Question 1:

A landlord may be trying to maintain a property. Let the state space reflect the condition of the house on a scale of -5 to 5, with -5 representing a severely deteriorated condition and 5 corresponding to an excellent condition. At each state $s \in \{-4, \dots, 4\}$ the landlord may choose to do maintenance (a=-1) or sleep/rest (a=1). If at the boundary state -5, large-scale repairs must be conducted (a=3), and if at the boundary state 5, the landlord has a night on the town (a=-3). The landlord's cost per week is defined as the reward for being in a state and selecting an action which is given by $r(s, a) = s \cdot a$. An action is selected each week and the horizon is infinite, with the objective function and transition probabilities as specified in the formulation of the assignment, λ is a discounting factor.

Question 2:

It's expected that optimal policies will generally try to avoid the boundary states, though the degree to which this is expressed will depend on the value of λ. For small λ the optimal policy will select action a=-1 when the state is negative, action a=1 when the state is positive, and the choice at 0 being irrelevant (as r(0,a)=0) (excluding boundaries as they're fixed). Trying to avoid the boundaries isn't hugely concerning as λ rapidly goes to 0, hence the short-term reward is maximized. For example, λ≈0 should yield an optimal policy $\{3, -1, -1, -1, -1, \pm 1, 1, 1, 1, 1, -3\}$. Conversely, if λ is large, the optimal policy will place a lot of emphasis on trying to avoid the boundary states as λ goes to 0 slowly. This translates to choosing action a=1 when the state is negative and action a=-1 when the state is positive, the choice at state 0 being irrelevant (as r(0,a)=0) (excluding boundaries as they're fixed). For example, λ≈1 should yield an optimal policy $\{3, 1, 1, 1, 1, \pm 1, -1, -1, -1, -1, -3\}$.

Questions 3 to 5: code attached as appendix

For the parameters : $\begin{cases} \lambda = 0.5 \\ \varepsilon = 0.0001 \end{cases}$

All approaches yielded:

V = (-21.2388, -3.7164, 1.9850, 3.0292, 2.2783, 1.1392, 2.2783, 3.0292, 1.9850, -3.7164, -21.2388)

Policy = (3, -1, -1, -1, -1, -1, 1, 1, 1, 1, -3)

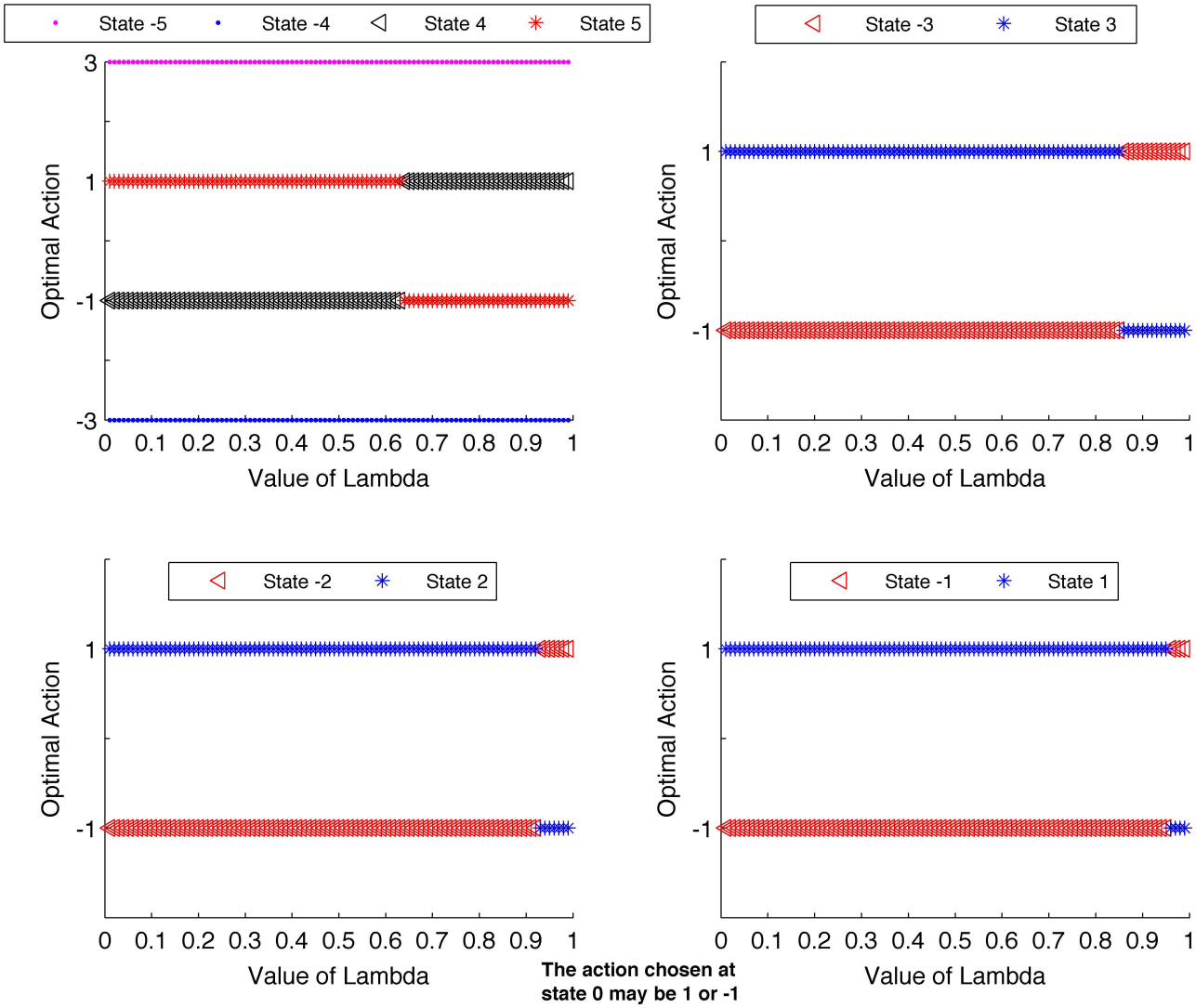
In addition, for the same V, the brute force method also yielded the policy:

Policy = (3, -1, -1, -1, -1, 1, 1, 1, 1, 1, -3)

Maintaining the same ε, all approaches achieved consistent (same as each other) results for different values of λ, with the brute force method yielding the same behavior in returning 2 policies. Increasing ε resulted in less accurate approximations of V (value iteration).

Question 7:

Figure 1: Optimal Policy for Different Values of λ



As can be seen in figure 1, the postulations of question 2 hold. That is, when λ≈0, hitting the boundaries isn't hugely important as action a=-1 is chosen for state -4 and a=1 for state 4. But as λ increases, the policy tries to avoid the boundaries, that is near state -5, action a=1 is selected, and near 5 action a=-1 is chosen. The extent to which such behavior is exhibited in the optimal policy depends of the proximity of λ to 1 (closer implies greater expression and vise versa).

Appendix:

Code for questions 3 to 5

22/09/14 6:49 PM /Users/macbook/Documents/Jac.../question3.m 1 of 2

```
clear
clc
tic
%% Model Parameters
lambda=0.5;%Discounting Factor
epsilon=0.0001;%level for epsilon-optimal policy (Q4)
States=-5:5;%All states
choice_states=-4:4;%States where an action is chosen
choice_actions=[-1,1];%Set of choosable actions
A_5=-3;%Specified action for state 5
A_neg_5=3;%Specified action for state -5

%% Action Transition Probabilities:
%Create transition probabilities for choosable actions a=(-1,1)
%Index 1 corresponds to state -5
%Index 11 corresponds to state +5
p_neg_1=zeros(length(States),length(States));%a=-1
p_pos_1=zeros(length(States),length(States));%a=1
p_neg_1(1,1:2)=[1/2,1/2];
p_neg_1(11,10:11)=[1/2,1/2];
p_pos_1(1,1:2)=[1/2,1/2];
p_pos_1(11,10:11)=[1/2,1/2];
for s=2:length(States)-1
    p_neg_1(s,s-1)=0.75;
    p_neg_1(s,s+1)=0.25;
    p_pos_1(s,s-1)=0.25;
    p_pos_1(s,s+1)=0.75;
end

%% Generate all possible policies:
Psuedo_States = [0:2^size(choice_states,2)-1]';
Psuedo_Policies = rem(floor(Psuedo_States*...
    pow2(-(size(choice_states,2)-1):0)),2);
Psuedo_Policies(Psuedo_Policies==0)=-1;

%% Rewards Matrix
X=meshgrid(States,0:2^size(choice_states,2)-1);
newcol_1=A_neg_5*ones(size(0:2^size(choice_states,2)-1,2),1);
newcol_2=A_5*ones(size(0:2^size(choice_states,2)-1,2),1);
Policies=[newcol_1 Psuedo_Policies newcol_2];
Rd=X.*Policies;

%% QUESTION 3: BRUTE FORCE
for i=1:2^size(choice_states,2)
    P(1,:)=p_neg_1(1,:);
    P(size(States,2),:)=p_neg_1(size(States,2),:);
    %Create the transition probability matrix specific to the policy i
    for j=1:size(choice_states,2)
        if Psuedo_Policies(i,j)==-1
            P(j+1,:)=p_neg_1(j+1,:);
        else
            P(j+1,:)=p_pos_1(j+1,:);
        end
    end
    %Get the expected total discount reward
    V(i,:)=((eye(size(States,2)) - lambda * P) \ Rd(i,:)')';
    %Calculating V via Gaussian Elimination
end
[Expected_Total_Discount_Reward, Index]=max(V);%Find the optimal policy
Optimal_Policy=Policies(unique(Index),:)%Return the optimal policy
Expected_Total_Discount_Reward%Return the policy value
```

```
%% QUESTION 4: VALUE ITERATION
V_old=zeros(1,size(States,2));%Initialise V
V_new=V_old+epsilon*(1-lambda)/(2*lambda)+1;%Ensures the loop is entered
Q4_d=[A_neg_5 zeros(1,9) A_5];%Initialise action matrix
while norm(V_new-V_old) > epsilon*(1-lambda)/(2*lambda)
    V_old=V_new;
    %Specify Boundaries
    V_new(1)=A_neg_5*(-5)+lambda*sum( p_neg_1(1,:).*V_old );
    V_new(end)=A_5*(5)+lambda*sum( p_neg_1(end,:).*V_old );
    %Loop for intermediate states
    for i=2:size(Policies,2)-1
        a_neg_1= choice_actions(1)*(States(i))+...
            lambda*sum( p_neg_1(i,:).*V_old );
        a_pos_1= choice_actions(2)*(States(i))+...
            lambda*sum( p_pos_1(i,:).*V_old);
        [V_new(i), Q4_d(i)]=max([a_neg_1,a_pos_1]);
    end
    Q4_d(Q4_d==1)=-1;
    Q4_d(Q4_d==2)=1;
end
Q4_d%Return the optimal policy
V_new%Return the policy value

%% QUESTION 5:
Q5_d_old=[A_neg_5 ones(1,9) A_5]; %Initialise policy vectors
Q5_d_new=[A_neg_5 -ones(1,9) A_5];
indicator = 0; %Indicator variable: 1 if Q5_d_old = Q5_d_new, 0 otherwise
while indicator ~= 1
    Q5_d_old=Q5_d_new;
    %Create the transition probability matrix specific to the policy
    for j=2:size(Policies,2)-1
        if Q5_d_old(j)==-1
            P(j,:)=p_neg_1(j,:);
        else
            P(j,:)=p_pos_1(j,:);
        end
    end
    %Policy Evaluation
    Q5_V=((eye(size(States,2)) - lambda * P) \ (States.*Q5_d_old)')';
    %Policy improvement step
    for i=2:size(Policies,2)-1
        Q5_a_neg_1= choice_actions(1)*(States(i))+...
            lambda*sum( p_neg_1(i,:).*Q5_V);
        Q5_a_pos_1= choice_actions(2)*(States(i))+...
            lambda*sum( p_pos_1(i,:).*Q5_V);
        [Value, Q5_d_new(i)]=max([Q5_a_neg_1,Q5_a_pos_1]);
    end
    Q5_d_new(Q5_d_new==1)=-1;
    Q5_d_new(Q5_d_new==2)=1;
    indicator=isequal(Q5_d_old,Q5_d_new);
end
Q5_d_new%Return the optimal policy
Q5_V%Return the policy value
toc
```

Code for question 7:


```
lambda_start=0.01;%Discounting Factor
increments=0.01;
epsilon=0.0001;%level for epsilon-optimal policy (Q4)

for lambda=lambda_start:increments:(1-increments)
    %% Model Parameters
    States=-5:5;%All states
    choice_states=-4:4;%States where an action is chosen
    choice_actions=[-1,1];%Set of choosable actions
    A_5=-3;%Specified action for state 5
    A_neg_5=3;%Specified action for state -5

    %% Action Transition Probabilities:
    %Create transition probabilities for choosable actions a=(-1,1)
    %Index 1 corresponds to state -5
    %Index 11 corresponds to state +5
    p_neg_1=zeros(length(States),length(States));%a=-1
    p_pos_1=zeros(length(States),length(States));%a=1
    p_neg_1(1,1:2)=[1/2,1/2];
    p_neg_1(11,10:11)=[1/2,1/2];
    p_pos_1(1,1:2)=[1/2,1/2];
    p_pos_1(11,10:11)=[1/2,1/2];
    for s=2:length(States)-1
        p_neg_1(s,s-1)=0.75;
        p_neg_1(s,s+1)=0.25;
        p_pos_1(s,s-1)=0.25;
        p_pos_1(s,s+1)=0.75;
    end

    %% Generate all possible policies:
    Psuedo_States = [0:2^size(choice_states,2)-1]';
    Psuedo_Policies = rem(floor(Psuedo_States*pow2(-(size(choice_states,2)-1):0)),2);
    Psuedo_Policies(Psuedo_Policies==0)=-1;

    %% Rewards Matrix
    X=meshgrid(States,0:2^size(choice_states,2)-1);
    newcol_1=A_neg_5*ones(size(0:2^size(choice_states,2)-1,2),1);
    newcol_2=A_5*ones(size(0:2^size(choice_states,2)-1,2),1);
    Policies=[newcol_1 Psuedo_Policies newcol_2];
    Rd=X.*Policies;

    %% QUESTION 3: BRUTE FORCE
    for i=1:2^size(choice_states,2)
        P(1,:)=p_neg_1(1,:);
        P(size(States,2),:)=p_neg_1(size(States,2),:);
        %Create the transition probability matrix specific to the policy i
        for j=1:size(choice_states,2)
            if Psuedo_Policies(i,j)==-1
                P(j+1,:)=p_neg_1(j+1,:);
            else
                P(j+1,:)=p_pos_1(j+1,:);
            end
        end
        %Get the expected total discount reward
        V(i,:)=((eye(size(States,2)) - lambda * P) \ Rd(i,:)')';
        %Calculating V via Gaussian Elimination
    end
    [Expected_Total_Discount_Reward, Index]=max(V);%Find the optimal policy
    Optimal_Policy=Policies(unique(Index),:)%Return the optimal policy
    Expected_Total_Discount_Reward%Return the policy value
```

```
    %% QUESTION 4: VALUE ITERATION
    V_old=zeros(1,size(States,2));%Initialise V
    V_new=V_old+epsilon*(1-lambda)/(2*lambda)+1;%Ensures the loop is entered
    Q4_d=[A_neg_5 zeros(1,9) A_5];%Initialise action matrix
    while norm(V_new-V_old) > epsilon*(1-lambda)/(2*lambda)
        V_old=V_new;
        %Specify Boundaries
        V_new(1)=A_neg_5*(-5)+lambda*sum( p_neg_1(1,:).*V_old );
        V_new(end)=A_5*(5)+lambda*sum( p_neg_1(end,:).*V_old );
        %Loop for intermediate states
        for i=2:size(Policies,2)-1
            a_neg_1= choice_actions(1)*(States(i))+lambda*sum( p_neg_1(i,:).*V_old );
            a_pos_1= choice_actions(2)*(States(i))+lambda*sum( p_pos_1(i,:).*V_old);
            [V_new(i), Q4_d(i)]=max([a_neg_1,a_pos_1]);
        end
        Q4_d(Q4_d==1)=-1;
        Q4_d(Q4_d==2)=1;
    end
    Q4_d%Return the optimal policy
    V_new%Return the policy value

    %% QUESTION 5:
    clear Q5_V
    Q5_d_old=[A_neg_5 ones(1,9) A_5]; %Initialise policy vectors
    Q5_d_new=[A_neg_5 -ones(1,9) A_5];
    indicator = 0; %Indicator variable: 1 if Q5_d_old = Q5_d_new, 0 otherwise
    while indicator ~= 1
        Q5_d_old=Q5_d_new;
        %Create the transition probability matrix specific to the policy
        for j=2:size(Policies,2)-1
            if Q5_d_old(j)==-1
                P(j,:)=p_neg_1(j,:);
            else
                P(j,:)=p_pos_1(j,:);
            end
        end
        %Policy Evaluation
        Q5_V=((eye(size(States,2)) - lambda * P) \ (States.*Q5_d_old)')';
        %Policy improvement step
        for i=2:size(Policies,2)-1
            Q5_a_neg_1= choice_actions(1)*(States(i))+lambda*sum( p_neg_1(i,:).*Q5_V);
            Q5_a_pos_1= choice_actions(2)*(States(i))+lambda*sum( p_pos_1(i,:).*Q5_V);
            [Value, Q5_d_new(i)]=max([Q5_a_neg_1,Q5_a_pos_1]);
        end
        Q5_d_new(Q5_d_new==1)=-1;
        Q5_d_new(Q5_d_new==2)=1;
        indicator=isequal(Q5_d_old,Q5_d_new);
    end
    Q5_d_new%Return the optimal policy
    Q5_V%Return the policy value
end
```