William Wijaya
3035992465

Lab 4: Stochastic Dynamic Programming

- The optimal decision rules for problem 1&2

Problem 1

Condition									
0	1	2	3	4	5	6	7	8	9
100060	100060	100060	60	43.661345	13.874712	3.108105	0.44724	0.115974	0.013096
100060	100060	100060	60	39.1059	9.2814	1.6749	0.1418	0.0256	0
100060	100060	100060	60	34.88	4.53	0.64	0	0	0
100060	100060	100060	34.5	32	0	0	0	0	0
100000	100000	100000	0	0	0	0	0	0	0

Problem 2

Condition									
0	1	2	3	4	5	6	7	8	9
100080.3444	100080.3444	100080.3444	80.3444	80.3444	58.3924	42.1948	29.7673	24.1497	21.4151

- The optimal costs for problem 1&2 in the following format

Problem 1

Condition									
0	1	2	3	4	5	6	7	8	9
'replacement'	'replacement'	'replacement'	'replacement'	'maintenance'	'do nothing'	'do nothing'	'do nothing'	'do nothing'	'do nothing'
'replacement'	'replacement'	'replacement'	'replacement'	'maintenance'	'do nothing'	'do nothing'	'do nothing'	'do nothing'	'do nothing'
'replacement'	'replacement'	'replacement'	'replacement'	'maintenance'	'do nothing'	'do nothing'	'do nothing'	'do nothing'	'do nothing'
'replacement'	'replacement'	'replacement'	'maintenance'	'maintenance'	'do nothing'	'do nothing'	'do nothing'	'do nothing'	'do nothing'
'do nothing'	'do nothing'	'do nothing'	'do nothing'	'do nothing'	'do nothing'	'do nothing'	'do nothing'	'do nothing'	'do nothing'

Problem 2

Condition									
0	1	2	3	4	5	6	7	8	9
'replacement'	'replacement'	'replacement'	'replacement'	'replacement'	'do nothing'	'do nothing'	'do nothing'	'maintenance'	'do nothing'

viteration.m

```matlab
%solution should be a matrix of structures with elements indexed by year
%and condition, e.g.solution(year,condition+1). Each structure in the matrix should
%have a value field, and a decision field. For example,
%solution(3,3).value = 4, solution(3,3).decision = 'do nothing'.
%When you load transitionMatrices.mat it will give you an array of structures with fields matrix and
%decision.
% In this lab, we assign transitionMatrices by
% load transitionMatrices
% cxu is a function to compute the cost function.
% Thus you might call the function below by typing solution = viteration ('transitionMatrices',@cxu,5)
function solution = viteration(tmatrixFilename,cxuHandle,horizon)
% Your code starts here

load(tmatrixFilename) % Load tmatrixFilename
value=zeros(horizon,10); %Empty value matrix (matrix of zeros 5x10)
decision=repmat({''},horizon,10); %Empty decision matrix 5x10

for i=1:horizon % for loop for year 1 until 5 (backward)
    for j=0:9 %for loop condition 0-9
        if i==1 % Assume V(5,j)=0
        % problem when the action is 'replacement'
        minp_1=cxu(j,'replacement');
        % problem when the action is 'maintenance'
        minp_2=cxu(j,'do nothing');
        % problem when the action is 'do nothing'
        minp_3=cxu(j,'maintenance');
        else
        % problem when the action is 'replacement'
        minp_1=cxu(j,'replacement')+transitionMatrices(1).matrix(j+1,:)*value(end+2-i,:)';
        % problem when the action is 'maintenance'
        minp_2=cxu(j,'do nothing')+transitionMatrices(2).matrix(j+1,:)*value(end+2-i,:)' ;
        % problem when the action is 'do nothing'
        minp_3=cxu(j,'maintenance')+transitionMatrices(3).matrix(j+1,:)*value(end+2-i,:)' ;
        end
        % Formulating Bellman equation and assigning the solution to value
        % matrix
        [value(end+1-i,j+1),idx]=min([minp_1,minp_2,minp_3]); %V(i,j)=min(...)
        if idx==1
```

```
            decision(end+1-i,j+1)={'replacement'};
        elseif idx==2
            decision(end+1-i,j+1)={'do nothing'};
        elseif idx==3
            decision(end+1-i,j+1)={'maintenance'};
        end
    end
end

%Final answer
solution.value=value;
solution.decision=decision;



% You code ends here
end
```

viterationInf.m

```
%solution should be an array of structures with elements indexed by year,
%e.g., solution(state). Each structure in the matrix should
%have a value field, and a decision field. For example,
%solution(3).value = 4, solution(3).decision = 'do nothing'.
%When you load transitionMatrices.mat it will give you an array of structures with fields matrix and
%decision.
% In this lab, we assign transitionMatrices by
% 'load transitionMatrices'
% cxu is a function to compute the cost function.
% Thus you might call the function below by typing solution = viterationInf
('transitionMatrices',@cxu,0.001)

function solution = viterationInf (tmatrixFilename, cxuHandle, epsilon)
% In the case of infinite horizon, solution is not a function over time,
% it is only a function of state, i.e.
% solution(state).value, solution(state).decision

%You code starts here

%Fill epsilon=10^-8
load(tmatrixFilename) %Load tmatrixFile name
value=zeros(1,10); %Empty value matrix (matrix of zeros nx10, it acts like a history.
decision=repmat({''},1,10); %Empty decision matrix nx10
difference=inf; %Set difference=inf as a starting point
diffmat=repmat(difference,3,10); %Difference matrix
epsmat=repmat(epsilon,1,10); %Epsilon matrix
alpha=0.95; %Discount factor

i=1;
while all(abs(diffmat(i+1,:)-diffmat(i,:))<epsmat(1,:))==0
    for j=0:9 %condition 0 until 9
        if i==1
        minp_1=cxu(j,'replacement');
        % problem when the action is 'maintenance'
        minp_2=cxu(j,'do nothing');
        % problem when the action is 'do nothing'
        minp_3=cxu(j,'maintenance');
```

```
        else
        % problem when the action is 'replacement'
        minp_1=cxu(j,'replacement')+alpha*transitionMatrices(1).matrix(j+1,:)*value(i-1,:)';
        % problem when the action is 'maintenance'
        minp_2=cxu(j,'do nothing')+alpha*transitionMatrices(2).matrix(j+1,:)*value(i-1,:)' ;
        % problem when the action is 'do nothing'
        minp_3=cxu(j,'maintenance')+alpha*transitionMatrices(3).matrix(j+1,:)*value(i-1,:)';
        end

        % Formulating Bellman equation and assigning the solution to value
        % matrix
        [value(i,j+1),idx]=min([minp_1,minp_2,minp_3]); %V(i,j)=min(...)

        if idx==1
            decision(i,j+1)={'replacement'};
        elseif idx==2
            decision(i,j+1)={'do nothing'};
        elseif idx==3
            decision(i,j+1)={'maintenance'};
        end

        if i>=2
            %Stores the difference between each entry of v(i+1) and v(i),
            %it acts like a history of differences.
            diffmat(i+2,:)=value(i,:)-value(i-1,:);
        elseif i==1
            diffmat(i+1,:)=inf;
        end
    end
    i=i+1;
end

solution.value=value(end,:);
solution.decision=decision(end,:);
%You code ends here
end
```