1 Social planners problem

Imagine an economy consisting of one representative consumer with the reservation price p. Every period the social planner produces a single unit of a good at cost x, selling the unit to the consumer, thereby creating surplus $p - x$. The social planner also decides whether to invest $a = 1$, or not $a = 0$, in cost reducing production technology, at an investment cost $K(c)$. Consequently the social planners per period payoff can be written as:

$$u(s, a) = p - x - K(c)a \tag{1}$$

where a is control and $s = (x, c)$ are state variables. The variable c are *state of the art* cost and should the social planner decide to invest, in the current period, the cost of production next period x', will take the weakly lower value c, according to the law of transition:

$$x' = x(1 - a) + ca \tag{2}$$

The state of the art marginal cost are assumed to decrease, according to an exogenous markov proces of technological development, with four possible values $c_1 > c_2 > c_3 > c_4$. With probability $\pi_h = Pr(c' = c_{h+1}|c_h)$, the state of the art marginal cost, drops one level and with probability $1 - \pi_h$, they stay at last periods levels c_h. When state of the art marginal cost reach their lowest level c_4, the probability of further technological development is zero, $\pi_4 = 0$.

To maximize social surplus over the infinite horizon, the social planner therefore has to solve the problem of optimal timing of investments:

$$\max_{\{a_t\}_{t=0}^{\infty}} \mathbb{E} \sum_{t=0}^{\infty} \beta^t u(s_t, a_t) \tag{3}$$

The social planners optimization problem can be solved using value function iteration. Specifically we are going to use the alternative specific value functions $v_N(s)$ and $v_I(s)$, which we find rewriting the Bellman equation:

$$V(s) := \max_{a \in \{0,1\}} \{u(s, a) + \beta \mathbb{E}[V(s')|s, a]\} \tag{4}$$

defining alternative specific value functions, for investing and not investing respectively:

$$v_I(s) := u(s, 1) + \beta \mathbb{E}[V(s')|s, a = 1] \tag{5}$$

$$v_N(s) := u(s, 0) + \beta \mathbb{E}[V(s')|s, a = 0] \tag{6}$$

such that the value function can be written:

$$V(s) := \max \{v_N(s), v_I(s)\} \tag{7}$$

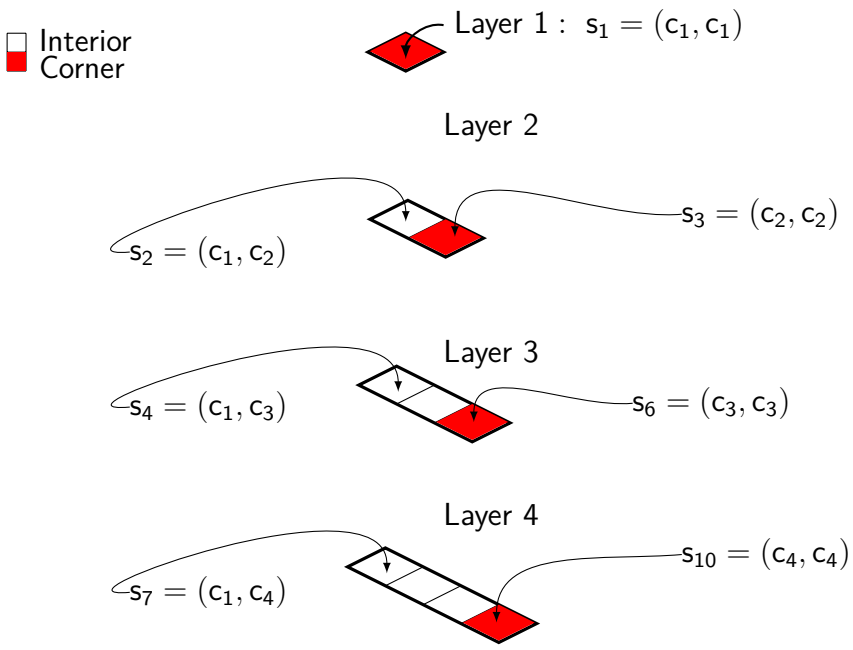

Figure 1: State space for social planners problem

If you want to experiment writing some `Matlab code` there's an algorithm description in appendix A, as well as code, for solving the problem using classic value function iteration. Here we take another approach, solving the problem using *state recursion*, which intuitively is defined as doing backwards induction, on the possible states of the state variable. Before doing this, please solve exercise 1, in order to get a grasp of the *directedness* of the state space, which is a precondition for doing backwards induction.

Exercise 1

Assume that the state of the economy starts out in $s_1 = (c_1, c_1)$, as given in figure 1. Using the law of transition, what state does the economy transition to, if the social planner decides to invest, assuming there is no technological development, from current period to the next period?

Assuming instead that technology does develop, what state will the economy be in next period, irrespective of whether the social planner invest or not?

Assume that the economy is in the state s_2, what state will the economy be in if the social planner invest and technology does not develop?

Assume that the economy is in the state s_2, what state will the economy be in if the social planner invest and technology does develop?

2 State recursion algorithm

The movements of the state variable s in the state space S, as shown in figure 1, is always directed downwards and towards the corner. The directed movements implies, that the state space can be ordered:

$$s_1, s_2, ..., s_{10} \tag{8}$$

To find the value function $V(s)$, using the state recursion algorithm, we first need to solve for $V(s_{10})$, then for $V(s_9)$ etc. all the way to $V(s_1)$. The state recursion algorithm, does not say how to find these values, but it turns out that we need four procedures, according to whether s is a corner or not, and according to whether s is in layer 4 or not. Layer 4, composed of s_7, s_8, s_9, s_{10}, is special because there is no uncertainty about next periods state of the art cost and the corners are special because, even if the social planner decides to invest, production cost will not decrease, because production is already using state of the art production technology.

The derivation of the solution procedures are given in appendix B, except for $V(s)$, where s is in interior state of a layer not equal to layer 4, the solution procedure of which you are asked to derive in exercise 2.

Exercise 2:

Let $s \in \{s_2, s_4, s_5\}$ such that $s = (x, c)$ is an interior point not from layer 2 or 3. Assume that $\pi_h = \pi$ is the probability of technological development. Show that:

$$V(x,c) = \max\{v_I(x,c), v_N(x,c)\} = \max\left\{\frac{p - x + \beta\pi V(x,\tilde{c})}{1 - \beta(1-\pi)}, v_I(x,c)\right\} \tag{9}$$

where $\tilde{c}$ is next level of state of the art production, such that $\tilde{c} = c_2$ if $c = c_1$ and $\tilde{c} = c_3$ if $c = c_2$. Solve the excercise using the following steps:

Step 2.1

Using the order of the state space as indicated in figure 1, convince yourself that $V(x, \tilde{c})$, $V(c, \tilde{c})$ and $V(c, c)$ are all known values, when the state recursion algorithm solves for $V(x, c)$ due to the backwards induction approach.

Step 2.2

Show that the alternative specific value function for investing $v_I(s)$ can be written as:

$$v_I(x,c) = p - x - K(c) + \beta[\pi V(c,\tilde{c}) + (1-\pi)V(c,c)] \tag{10}$$

and conclude that it depend on known values only (The function K(c) is assumed known).

Step 2.3

Show that the alternative specific value function for investing $v_N(s)$ can be written as:

$$v_N(x,c) = p - x + \beta[\pi V(x,\tilde{c}) + (1-\pi)V(x,c)] \tag{11}$$

then assume that $v_N(x,c) \geq v_I(x,c)$ implying that $V(x,c) = \max\{v_I(x,c), v_N(x,c)\} = v_N(x,c)$ and show that under this assumption:

$$v_N(x,c) = \frac{p - x + \beta\pi V(x,\tilde{c})}{1 - \beta(1-\pi)} \tag{12}$$

concluding that if $v_N(x,c) \geq v_I(x,c)$ then:

$$\max\{v_I(x,c), v_N(x,c)\} = \max\left\{\frac{p - x + \beta\pi V(x,\tilde{c})}{1 - \beta(1-\pi)}, v_I(x,c)\right\} \tag{13}$$

Step 2.4

Now assume that $v_I(x,c) > v_N(x,c)$ and show that the following two conditions hold:

$$v_N(x,c) = p - x + \beta[\pi V(x,\tilde{c}) + (1-\pi)v_I(x,c)] \tag{14}$$

$$v_N(x,c) > p - x + \beta[\pi V(x,\tilde{c}) + (1-\pi)v_N(x,c) \tag{15}$$

use this to argue that:

$$v_I(x,c) > v_N(x,c) > \frac{p - x + \beta\pi V(x,\tilde{c})}{1 - \beta(1-\pi)} \tag{16}$$

concluding that if $v_I(x,c) > v_N(x,c)$ then:

$$\max\{v_I(x,c), v_N(x,c)\} = \max\left\{\frac{p - x + \beta\pi V(x,\tilde{c})}{1 - \beta(1-\pi)}, v_I(x,c)\right\} \tag{17}$$

such that the equality holds whether v_I or v_N is largest.

Code 2.5

Go to the folder `SocialPlanner` and open the file `solveScriptSP.m`. Using the following settings:

```
techlevels = 4; % Number of states in technological proces (c1,c2,c3,,c_[techlevels])
pf = 1; % Easy way to alter all probabilities of tech. development
        % pi_h := Pr(c' = c_h+1 | c = c_h)
        % pi_h = pf
mp.beta = exp(-0.05); %discount parameter: beta = exp(-r*dt)
mp.k1 = 8.3; % investment cost parameter ... change function to get non-constant investment cost
mp.c = linspace(5,0,techlevels); % possible values of state of the art cost and social planners cost
mp.p = max(mp.c); % reservation price
mp.prob = [ones(1,techlevels-1),0].*pf; % transition probabilities for technological proces
mp.dt = 1; %  timeincrement
doplot = 0; % 0 for no plotting and 1 for plotting
```

solveScriptSP.m

solve the social planners problem and find the value for $V(s_1) = V(c_1, c_1) = V(5, 5) =$ `V(?,?)`, noting that 5 is a value for marginal cost not the index value. We are going to use the value later on so make sure to write it down or store it as a variable, you do not overwrite [1].

Code 2.6

In order to describe the investment policy of the social planner increase the number of `techlevels` to `10`. Access the code for the function $K(c)$ and change the cost function to be constant:

```
function [out] = K(c,mp)
    % Calculating investment cost as function of
    % state of the art production cost c
     out = mp.k1;
    % out = mp.k1/(1+c);
end
```

Constant investment cost K(c) = k1

Consider the function:

$$f(x, c) = \frac{\beta}{1 - \beta}(x - c) \tag{18}$$

try to maximize $f(.,.)$ with respect to (x, c) over the set $C \times C$ with

$$C = \texttt{linspace(5,0,techlevels) =(5,...,0)} \tag{19}$$

as set in the modelspecifications. Having found the maximum value $y^* = f(x^*, c^*)$, set the constant investment cost `mp.k1`, equal to this value, y^*, and solve the model.

- Does the social planner invest in any period? Inspect the `policy` object giving the solution for the social planners policy rule. You can inspect the solvers to figure out what is stored in `policy` or use the following identities:

$$\texttt{policy} = \begin{bmatrix} \delta(c_1, c_1) & \delta(c_1, c_2) & \delta(c_1, c_3) & .... & \delta(c_1, c_{10}) \\ \texttt{NaN} & \delta(c_2, c_2) & \delta(c_2, c_3) & .... & \delta(c_2, c_{10}) \\ \texttt{NaN} & \texttt{NaN} & \delta(c_3, c_3) & .... & \delta(c_3, c_{10}) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \texttt{NaN} & \texttt{NaN} & \texttt{NaN} & ... & \delta(c_{10}, c_{10}) \end{bmatrix} \tag{20}$$

$$\delta(x, c) = \begin{cases} 1 & \text{if invest} \\ 0 & \text{if non-invest} \end{cases} \tag{21}$$

Is this investment pattern found using the code as you would expect? (you are welcome to consult **Theorem 1** of the *The Dynamics of Bertrand Price Competition with Cost-Reducing Investments* paper by Fedor Iskhakov, John Rust and Bertel Schjerning (2015)).

Change the costfunction to

[1] The object `V` is going to be overwritten.

```
function [out] = K(c,mp)
    % Calculating investment cost as function of
    % state of the art production cost c
    %  out = mp.k1;
     out = mp.k1/(1+c);
end
```

Constant investment cost K(c) = mp.k1/(1+c)

without changing the value for `mp.k1`$= y^*$, solve the model and consider the policy function of the social planner once more. Does the social planner invest in any states $s = (x, c)$ and would you expect this? You may find it helpful to consider how the investment cost change, when the state of the art marginal cost c change.

3 The leapfrogging game

Consider a discrete-time, infinite horizon stochastic game, where two firms $j \in \{1, 2\}$, are producing an identical good, at a constant marginal cost of x_1 and x_2, respectively. The two firms are assumed price setters using a Bertrand equilibrium pricing strategy $p(c_1, c_2) = \max\{c_1, c_2\}$. Each period the firms get the per period profit:

$$r_j(x_1, x_2) = \max\{c_1, c_2\} - x_j \tag{22}$$

The two firms have the ability to make an investment to replace their existing plant with a new state of the art production facility. An investing firm pays the investment cost $K(c)$ and the firms marginal cost x_j transition to the state of the art marginal cost c the next period, according to the law of transition:

$$x'_j = x_j(1 - a_j) + ca_j \tag{23}$$

where $a_j = 1$ if firm j decides to invest and $a_j = 0$ otherwise.

Each period, both firms observe the state of the industry $s = (x_1, x_2, c)$, set their prices and simultaneously decides whether or not to invest in the state of the art production technology. The firms are assumed to start in the state $s = (c_1, c_1, c_1)$, as illustrated in figure 2. The state of the art marginal cost c develop according to the exogenous markov proces, decreasing from c_{h-1} to c_h with probability $\pi(c' = c_h|c_{h-1}) = \pi_h$, until eventually reaching c_4, where further decrease is impossible $\pi(c' = c_4|c_4) = 1$.

Assuming that the two firms are expected discounted profit maximizers and have a common discount factor β, the alternative specific value functions:

$$v_{N,j}(x_1, x_2, c) = r_j(x_1, x_2) + \beta\mathbb{E}[V(x'_1, x'_2, c')|a_j = 0, \sigma_{is}, c] \tag{24}$$

$$v_{I,j}(x_1, x_2, c) = r_j(x_1, x_2) + -K(c) + \beta\mathbb{E}[V(x'_1, x'_2, c')|a_j = 0, \sigma_{is}, c] \tag{25}$$

must hold for each player j, in each state $s = (x_1, x_2, c)$ with σ_{is} being the strategy played by firm $i \neq j$ in the state s. The strategy σ_{is} is a probability distribution determining the

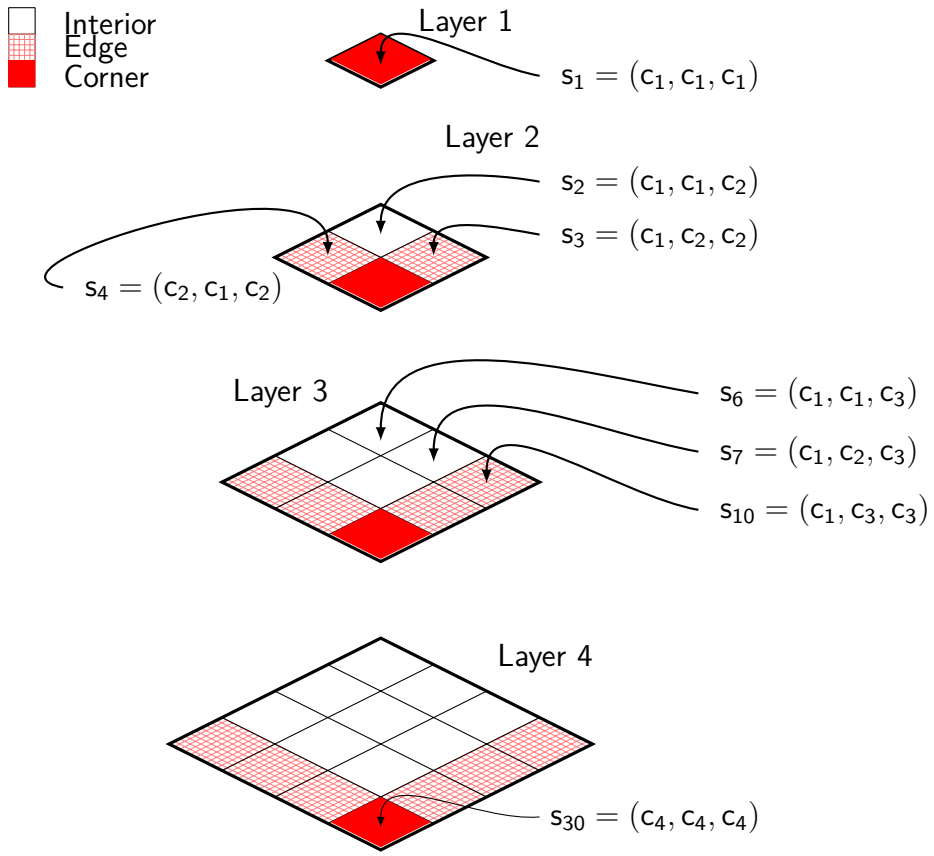

Figure 2: State space of leapfrogging game

probability that firm i invest and $x_i' = c$ or that firm i does not invest, such that $x_i' = x_i$. A complete markov strategy for firm j includes a strategy σ_{js} for each state of the state space in figure 2.

Exercise 4

First make sure you have the `Leapfrog` library on your `Mathlab path`. To get acquainted with the state space of the model execute the following codeblock, to set up the parameters of the leapfrog model under consideration:

```
probfactor = 1;
markovProbs = [1,1,1,0].*probfactor;
marginalCost = linspace(5,0,length(markovProbs));
investmentCost = 8.3;

cost = init_cost(markovProbs,marginalCost,investmentCost);
mp = init_mp(cost);
```

(1) solveLeapfrog

where $\pi_h =$ `markovProbs(h)`, $c_h =$ `marginalCost(h)` and `investmentCost = cost.k1` in the function $K(c)$ as defined in `K.m`.

Inspect the vector (c_1, c_2, c_3, c_4) given by `markovProbs` and use the following commands to print out marginal cost matrices of the state space layers:

```
cost.firm1
cost.firm2
```

solveLeapfrog

Notice that `cost.firm1(i,j,h)` are the marginal cost of firm 1 in the state $s = (c_i, c_j, c_h)$, hence equal to c_i.

Try also to excute the call:

```
showss(cost)
```

solveLeapfrog

What are the marginal cost of firm 1 and two respectively in $s = (c_1, c_2, c_3)$? Is this point an interior, edge or corner point in figure 2?

Execute the line:

```
ss = cSS(cost.nC);
```

(2) solveLeapfrog

in order to initialize an empty state space structure. Try to access the slot reserved for $s = (c_i, c_j, c_h)$ where $(i, j, h) = (1, 1, 4)$. Because each state space point can hold several equilibria, you also need to choose an index value for the equilibrium you are interested in. Set the equilibrium index $k = 5$ and then execute:

```
ss(h).EQs(i,j,k).eq
ans =

    P1: []
   vN1: []
   vI1: []
    P2: []
   vN2: []
   vI2: []
```

solveLeapfrog

the output shows, that an equilibrium holds information on probability of investment for firm 1, `P1`, as well as the values of not investing `vN1` and investing `vI1`. Similarly for firm 2.

Exercise 5

When the firms invest, the state changes in the direction from interior to the edge or corner, or they move from the edge towards the corner in figure 2. If technology develops, which it eventually will, the movement are directed downwards, to the next layer. When reaching s_{30}, neither investments of firms, nor technological development changes the state of the economic system.

Due to this directionality of the state space we can order the state space points:

$$s_1, s_2, ..., s_{30} \tag{26}$$

and apply the state recursion algorithm by solving first the stage game associated with s_{30}, then the stage game associated with s_{29} etc.

Focusing on layer 4, the last layer, where technology is fully developed, you should first solve the corner, then the edges and finally the interior.

$$\underbrace{s_{14}, ..., s_{22}}_{interior} \quad \underbrace{s_{23}, ..., s_{29}}_{edges} \quad \underbrace{s_{30}}_{corner} \tag{27}$$

the method used to identify stage game equillibria pertains to each of these groups, hence you need to use three different solvers. If you have cleared the workspace rerun codeblocks *(1) solveLeapfrog* and *(2) solveLeapfrog*, then execute the following lines:

```
ESS = cESS(cost.nC); %Equilibrium selection rule ... more on this later
[ss,ESS] = solve_last_corner(ss,ESS,cost,mp)
[ss,ESS] = solve_last_edge(ss,ESS,cost,mp)
[ss,ESS] = solve_last_interior(ss,ESS,cost,mp)
```

(3) solveLeapfrog

You have now solved the final layer and should answer the following questions, by accessing the relevant equilibria information, using `ss(h).EQs(i,j,k).eq`. However, you may want to call `ss(4).nEQ`, to see how many equilibria have been found in each state space point of the final layer, in order to avoid inserting values of `k`, where nothing has been stored.

1. Does any of the firms invest in the final stage game $G(s_{30})$ and what is the value the firms receive?

2. Does the cost follower ever invest in the edge game $G(s_{23}), ..., G(s_{29})$ and what value does the cost follower have?
   $(i, j, h) \in \{(1, 4, 4), (2, 4, 4), (3, 4, 4), (4, 1, 4), (4, 2, 4), (4, 3, 4)\}$

3. Find equilibrium `k=2` for the point $s = (c_1, c_2, c_4)$ and control the values manually. Because this is an equilibrium where firm 1 invest, `P1=1`, we have the following formulas:

$$v_{N2}(c_1, c_2, c_4) = r_2(c_1, c_2) + \beta p_1 V_2(c_4, c_2, c_4) + \beta \underbrace{(1 - p_1)}_{=0} V_2(c_1, c_2, c_4) \tag{28}$$

   there is no uncertainty of technological development, hence no expectation over state of the art marginal cost. Only the marginal cost of firm 1 is uncertain to firm 2, but given that firm 1 invests only $V_2(c_4, c_2, c_4)$ remains relevant. However $V_2(c_4, c_2, c_4) = 0$, because firm 2 is cost follower and the state has moved to the edge. Therefore the value of not investing for firm 2 is a single period of the relevant flow profit:

$$v_{N2}(c_1, c_2, c_4) = r_2(c_1, c_2) = c_1 - c_2 \tag{29}$$

   If firm 2 invests when firm 1 invests for certain:

$$v_{I2}(c_1, c_2, c_4) = r_2(c_1, c_2) - K(c_4) + \beta V_2(c_4, c_4, c_4) \tag{30}$$

use answer to (1) to argue that $V_2(c_4, c_4, c_4) = 0$ and conclude that:

$$v_{I2}(c_1, c_2, c_4) = r_2(c_1, c_2) - K(c_4) = c_1 - c_2 - K(c_4) \tag{31}$$

check if manual evaluation of these formulas results in the correct values for firm 2's payoff using `ss(4).EQs(1,2,2).eq`.

4. Why can't firm 2 uphold its cost leadership in $s = (c_1, c_2, c_4)$, by using the threat, of investing if firm 1 invests, and thereby succesfully discourage firm 1 from investing in the present state space point $s = (c_1, c_2, c_4)$?

5. Finally take a look at the objects `ESS.esr` and `ESS.bases`. These vectors will change, when the model is solved for stage equilibria in rest of the state space, in the next exercise.

Exercise 6

Considering the non-final layers of the state space, each have interior, edges and corner except for the first layer, which is only a corner:

$$\text{Layer 1:} \quad \underbrace{s_1}_{corner} \tag{32}$$

$$\text{Layer 2:} \quad \underbrace{s_2}_{interior} \quad \underbrace{s_3, s_4}_{edges} \quad \underbrace{s_5}_{corner} \tag{33}$$

$$\text{Layer 3:} \quad \underbrace{s_6, s_7, s_8, s_9}_{interior} \quad \underbrace{s_{10}, s_{11}, s_{12}, s_{13}}_{edges} \quad \underbrace{s_{14}}_{corner} \tag{34}$$

because the technology can develop, in any of these states, the solution procedure is different from the one used for the final layer. Similar to the case of the final layer, there are three different solvers, applied according to whether the state is interior, edge or corner. According to the backwards induction principle of the state recursion algorithm, the solvers are applied first to s_{14}, then s_{13} etc. taking care to use the right solver, which can be achieved using the following loop structure:

```
for ic = fliplr(1:cost.nC-1)   % [3,2,1]
      % Insert your code here ;
       if ic ~= 1
           % Insert your code here
       end
 end
```

(4) solveLeapfrog

- With *solveLeapfrog (1-3)* executed you now have to insert calls to the correct solver and capture the output from them in the loop structure in *solveLeapfrog (4)*. The solvers to call are named `solve_corner`, `solve_edge` and `solve_interior`.

When this is done you have succesfully found *one* Markov Perfect Equilibrium of the Leapfrog model.

- Inspect the object `ESS.bases`

You should have a vector like the following:

$$ne(\boldsymbol{\gamma}) = \Big(1,1,1,1,1,1,1,3,3,3,3,3,3,3,3,3,1,1,1,1,1,3,1,1,1,1,1,1,3,1\Big) \tag{35}$$

informing, you of the number of stage equilibria associated with a certain state space point. In particular coordinate `ESS.index(i,j,h)` of `ESS.bases` is the number of equilibria associated with the state space point (h,i,j).

- Inspect the object `ESS.esr`

You should have a vector like the following:

$$\boldsymbol{\gamma} = \Big(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0\Big) \tag{36}$$

consisting of 30 zeros, which is the *equilibrium selection rule*. Coordinate `ESS.index(i,j,h)` of the equilibrium selection rule informs you, which equilibrium is used in the s-stage game associated with the state space point (h,i,j). Each $'0'$ indicates that the first equilibrium is used and in general a number z indicates, that the $z+1$'th equilibrium is used.

Exercise 7

In the following exercise we focus on some economic aspects of the different equilibria.

- Using the function `simulatepath()` make a priceplot for the equilibrium found in exercise 6. Are there any change in the price? Why/Why not?

```
13 [tech,mcCost,prices,P] = simulatepath ( ss , ESS , cost , 3 );
```

solveLeapfrog

- Compare the value $V_j(s_1)$ in `ss(1).EQs(1,1,1).eq`, where j is the firm index of the firm making all the investments, to the value $V(s_1)$ for the social planner. What can you conclude about the social efficiency of the equilibrium found in exercise 6?

  You are welcome to relate your answer to **Theorem 2** in *The Dynamics of Bertrand Price Competition with Cost-Reducing Investments* by Iskhakov, Rust and Schjerning.

To get a more interesting equilibrium to simulate increase the number of technological levels to 11 by setting `markovProbs = [1,1,1,1,1,1,1,1,1,1,0]`. Load the matrix containing equilibrium strings using `csvread('leapfrog_esr_1.csv')`. Each column of the matrix is an equilibrium selection string, with unequal columns being for deterministic technological development, `propfactor=1`, and equal columns for stochastic, specifically `propfactor=0.5`.

- Make plots of the price curves for the equilibria by first solving for the equilibrium and then using `simulatepath()`.

```
probfactor = %set the probfactor to 1 or 0.5;
markovProbs = [1,1,1,1,1,1,1,1,1,1,0].*probfactor;
marginalCost = linspace(5,0,length(markovProbs));
investmentCost = 8.3;

cost = init_cost(markovProbs,marginalCost,investmentCost);
mp = init_mp(cost);

ss = cSS(cost.nC);
ESS = cESS(cost.nC);
temp = csvread('leapfrog_esr_1.csv');

ESS.esr = temp(:,% column index must be inserted % );


    [ss,ESS] = solve_last_corner(ss,ESS,cost,mp);
    [ss,ESS] = solve_last_edge(ss,ESS,cost,mp);
    [ss,ESS] = solve_last_interior(ss,ESS,cost,mp);


    for ic = fliplr(1:cost.nC-1)
        [ss,ESS] = solve_corner(ss,ic,ESS,cost,mp);
        if ic ~= 1
            [ss, ESS] = solve_edge(ss,ic,ESS,cost,mp);
            [ss, ESS] = solve_interior(ss,ic,ESS,cost,mp);
        end
    end

 % call to function simulating the MPE in ss of the model
[tech, mcCost, prices , P ] = simulatepath(ss,ESS,cost,[1,2,3])

% tech return matrix of technological level of firms and state of the art technological level column
    1,2,3
% mcCost return matrix of marginal cost of firms and state of the art marginal cost column 1,2,3
% prices: the price is the marginal cost of the high cost follower
% P is the firm strategies as probability of investment and the probability of technological
    development
% argument : [1,2,3] controls what plots are made ... here all possible are selected.
```

solveLeapfrog

Solving for the equilibria in `csvread('leapfrog_esr_1.csv')` should result in two monopoly equilibria, where one of the firms makes all the investments. If the timing of these investments coincide with the investments of the social planner they should be socially optimal and the value function of the social planner in the initial state `V(1,1)` should equal, the value of the firm making all the investments, which can be found in `ss(1).EQs(1,1,1).eq`.

- If time permits it you can solve the social planners problem with 11 technological levels and `probfactor` equal to `1` or `0.5` and compare `V(1,1)` to the monopoly expected payoff given in `ss(1).EQs(1,1,1).eq`. Are the monopoly equilibria socially optimal? If not then:

  `V(1,1) > max{ss(1).EQs(1,1,1).eq.vN1,ss(1).EQs(1,1,1).eq.vN2}`.

- If they are not socially optimal why do you think this is? Note you can see the frequency of investments of the social planner by making a plot `plotpath(policy,mp)` after solving the social planners problem [2].

[2]Note that both the social planner code and the leapfrog code generates an object `mp` and they are not identical, so if errors occur it could be due to not running `plotpath(policy,mp)` directly after solving the social planners problem, such that the object `mp` is the object generated by the leapfrog code.

Exercise 8

In this exercise you are asked to reproduce the plot shown in figure 3, where each point plotted is $(V_1(s_1), V_2(s_1)) = \left(\texttt{ss(1).EQs(1,1,1).eq.vN1,ss(1).EQs(1,1,1).eq.vN2}\right)$ for a given MPE. To plot the values for all MPE you first have to solve the model for all MPE using the Recursive Lexicographical Search algorithm.

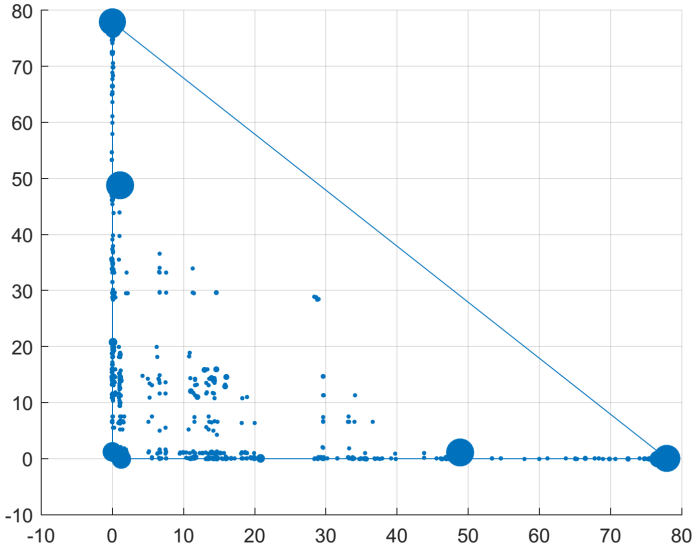

Figure 3: (V_1, V_2)-plot

Considering the leapfrog model with only 3 technological levels there is a total of 7 stages, now referring to interior, edges and corner of each layer as stages:

$$\text{Layer 1:} \quad \underbrace{s_1}_{stage\ 1} \tag{37}$$

$$\text{Layer 2:} \quad \underbrace{s_2}_{stage\ 2} \quad \underbrace{s_3, s_4}_{stage\ 3} \quad \underbrace{s_5}_{stage\ 4} \tag{38}$$

$$\text{Layer 3:} \quad \underbrace{s_6, s_7, s_8, s_9}_{stage\ 5} \quad \underbrace{s_{10}, s_{11}, s_{12}, s_{13}}_{stage\ 6} \quad \underbrace{s_{14}}_{stage\ 7} \tag{39}$$

Using the default equilibrium selection string consisting of zeros only:

$$\boldsymbol{\gamma}^1 = \left(0_{14}, 0_{13}, 0_{12}, 0_{11}, 0_{10}, 0_9, 0_8, 0_7, 0_6, 0_5, 0_4, 0_3, 0_2, 0_1\right) = \texttt{ESS.esr}$$

the model is solved using state recursion. The indexation on the zeros, indicate which state space point, the digit of the equilibrium selection string is associated with.

Having solved the model the bases are updated:

$$ne(\boldsymbol{\gamma}^1) = \left(1_{14}, 1_{13}, 1_{12}, 1_{11}, 1_{10}, 3_9, 3_8, 3_7, 3_6, 1_5, 1_4, 1_3, 3_2, 1_1\right) = \texttt{ESS.bases}$$

the model is solved using backwards induction, because the payoffs and equilibria associated with state space points in a lower indexed stage, for example s_2, depend on the equilibria played in state space points of the higher indexed stages, for example s_6. Since the bases are the number of equilibria associated with a given state space point, the base of the lower indexed state space points, 3_2 in $ne(\boldsymbol{\gamma}_1)$ associated with s_2, depend on the equilibria played in state space points, belonging to higher indexed stages, such as 0_6 choosing the first equilibrium for s_6 according to $\boldsymbol{\gamma}_1$. This means that changing the equilibrium played in s_6 could change the number of available equilibria in s_2. However, it could never be the case, that changing the equilibrium played in s_2, changes the number of available equilibria in s_6. Because of this logic, we change the equilibrium selection rules from the right to the left.

This is achieved by concieving the equilibrium selection string as a number and realizing, that when we add one to a number, for example $1000 + 1 = 1001$, we change the number from the right to the left. Working in base 10, each digit is allowed to reach 9 before it overflows, returns to 0 and one is added to the next digit, as in $1009 + 1 = 1010$. When we add one to an equilibrium selection string, we do not allow the digits to reach 9, instead we let the bases `ESS.bases`, determine the overflow. To update the equilibrium selection rule `ESS` the code uses the function `addOne()` such that:

$$\boldsymbol{\gamma}^2 = \texttt{addOne}(\boldsymbol{\gamma}^1) = \Big(0,0,0,0,0,0,0,0,0,0,0,0,0,1,0\Big)$$

choosing the second equilibrium in s_2 since $\boldsymbol{\gamma}^2(s_2) = 1$. And because any digit to the left of 1 in $\boldsymbol{\gamma}^2$ is the same as in $\boldsymbol{\gamma}^1$, we must have that the relevant base of $\boldsymbol{\gamma}^2$ is still 3 because $ne[\boldsymbol{\gamma}^2](s_2) = ne[\boldsymbol{\gamma}^1](s_2) = 3$, the base was 3 for $\boldsymbol{\gamma}^1$.

To apply the algorithm the state recursion can be inserted into the following while-loop structure:

```
% Define model probfactor, techlevels etc.
% Initialize ss and ESS

neq = 0; %Start with 0 equilibria
maxEQ = 10;
out = NaN(2*maxEQ,length(ESS.bases));
while neq < maxEQ;
     % Begin state recursion

     % End state recursion
    neq = neq + 1; %We have one more equilibria now
    out(2*neq-1,:) = ESS.esr; %Store the equilibrium selection rule
    out(2*neq,:) = ESS.bases; %Store the associated bases
    ESS = addOne(ESS) %Given the new bases addOne
    if all(ESS.esr==-1) % Stop while loop when ESS overflows
       maxEQ=neq;
       out = out(1:neq,:);
    end
end
```

solveLeapfrog

- Use the while loop structure along with the previous code for the state recursion, to solve for the first 10 equilibria, `maxEQ=10`, in the leapfrog model with deterministic technological development, `probfactor =1`, and with 3 technological levels. Look

at the matrix `out` to see how the equilibrium selection rules change compared to the bases (bases are equal row numbers)[3].

- Consider whether $\boldsymbol{\gamma}^k = (0,0,0,0,0,0,0,0,1,0,0,0,1,0)$ is feasible?
- Solve for all equilibria for the leapfrog model with deterministic technological development and with respectively 2,3 and 4 technological levels. Use `tic` and `toc` codes to time the procedure. How many equilibria are there for each of these models?
- For the model with 4 technological levels make a plot as shown in figure 3, using the function `vscatter(.)`. The last argument of `vscatter(.)` called `VM` should be the value found for the social planners value function in exercise 2.2.5.

Exercise 9

Consider the leapfrog model with 4 technological levels and assume that the equilibrium selection rule is updated from $\boldsymbol{\gamma}^k$ to $\boldsymbol{\gamma}^{k+1}$, in such a manner that another equilibrium is played in the stage game $G(s)$ where $s \in$ *stage* 5. Assume further that no changes are made in the equilibria played in any stage game $G(s)$ with $s \in$ *stage* $\tau > 5$:

$$\text{Layer 1: } \underbrace{recalculate}_{stage\ 1}$$

$$\text{Layer 2: } \underbrace{recalculate}_{stage\ 2} \quad \underbrace{recalculate}_{stage\ 3} \quad \underbrace{recalculate}_{stage\ 4}$$

$$\text{Layer 3: } \underbrace{change}_{stage\ 5} \quad \underbrace{s_{10}, s_{11}, s_{12}, s_{13}}_{stage\ 6} \quad \underbrace{s_{14}}_{stage\ 7}$$

$$\text{Layer 4: } \underbrace{s_{15}, ..., s_{23}}_{stage\ 8} \quad \underbrace{s_{24}, ..., s_{29}}_{stage\ 9} \quad \underbrace{s_{30}}_{stage\ 10}$$

this implies that only the payoffs associated with state space points, in the stages of a lower index than 5 has to be updated. A full implementation of the recursive lexicographical search algorithm takes this into account.

When the equilibrium selection rule is updated $\boldsymbol{\gamma}^{k+1} =$ `addOne(`$\boldsymbol{\gamma}^k$`)`, the highest stage index, τ_0, with a change in played equilibria of a stage game is found and the state recursion algorithm is only applied to the stages $\tau < \tau_0$. This is implemented in the script `solveRLS.m`.

- Solve the leapfrog model with deterministic technological development and 4 technological layers using `solveRLS.m`.
- Look at the generated table, stating how many times the algorithm starts the state recursion algorithm in a certain stage $\tau = \tau_0 - 1$.

[3] If you give up on writing the code there is a working code in `exercise8.m`.

- Compare the computation time with the time you found in exercise 8 for the same model.