Contents


```
clear all;
close all;
clc;
warning('off')
```

This code is to compute the primal game LP of player 1. This code will

give the game value, player 1's optimal strategy and the initial vector

payoff over player 2's state

Initialization

```
P{1,1}=[.8 .1 .1; .1 .4 .5; .2 .7 .1 ]; %Player 1's Payoff matrix if a=1 and b=1
P{1,2}=[.4 .5 .1; .2 .3 .5; .4 .4 .2 ]; %Player 1's Payoff matrix if a=1 and b=2
P{2,1}=[.2 .2 .6; .5 .2 .3; .2 .2 .6 ]; %Player 1's Payoff matrix if a=2 and b=1
P{2,2}=[.3 .3 .4; .1 .8 .1; .1 .1 .8 ]; %Player 1's Payoff matrix if a=2 and b=2
save P.mat;
Q{1,1}=[.8 .2;.5 .5]; %Player 2's Payoff matrix if a=1 and b=1
Q{1,2}=[.2 .8;.1 .9]; %Player 2's Payoff matrix if a=1 and b=2
Q{2,1}=[.6 .4;.5 .5]; %Player 2's Payoff matrix if a=2 and b=1
Q{2,2}=[.7 .3;.1 .9]; %Player 2's Payoff matrix if a=2 and b=2
save Q.mat;
T=2; %Number of stages in a game
A=2; %Number of player 1's actions
B=2; %Number of player 2's actions
k=3; %Number of states of player 1
l=2; %Number of states of player 2
lm=0.3; %discounted value
p=[0.5 0.3 0.2]; %player 1's initial probability for state
q=[0.5 0.5];      %player 2's initial probability for state
load M.mat;       %payoff matrix
G=M;
```

```
%information set of player 1 and 2
[is1,n_is1]=info_I(T,A,B,k);
[is2,n_is2]=info_J(T,A,B,l);
```

Constraint \sum_{a_t} R_{I_t}(a_t)=P_{a_{t-1},b_{t-1}}(k_{t-1},k_t)R_{I_{t-1}}(a_t-1);

```
%\forall t=1,...n, \forall \mathcal{I}_t
%Creating the equation as matrix multiplication [Aeq]*[variable]=[beq]. In
%this equation the variable is R_{I_t}

Aeq=zeros(sum(n_is1),sum(n_is1)*A);
beq=zeros(sum(n_is1),1);

row_index=0;
for t=1:T
    for i=1:length(is1{t})
        row_index=row_index+1;
        for a=1:A  %As there is summation a_t in R_{I_{t}}
            I=is1{t}(i,:); %for selecting each row from the information set of player 1
            [col_index_RIt] = RIt_col_index_P1(t,I,A,B,k,a,n_is1); %finding the column ind
ex for that particular information set row
            Aeq(row_index,col_index_RIt)=1;
        end
        kt=I(end);
        if t==1
           beq(row_index,1)=p(kt); %As at t=1, P_{a_0,b_0}(k_0,k_1)=1 and R_{I_0}(a_0)=Pr(
k1)
        else
            Ipre=is1{t}(i,1:(length(is1{t}(i,:))-3));
            aprev=is1{t}(i,(length(is1{t}(i,:))-2));
            bprev=is1{t}(i,(length(is1{t}(i,:))-1));
            [col_index_RItprev] = RIt_col_index_P1(t-1,Ipre,A,B,k,aprev,n_is1);
            ktpre=I(end-3);
            Aeq(row_index,col_index_RItprev)=-P{aprev,bprev}(ktpre,kt);
        end
    end
end
```

The inequality contraint

```
%Creating the equation as matrix multiplication [Ain]*[variables]=[bin]. In
%this equation the variables are R_{I_t},(U_{J_t},U_{J_{t+1}})
[kset,n_kset] = Kset(T,k); %to get player 1's state information only

Ain=zeros(sum(n_is2)*B,(sum(n_is1)*A+sum(n_is2)));
bin=zeros(sum(n_is2)*B,1);

row_index=0;
for t=1:T
    for j=1: length (is2{t}) %for different J_t and different b_t there will be different
row
        for b=1:B
            row_index=row_index+1;
            for kn=1:length(kset{t})
```

```
                for a=1:A
                    [I] = construct_Is(j,kn,t,kset,is2);
                    [col_index] = RIt_col_index_P1(t,I,A,B,k,a,n_is1);
                    Ain(row_index,col_index)=lm^(t-1)*G{I(3*t-2),(is2{t}(j,(3*t-2)))}(a,b)
; %R_{I_t} coefficient
                end
            end
            %To find the term for U_{J_{t+1}}
            if t<T  %As for U_{J_{N+1}}=0
                for a1=1:A
                    for lplus=1:l
                        Jplus=[is2{t}(j,:) a1 b lplus];
                        [col_index_Jplus] = J_col_index(n_is1,A,B,l,Jplus,n_is2,t+1);
                         Ain(row_index,col_index_Jplus)=Q{a1,b}(is2{t}(j,end),lplus);%The
coefficient for U_{J_{t+1}}
                    end
                end
            end
            %To find the term for U_{J_{t}}
            Jpre=is2{t}(j,:);
            [col_index_Jpre] = J_col_index(n_is1,A,B,l,Jpre,n_is2,t);
            Ain(row_index,col_index_Jpre)=-1;
        end
    end
end
```

Construct the objective function: \max_{R}max_{U} \sum_{I} q(I)*U_{\mathcal{J}_1}

```
f=[zeros(1,sum(n_is1)*A) q zeros(1,sum(n_is2(2:end)))]; %zeros(1,sum(n_is1)*A) is for R_{I
_t} as there is no R_{I_t} in the objective function. zeros(1,sum(n_is2(2:end))) is for U_
{J_{2:T}} as in objective function there is only U_{\mathcal{J}_{1}}

%Rearrange every coefficient to use in linprog
Ain1=-Ain; %As linprog use <=
bin1=-bin; %As linprog use <=
f1=-[zeros(1,size(Ain1,2)-size(f,2)) f ]; %as linprog works for only minimize obj function
 thats why -ve. To ensure that matrix of objective function and Ain,Aeq are of same size w
e add zeroes
Aeq1=padarray(Aeq,[0, (size(Ain1,2)-size(Aeq,2))],0,'post'); % to use linprog Ain and Aeq
must have same column number as column number indicates variables. Both equation must have
 same variables
lb=[zeros(sum(n_is1)*A,1);-Inf((size(Ain1,1)-sum(n_is1)*A),1)];
ub=+Inf;

options = optimoptions('linprog','Display','none');
[x,v1]=linprog(f1,Ain1,bin1,Aeq1,beq,lb,ub,options); %x=[R_{I_t} U_{J_t}] and v1 is the ga
me value
```

Game value

```
v1
```

```
v1 =
```

```
-103.6310
```

Finding the optimal strategy from the realization plan we got from previous linprog result

```
%sigma=R_{I_t}/(P_{a_{t-1},b_{t-1}}(k_{t-1},k_t)*RI_{t-1})

sigma=zeros(1,sum(n_is1));
sigma_col=0;
for t=1:T
    for i=1:length(is1{t})
        I=is1{t}(i,:); %for selecting each row from the information set of player 1
        sigma_col=sigma_col+1;
        for a=1:A  %As there is summation a_t in R_{I_{t}}
            [row_index_RIt] = RIt_col_index_P1(t,I,A,B,k,a,n_is1); %finding the column ind
ex for that particular information set row. The number of column index in RIt is the numbe
r of row index in optimal realization plan x
            RIt=x(row_index_RIt,1);
            kt=I(end);
            if t==1
               RItprev=p(kt);
               pvalue=1;
            else
                Ipre=is1{t}(i,1:(length(is1{t}(i,:))-3));
                aprev=is1{t}(i,(length(is1{t}(i,:))-2));
                bprev=is1{t}(i,(length(is1{t}(i,:))-1));
                [row_index_RItprev] = RIt_col_index_P1(t-1,Ipre,A,B,k,aprev,n_is1);
                ktpre=I(end-3);
                RItprev=x(row_index_RItprev,1);
                pvalue=P{aprev,bprev}(ktpre,kt);
            end
            sigma(a,sigma_col)=RIt/(RItprev*pvalue);
        end
    end
end
```

sigma is player 1's optimal strategy

```
sigma %sigma(1st row) for a=1 and sigma(2nd row) for a=2. Each column indicates each infor
mation set at different stages. For example. 5th column is for T=2 (k_1=1, a_1=1, b_1=1, k
_2=2)
```

```
sigma =

  Columns 1 through 7

    1.0000    0.0000    1.0000         0         0         0         0
         0    1.0000         0    1.0000    1.0000    1.0000    1.0000

  Columns 8 through 14

         0         0       NaN       NaN       NaN       NaN       NaN
```

```
    1.0000    1.0000       NaN       NaN       NaN       NaN       NaN

  Columns 15 through 21

       NaN         0         0         0         0         0         0
       NaN         0         0         0         0         0         0

  Columns 22 through 28

         0    0.2774         0         0    0.4392         0         0
    1.0000    0.7226    1.0000    1.0000    0.5608    1.0000    1.0000

  Columns 29 through 35

    0.4899         0         0    0.3135         0       NaN       NaN
    0.5101    1.0000    1.0000    0.6865    1.0000       NaN       NaN

  Columns 36 through 39

       NaN       NaN       NaN       NaN
       NaN       NaN       NaN       NaN
```

to find ontimal nu

```
%to find UJ1
for l_present=1:l
    t=1;
    info=l_present;
    [col_index_UJ1] = J_col_index(n_is1,A,B,l,info,n_is2,t);
    UJ1=x(col_index_UJ1,1);
    nu(l_present)=[UJ1];
end
```

nu is the initial vector payoff over player 2's state

```
nu=-nu %as nu=-Z_{I_1}. nu[1] is for nu(l=1) and nu[2] is for nu(l=2)
```

```
nu =

  -89.5061 -117.7559
```