Problem Set 1 – Solution

Programming Camp 2021

Brian Higgins Ciaran Rogers

Question 1: Recursion

We use exactly the same logic as in the Fibonacci example in class: write down the simple step computation and reuse the function to compute everything else. The main difference – whereas in Fibonacci example we were looking at values of previous elements, in dynamic programing we need the values of the patent in the future.

```
function [v, d] = patent_recursion(p,r,realized,k,K,t,T)

if t>T
    v = 0;
    d = 0;
elseif realized == 1
    v = patent_recursion(p,r,1,k,K,t+1,T);
    [v, d] = max([K,K - k + v/(1+r)]);
    d = d-1;
else
    v_r = patent_recursion(p,r,1,k,K,t+1,T);
    v_nr = patent_recursion(p,r,0,k,K,t+1,T);
    [v, d] = max([0,- k + (1/(1+r))*(p*v_r + (1-p)*v_nr)]);
    d = d-1;
end
```

Answer the following questions using the function and any supporting code:

1. Given parameter values from the introduction, and T=20, p = 0.05, r = 0.05, what is the value of an unrealized patent at period 0? What is the value when p = 0.5?

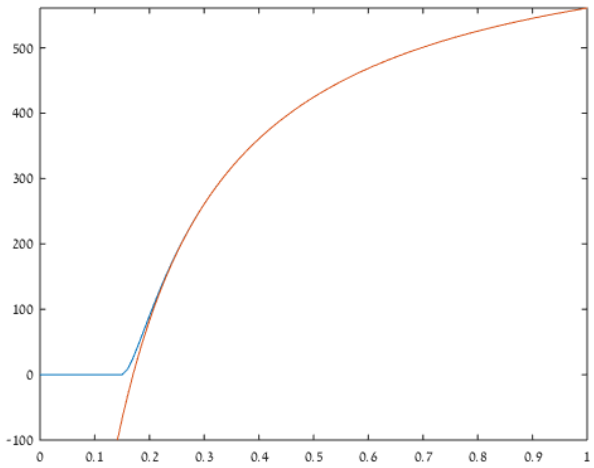
   The value of the patent is 0 with the given parameter values. Why is that? Because the cost of renewing the patent is just too high compared to the expected value of the cash flow that it will provide starting next period. The value of a realized patent at period 1 is $694 (verify!), and the expected value (0.05 times that) is only $34.7 – less than the cost of renewal. If p = 0.5, both the realized and unrealized values of the patent will be higher, making the unrealized value at t=0 equal $424.

2. Plot the value of the patent for p in between 0 and 1. Make sure that your plot shows discontinuities, if they exist. On the same graph, plot the present value of the patent when the renewal strategy is "always renew". What is the source of the difference?

   To plot the value we can use a simple loop that goes through values of p. An example of such loop:

```
N_p = 100;
p_vec = linspace(0.01,1,N_p);
v = zeros(N_p,1);

for i = 1:N_p
    v(i) = patent_recursion(p_vec(i),r,0,k,K,0,T);
end
```

Then we can plot the values of the patent as they vary with p. To plot the "always renew" patent, use the code for the patent value and fix the decision rule (instead of using max) except for the last period. The plot should look like this:



Notice that always renew overlaps with the value for p>0.25 – suggesting that the optimal strategy is to always renew. What is the source of the difference? Having the option not to renew guarantees that the value cannot be less than zero, today and in the future. The direct effect can be seen at very low probabilities, where the value is set to zero, even though the "always renew" has a negative present value. The indirect effect, through the optimal choice in the future, can be seen where the value is positive even though "always renew" has a negative PV.

Question 2: Loops

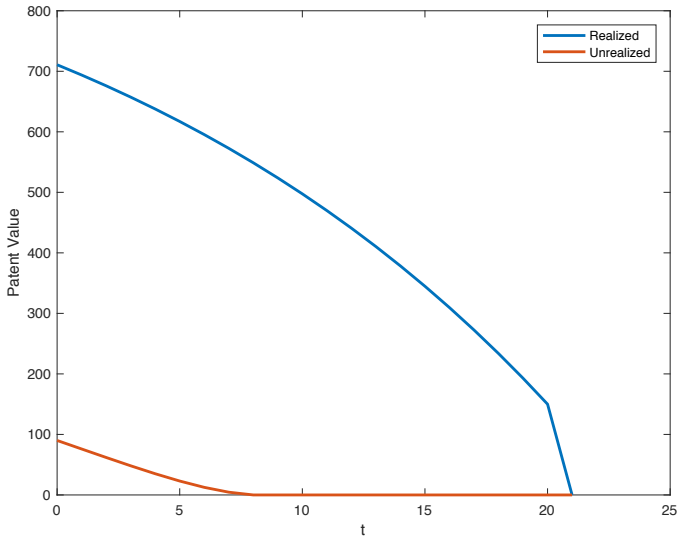
Write a function called patent_value0, that has all the same input arguments except realized and t, and computes the value for unrealized and realized for all periods.

```
function [V, D] = patent_loop(p,r,k,K,T) % No longer any "t" or "realized"

V = zeros(2,T+2);
D = zeros(2,T+1);
p_vec = [1 p]; % This is the difference in probability of renewal across two states
K_vec = [K 0]; % This is the payment conditional on state
for t = T+1:-1:1
    for j = 1:2
        % For me at least, it was crucial to have brackets around second
        % element in order for it to work
        [v, d] = max([K_vec(j);( K_vec(j) - k +...
            (1/(1+r))*[p_vec(j) 1-p_vec(j)]*V(:,t+1))]);
        V(j,t) = v;
        D(j,t) = d-1;
    end
end
```

The main thing to notice: computing the value for different states is just a matter of period payoffs (0 or K) and of the transition probabilities to the realized state (p or 1). That means that any additional states could also be described by their period payoffs and the probability of ending up in other states. Challenge: the nested loop can be replaced with a single vectorized command.

Answer the following questions using the function and any supporting code:

1. Pick interesting parameter values, explain why you picked them and plot a graph of value function against time.
   Consider the same parameter values with p = 0.2. We get the following graph:



2. Can you describe the firm strategy using the results? When does it renew and when not?
   The firm is always renewing when the patent is realized, except in the last period T (patent expires anyway). In unrealized states, the firm renews as long as V>0, that is if the patent has not been realized for the first 12 periods, the firm chooses not to renew.
3. What do you think was the advantage of writing the program as a loop?
   Two advantages come to mind. One is efficiency – the program only computes each value once, as opposed to many times in recursion. Second is that it stores all the iterations, and using the output of the loop we can analyze the whole strategy of the firm, without separately solving for every state and time.

Question 3: Large State Space

We change the problem in the following way. Instead of having two states ("realized" and "unrealized"), consider a patent that generates cash flow $x at period t, and at period t+1 x*q with probability p and x/q with probability 1-p. All the other details stay the same: the patent expires after period T and there is a renewal fee that has to be paid each period, otherwise the patent rights are lost.

Write code that computes the value of the patent across states and across cash flow realization.

```
function [X, T, V, D] = patent_value_Q3(p,q,r,k,x,T,n)

N = T+1+n;
V = zeros(N,T+2);
D = zeros(N,T+1);
payoffs = x*(q.^(0:2:2*(N-1)))'*(q.^-(1:T+1));


for t = T+1:-1:1
    [v, d] = max([payoffs(1:end-1,t), payoffs(1:end-1,t) - ...
        k + (1/(1+r))*[V(2:end,t+1) V(1:end-1,t+1)]*[p;1-p]]');
    V(1:end-1,t) = v;
    D(1:end-1,t) = d-1;
end

% The next line is to get rid of the last row, which is beyond the last
% date and is hepful for the mesh that we will graph
V(:,T+2) = [];
X = payoffs;
T = repmat(0:T,N,1);
```

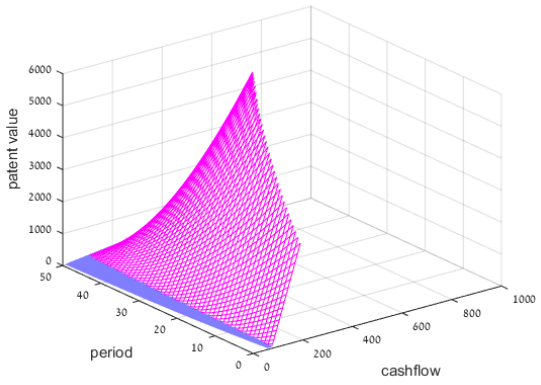
Things to notice: (1) There are t+n states at each period t. This is because x can only change by a fixed fraction, and x*q*(1/q) = x*(1/q)*q = x (2) Precomputing the payoffs at each state and time allows vectorization of the nested loop. No "for" or "if" statements are needed within the main loop. (3) n input argument determines the number of initial states to compute (all powers of q). Otherwise plotting multiple states at t=0 would require many calls to the function. (4) The function returns everything that is necessary for plotting, including the grid of states.

Answer the following questions using the function and any supporting code:

1. Using mesh command, present the value function in a 3d graph. Try arguments p = 0.8, q = 1.03, r=1.05, k = 100, 60<x<100, T = 50. Use colors to show the firm decision at each state.

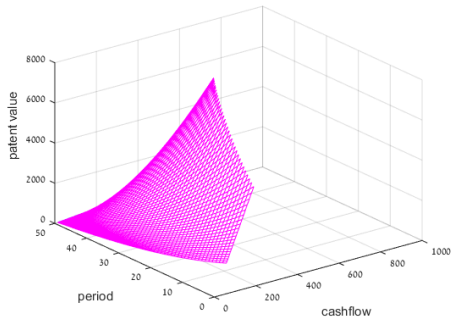
This code does that:

```
[X,T,V,D] = patent_value_Q3(p,q,r,k,K,T,2);

% Now, "triu" basically makes the upper triangular matrix of X, but sets
% the lower triangular part to 0. Thus, we set these values to NaN.
% Finally, we only go down 20 rows for the diagonal because beyond this the
% function explodes and makes the graph look terrible

X = triu(X,-20);
X(X==0) = NaN;

figure;mesh(X,T,V,D)
xlabel('cashflow');
ylabel('period');
zlabel('patent value');
colormap(cool)
```



One important thing to notice is the exclusion of states that are not well computed (below the -20 diagonal of X). The picture shows that the value increases nonlinearly with the state, and decreases with time. Also, using the colors, you can notice that the threshold for renewal is increasing with time, and is always bellow k=100, so that the firm is willing to renew in some cases where the current payoff is negative.

2. How would this analysis change if renewal fees were eliminated?



The firm always renews! So computing the value becomes as easy as computing the expected value of the payoff at each state and discounting.