)];


    yval=[y1(:,i) y1*-1 zeros(k,m) Ryo];

    betas=[ones(m+k,1)*-1 zeros(m+k,n) eye(m+k,m+k)];

    %den=[1 zeros(1,n) ones(1,m)*(1/(m))*-1 zeros(1,k)] ;

    den=[1 zeros(1,n) ones(1,m)*-.3333 zeros(1,k)] ;% weightage


   lambda=[-1 ones(1,n) zeros(1,m+k)];
```

```
   A=[yval;xval;betas];

   Aeq=[den;lambda];


   b=[zeros(2*(k+m),1)] ;

    beq=[1;0] ;

  Lower bounds on decision variables are

lb=[zeros(1,1);zeros(n,1); zeros(k+m,1)];          %lower bounds

%   ub=[inf;ones(n+k+m,1)];
options =
optimset('LargeScale','on','MaxIter',1000000,'Display','iter');

[w,fval,EXITFLAG,options]=linprog(f,A,b,Aeq,beq,lb) ;

 % fprintf('In-Efficiency measure beta = %1.4f\n\n', (fval(1)));
  fval1=round(-10000*fval)/10000;
  e1(i)=fval1;
  e(i)=1/fval1;
   ww(:,i)=w;

w1=w/w(1);
www(:,i)=w1;

ip_proj=x1*w1(2:n+1);     % Projected values of input variables

  ip1(:,i)=ip_proj;

   op_proj=y1*w1(2:n+1);    % Projected values of output variables

 op1(:,i)=op_proj;

betai=w1(n+2:n+m+1);

 bbx=betai'*Rio;
 ip_pro=x1(:,i)-bbx';

 ip2(:,i)=ip_pro;

 betar=w1(n+m+2:n+m+k+1);
 bby=betar'*Ryo;

 oppro=y1(:,i)+bby';

 op2(:,i)=oppro;

   end

  ee=rankd(e)   % Ranking the Decision making units
     sl=1:n ;
%
     RO=[sl' e' ee'];
```