Solving Dynamic Oligopoly Games

Chuqing Jin

Due date: Mar. 14th 2025.

Please submit both your answer to the questions, and the codes that produces your answers.

In this problem set, you will be asked to compute the equilibrium of a dynamic oligopoly game (Ericson and Pakes 1995) using the algorithm in Pakes and McGuire (1994). You will be provided **a framework of the code** in Matlab and asked to **complete specific functions**. For those of you who use other programming languages, please feel free to use LLMs to translate the codes. You are encouraged to work together on this problem set, but each of you need to submit your own code and answers.

This problem set will be organized as follows. I will first state the dynamic oligopoly game. Then, I will overview the framework of the code. After that, I will specify your tasks for this problem set. Finally, in the appendix, I will provide a dictionary of the variables I use in the code.

1 Dynamic Oligopoly Game

Consider a market with maximum 3 active firms, i.e., $N = 3$. The state variable for firm i, w_i, is any integer between 0 and 19, i.e., $w_i \in \{0, 1, 2, ..., \bar{w}\} \equiv \mathcal{W}$ and $\bar{w} = 19$. An active firm has $w_i > 0$ and inactive firm has $w_i = 0$. Firms' marginal cost is $\theta_i = \gamma \exp(-w_i + 4)$, which is decreasing in w_i, so you can think of w_i as firm i's efficiency level.

Assume that the firms are symmetric and anonymous. Then, we can also express the state faced by a particular firm by (w, j), where $w \in \mathcal{W}^* \equiv \{(w_1, w_2, w_3) | w_i \in \mathcal{W}, w_1 \geq w_2 \geq w_3\}$ and $j \in \{1, 2, 3\}$. w represents the industry structure. j represents the position of the firm in the vector w.[1]

The firms play an infinite horizon discrete time dynamic oligopoly game. Here is the timeline of the game: in each period,

1. Firms observe their state variables w and j.
2. **Exit**: Incumbent firms know the constant scrap value $\phi = 0.1$.[2] If an incumbent's expected continuation value from staying in the game is lower than or equal to the scrap value ϕ, they will receive the scrap value and exit forever (become inactive). Otherwise, they will stay in the game.
3. **Entry**: One short-lived potential entrant learns its entry cost x_e this period, which is distributed *uniformly* between $[x_{el}, x_{eh}]$. If the potential entrant enters, it will have initial

[1] For example, a state represented by $w = (12, 5, 1)$ and $j = 2$ means the firm has efficiency level 5 and faces two rivals with efficiency levels 12 and 1.

[2] Here we assume the scrap value to be non-random for simplicity.

efficiency level $w_e = 4$ (which may still be subject to industry aggregate shock, described below). The potential entrant enters if the expectation continuation value of entering exceeds the entry cost x_e.

4. **Investment and industry aggregate shock**: Each incumbent firm i face a probability of receiving a binary efficiency gain τ_i next period. Incumbent firm i invests x_i to raise the probability of $\tau_i = 1$. All incumbent firms also face a common exogenous binary aggregate shock ν next period. Denote w_i' as the efficiency next period. Then, the transition probability of firm i's efficiency w_i is

$$w_i' = w_i + \tau_i - \nu$$
$$Prob(\tau_i) = \begin{cases} \frac{ax}{1+ax}, & \text{if } \tau_i = 1 \\ \frac{1}{1+ax}, & \text{if } \tau_i = 0 \end{cases}$$
$$Prob(\nu) = \begin{cases} \delta, & \text{if } \nu = 1 \\ 1-\delta, & \text{if } \nu = 0 \end{cases}$$

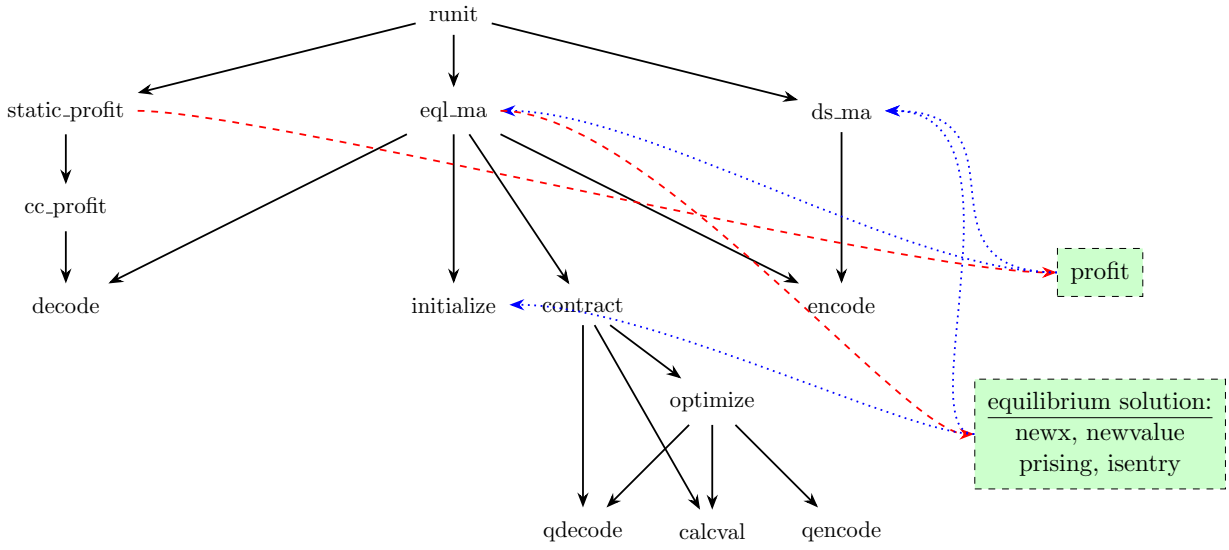
5. **Current profit**: Incumbent firms that have not exited optimally choose their output q_i to maximize their profit, i.e., they play Cournot competition with differentiated marginal cost. Then, incumbent firms realize their profits.

6. **State transition**: The exit, entry and investment outcomes, as well as and the industry aggregate shock are realized. The state variables are updated accordingly.

Now you will complete the code that computes the equilibrium for this game.

2 Code Framework

The Dropbox folder includes an incomplete code framework. Here is an overview of the structure of the code.



All the node in text are functions (e.g., `runit`). All the nodes in rectangles on the right are variables to be saved and read (e.g., `profit`). Black solid arrows indicate that a function calls another

function (e.g., `runit` calls `static_profit`). Red dashed arrows indicate that a function saves a variable (e.g., `static_profit` saves variable `profit`). Blue dotted arrows indicate that a variable is read by a function (e.g., `profit` is read by function `eql_ma`).[3]

Here is a description of the functions. All codes are provided except for the ones marked with task, which are incomplete and your main tasks for this problem set.

- `runit` (**main function**): takes parameter values and numerical settings such as tolerance level and number of simulations and runs the rest of the code.
- `static_profit`: loops the number of active firms from 1 to N to compute static profit and market share
  - `cc_profit`: for a given number of active firms, computes the profit and market share for the static Cournot competition for each firm in all states.
- `eql_ma`: computes the equilibrium with maximum N active firms. Note that this code imposes an implicit equilibrium selection criteria: the policy function and value function in a game with $n+1$ players is initialized with the equilibrium strategy with n players.
  - `initialize`: initializes the value and policy functions for a given n and $\bar{w}$. It uses the value function and strategy from the $n-1$-active-firm game to initialize the game with n-active-firm game. For $n=1$, it initializes with some ad hoc numbers.
  - `decode`: task takes a state code $\in \mathbb{N}^+$ and decodes it into a $w \in \mathcal{W}^*$ (note that w is a weakly descending N-tuple).
  - `encode`: task takes a $w \in \mathcal{W}^*$ and encodes it into a state code $\in \mathbb{N}^+$.
  - `contract`: performs one iteration on policy and value function. It first loops over all states to update entry decisions, and then loops over all states to update investment decisions and value function.
    - `qdecode`: quickly decodes using pre-computed decode table.
    - `qencode`: quickly encodes using pre-computed encode table and multiplier.
    - `optimize`: task loops over firms to calculate optimal investment and value function, for a given state w
    - `calcval`: task calculates the continuation value for increasing the efficiency level and staying at efficiency level for a given state w.
- `ds_ma`: task simulates forward the game for a specified number of times from an given initial state.

3 Your Tasks

1. Complete `decode` and `encode` functions. (Hint: both functions use the matrix `binom`, which is a matrix of binomial coefficients pre-computed in `static_profit, eqm_ma` and `ds_ma`, such that `binom(i,j)`$= \binom{i-1}{j-2}$. Therefore, `binom(N+`$\bar{w}$`+1,`$\bar{w}$`+2)`$= \binom{N+\bar{w}}{\bar{w}}$, which is $|\mathcal{W}^*|$.)

[3] There are two additional functions created for convenience, `maxind` and `minind`, which output the indices that maximizes and minimizes a given array.

To check if the `decode` function is working properly, print a table of dimension $N \times |\mathcal{W}^*|$ for $N = 3$ and $\bar{w} = 3$, such that $\forall i$, column i represents a weakly descending 3-tuple that would be encoded as state code i. As an example, the output for $N = 3$ and $\bar{w} = 2$ is

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 2 \end{bmatrix}$$

2. Complete `calcval` function. (Hint: I include the code to compute the value function for the case where the firm stays at the same efficiency level. Complete the case where the firm moves up in efficiency level.)

3. Complete `optimize` function. Impose another equilibrium selection criteria in this function: if firm i with efficiency level w_i exits, for all firm j such that $w_j \leq w_i$, firm j exits too. (Hint: I include a description of the steps you need to code up in the comments of the code.)

4. Now you should have everything you need to simulate forward the game! Complete `ds_ma` function. Use this function to compute equilibrium for two cases where entry cost is uniformly distributed between a) $[0.15, 0.25]$ b) $[0.01, 0.11]$. Start from an industry state $(6, 0, 0)$ and simulate forward 10000 periods. Compute average number of active firms and average total investment per period for both of these cases.

For your convenience, I also marked all the tasks in the code with the following.

```
%%%%%%%%Task%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%
```

Good luck! Feel free to drop by my office hour if you have any questions.

A Dictionary for variables

Variable Name	Description	Model
c.MAX_FIRMS, nfmax, nfirms, rlnfirms	Max number of active firms	N
c.KMAX, kkmax, kmax	Max efficiency level	$\bar{w}$
c.START_FIRMS, stfirm	Starting number of firms for equilibrium computation	
c.ENTRY_LOW, x_entryl	Uniform distribution support: lower bound	x_{el}
c.ENTRY_HIGH, x_entryh	Uniform distribution support: upper bound	x_{eh}
c.SCRAP_VAL, phi	Scrap value	ϕ
c.ENTRY_AT, entry_k	Efficiency level at which new firms enter	w_e
c.BETA, beta	Discount factor	
c.DELTA, delta	Probability of industry aggregate decline	δ
c.INV_MULT, a	Investment cost parameter	a
c.INTERCEPT, D	Cournot demand intercept	
c.FIXED_COST, f	Cournot fixed cost	
c.GAMMA, ggamma	Cournot marginal cost coefficient	γ
c.TOL, tol	Tolerance level for convergence	
c.PROFIT_DONE	Indicator for having finished computing profit	
c.EQL_DONE	Indicator for having finished equilibrium computation	
c.PREFIX	Prefix representing Cournot competition in saved results	
c.DS_WSTART	Initial state for simulation	
c.DS_NSIMX	Number of simulation periods	
binom	Binomial coefficient matrix for encoding/decoding n-tuples	
descn, wmax	Number of descending n-tuples or industry states	$\lvert\mathcal{W}^*\rvert$
profit	Profit for each state-firm	
w	State code or decoded state vector	w
locw	Decoded state vector for firm efficiency levels	w
theta	Marginal cost	θ
mask	Matrix of all possible binary outcomes of rivals	
dtable	Decode table mapping state code to state tuple	
multfac	Multiplication factor for encoding without sorting	
wtable	Table of industry states before encoding	
etable	Encode table mapping state code before symmetry to state code after symmetry	
oldvalue, newvalue, oval, nval, nval_t	Value function during equilibrium iteration (o = old, n = new)	
oldx, newx, ox, x, nx, nx_t	Policy function during equilibrium iteration (o = old, n = new)	
isentry	Entry probability	
norm, avgnorm	Convergence metrics for equilibrium iteration	
prising	Probability of efficiency level increase due to investment	
place, j	Firm's position in the state vector	j
justone	Dummy vector used for tracking firm's position in the state vector	
locmask	Adjusted mask for firm's position in state space	
p_up	Probability of moving up in efficiency level	
z1, z2	Lower and upper bounds for efficiency levels	
val_up	Value of moving up in efficiency level	
val_stay	Value of staying at the same efficiency level	
probmask	Transition probability of a given rivals' investment outcome	
d, e	New states after private and aggregate shocks	
pl1	Position of the firm in the new state after sorting	
locwx, locwe	Updated state vectors considering exit and entry	

Table 1: List of variables