Problem VI: Getting confidence in market equilibriums – an agent-based perspective

The aim of this work is to compare the classic Cournot oligopoly equilibrium (computed with optimality first order conditions) with results obtained with a simple iterative process during which market players are progressively discovering their optimal strategies.

We are studying a stylized industry where marginal cost of production is supposed to be constant and equal to zero. The market demand is well described by the inverse demand function: P(Q)=600 – Q.

Case 1 : traditional Cournot equilibriums

1. Compute the market equilibrium (price and quantity sold by each firm) when the market is served by: a) a monopoly; b) a symmetric duopoly and c) a symmetric triopoly.

Case 2 : Roth & Erev, a Reinforcement learning process

The goal of this problem is to build a Cournot one-sided auction. It is based on a set of stylised agent-based simulations with computational learning based on the Roth and Erev (1995) algorithm. This market will be simulated for several hundred periods until it reaches a stationary state. This exercise will be repeated many times for different market concentration (monopoly, duopoly and triopoly) before results will be compared to the traditional Cournot equilibriums.

We are assuming that producers are involved in a Cournot one-sided auction. In this kind of auction, each individual firm freely select its output (quantity). All the firms are then offering (simultaneously) their quantities to the market. A clearing process is then applied to compute a uniform price (price is the same for all the market participants) given all those submitted quantities. Taking this price information, market players can compute their profits and are allowed to adjust their output decision. Firms can then submit a new quantity to the market (simultaneously). A market clearing is then organized and so on (this iterative process is repeated)...

Market rules

We are studying an oligopoly wholesale commodity market linking a total of *n* "producers"[8] (i.e. wholesaler sellers), and an inverse demand function: P(Q)=600 – Q where $Q = q_1+...+q_i+...+q_n$ where *i* name player *i*. All players are homogeneous (cost = 0).

The simulations consist of supply bidding made by each individual producer. Each bid represents the quantity that will be produced by the firm who selected it. At each iteration, firms can freely selects a bid (also named a supply strategy) that is bounded between zero excluded, and a maximum capacity: K = 5×integer_part(120/*n*). In this framework, all the firms have identical capacities so that K will be the same for all the market players.

To simplify, let us suppose that for each individual firm, the set of allowable bids is discrete and can easily be indexed by an integer number *s=1, . . ., S* bounded between 1 and *S*=integer_part(120/n) so that bids can only take discrete values such as 5-units increments: 5, 10, ..., K-5, K with K= 5×*S*. Thus, players choose their supply strategies (supply quantity) from the set of *S* values described in Table 2.

Table 2: correspondence between index value and the supplied quantity

Bid index *s*	Quantity produced q_i
1	5
...	...
s	5.*s*
...	...
S = integer_part(120/n)	K =5×integer_part(120/n)

[8] In case of a Monopoly, n=1; in case of a duopoly, n=2 and n=3 in case of a triopoly...

Behavioural rules

The way market participant are updating their previous bid is called a behavioural assumption. In this case, we are focussing on a simple method called "**reinforcement learning**". At the beginning of every period the agents choose quantity strategies, and at the end of it they try to learn how to improve their bidding behaviour.

We are building a framework where each firm selects randomly a supply strategy in the discrete set, indexed by *s*. At the beginning of the simulation process (time t=1), all the strategies have a uniform initial probability to be played. In every period (time indexed as t), the interactions follows this pseudo-code:

> (1) Each Producer *i* selects **randomly** and independently a bid index. This corresponds to a bid: a quantity $q_i(t)$ proposed to the market in the period *t*.
>
> (2) All the bids are submitted to the market. The market clears, determining the uniform market price p(t) ;
>
> (3) Producers calculate their profits (for producer *i* : $\Pi_i(t) = p(t).q_i(t)$) ;
>
> (4) Producers reinforce strategies (which means: re-calculate the probabilities of playing each of them) in the next period;
>
> (5) Back to (1) and time t = t + 1.

We now describe in detail how a player can "reinforce" its strategies. To do so, we rely on the concept of propensity which is related to that of probability. Each producer *i* plays each possible action *s*=1, ..., *S* with a given "propensity", $r_s^i(t)$. The probability $p_s^i(t)$ that *i* plays *s* is given by its propensity divided by the sum of the propensities of all possible actions:

$$p_s^i(t) = \frac{r_s^i(t)}{\sum_{s=1}^{S} r_s^i(t)}.$$

The algorithm actualises the propensities assigned to each quantity strategy $r_s^i(t)$ and increases the probability of those that yield better results. We now describe how the propensity of each strategy at *t* + 1 is established as a function of $\Pi_i(t)$. Propensities are initialised to a given value (i.e. $r_s^i(1) = constant$ for all *s*), so that all actions have the same initial probability. At the end of each round, producers update their propensities according to the results obtained by the actions played using a version of the R&E reinforcement rule. This rule follows these steps:

i. Namely, traders reinforce the selected action, λ, through an increase in its propensity by $\Pi_i(t)$. Therefore, chosen strategies resulting in a positive payoff are reinforced to some extent, and there is a strong incentive for firms to trade. Those that do not place firms "in the money" will not be reinforced.

ii. Moreover, actions that are similar (i.e. λ−1 and λ + 1), are also reinforced, but to a lesser extent, through an increase in their propensity by $(1-\delta).\Pi_i(t)$ where $0 < \delta < 1$ ("persistent local experimentation").

iii. Independent of trading performance, the importance of past experience is reduced by discounting all propensities by γ ("gradual forgetting").

iv. Finally, actions whose probability falls below a certain threshold are removed from the choice space ("μ, extinction in finite time").

The propensities are computed in two steps. First, the pre-extinction propensities $r_s^i(t)'$ are:

$$r_s^i(t)' = \begin{cases} (1-\gamma)r_s^i(t)(t-1) + \Pi_i(t) & \text{if } s = \lambda, \\ (1-\gamma)r_s^i(t-1) + (1-\delta)\Pi_i(t) & \text{if } s = \lambda - 1 \text{ or } s = \lambda + 1, \\ (1-\gamma)r_s^i(t-1) & \text{if } s \neq \lambda - 1,\ s \neq \lambda \text{ and } s \neq \lambda + 1. \end{cases}$$

Second[9], the propensities are corrected when the "extinction in finite time" feature is binding:

$$r_s^i(t) = r_s^i(t)' I_{\left\{(r_s^i(t)')/\left(\sum_{s=1}^{Si} r_s^i(t)'\right) > \mu\right\}}$$

where *I* is an indicator function that takes value 1 if the condition is satisfied and 0 otherwise. The algorithm is used analogously on the buyers' side with the corresponding indices.

The probabilities are then determined for each agent with: $p_s^i(t) = \dfrac{r_s^i(t)}{\sum_{s=1}^{S} r_s^i(t)}$

<u>*Building a Matlab framework and running simulations*</u>

The aim of this part is to build a useful toolbox that will be intensively used to run several simulations. This will be realised on the Matlab platform (available on the IFP School network. To access Matlab on the Intranet, use login **Ext14** and password **Ext114** and then with the explorer open P:\Logiciels_IS\Matlab).

<u>Note : MATLAB</u>
MATLAB is a language for technical computing. Information about MATLAB can be obtained at the MathWorks Website. An extensive list of Online MATLAB Resources has also been compiled by Jerod Parker of George Mason University (http://bass.gmu.edu/matlab/matlab.html). A series of MATLAB Tutorials prepared by members of the Department of Mathematics at Southern Illinois University at Carbondale are also available online (http://www.math.siu.edu/matlab/tutorials.html). You can obtain a license by contacting the SMILE team at IFP School.

VERY IMPORTANT: This is just a suggestion. Feel free to use Python or alternative environment if you feel ill at ease with Matlab.

All of the following questions imply the development of a clear and well commented procedure. Of course, students are also invited to furnish a separate answer sheet where the main part of the algorithm is written in pseudo code.

2. Write your own code to simulate an iterative Cournot one-sided auction with R&E reinforcement learning process with parameters : γ, δ, μ. Does it converge to a stationary outcome in a reasonable computational time?

Cournot equilibrium vs simulation outcomes: a comparison

Three different market structures will be scrutinized (a monopoly, a duopoly and an oligopoly case). Each of the three cases will be simulated. All the simulations will be done with the following set of parameters: initial propensities set equal to 90000, $\delta = 0.3$, $\mu = 0.002$ and $\gamma = 0.01$).

3. For each of the three market structures, plot the market price p(t) as a function of time for t=1 to t= 2000. Please give a short comment regarding a possible convergence to stationary outcomes?

[9] When $\lambda=1$ or $\lambda=S$, there is only one neighbouring strategy. If that is the case, the experimentation parameter is applied only to $s = \lambda+1=1$ or $s = \lambda-1 = S-1$, respectively.

To get confidence in the results of this random mechanism, each of the three market structure case will be simulated 50 times (all those 50 simulations will be realised with the following set of parameters[10]: initial propensities set equal to 90000, $\delta = 0.3$, $\mu = 0.002$ and $\gamma = 0.01$).

The number of observations (simulation runs) for each market structure assumption will be 50, with 2000 trading periods in each run. As a result, a total of $3 \times 20 \times 2000 = 120\ 000$ markets clearings will be computed.

The simulations should normally converge to stationary outcomes in approximately 1000 periods. Our attention will focus on the price average for periods [1001, 2000] in each run.

As a result, those 20 experiments per crossholding will give us a sample of 20 equilibrium prices. This sample will be analysed using basic statistical techniques (both the mean and the unbiased estimator of standard deviation will be computed).

4. Display your own simulation results as in Table 2.

Table 2: Simulation results

Market structure	Results Price average for periods [1000, 2000] in run number *r* $\overline{P_r} = \frac{1}{200}\sum_{t=301}^{500} P_r(t)$			Interpretation	
	Run #1 $\overline{P_1}$	...	Run #20 $\overline{P_{20}}$	Mean estimation for a given market structure $Price = \sum_{r=1}^{20} \overline{P_r}$	Unbiased estimation of standard deviation $\sigma = \sqrt{\frac{1}{19}\sum_{r=1}^{20}\left(\overline{P_r} - Price\right)^2}$
Monopoly n=1					
Duopoly n=2					
Triopoly n=3					

Note: Special Matlab functions such as xlswrite and xlsread could be useful to save Matlab results in a .dat format (those kind of format can be used with MSExcel).

5. Compare those results with those computed in Part 1. What can you conclude?
6. You have just built an "agent-based simulation" experiment. What do you think about the "cleverness" of those agents? How do you qualify their learning mechanism (compared to the one classically exhibited by real human confronted to the same problem)? Can this kind of experiment bring confidence in the use of classic equilibriums concepts based on optimization conditions (in this case: a Monopoly and a Cournot oligopoly)?

[10] As quantities are randomly selected, note that these 20 runs can possibly converge to different stationary outcomes.