Solve Hopenhayn (1992) Using MATLAB

Yijiang Zhou

Department of Economics
The Chinese University of Hong Kong

March 13, 2021

Contents

Introduction and useful resources

- Hopenhayn (1992): workhorse model in firm and industry dynamics
- Lecture slides by Prof. Chris Edmond available at his website.
  - Lecture 2 and 3 are essential for understanding the model and solution algorithm.
- Original MATLAB program by Prof. Alessandro Ruggieri available at his GitHub page.
- Four MATLAB code files needed: `main.m`, `solve_vfi.m`, `compute_parameters.m` and `compute_statespace.m`.
- Detailed code notes on my GitHub page.

Initialize model parameters

- Firm's production function (labor n is the only input):

$$y = zn^{\theta} \tag{1}$$

- Log of productivity z follows a $AR(1)$ process:

$$\ln z_t = a + \rho \ln z_{t-1} + \epsilon_t \tag{2}$$

  $\sigma^2_{\ln z}$ calculated for later use given σ^2_{ϵ}.
- Market demand is $D(p) \equiv D - p$ with D exogenously given.
- Discount rate β, fixed cost of operating c^f and entry cost c^e.

Initialize state space

- Discretize z: assume $\ln z$ follows a 21-state Markov chain on $\ln z_i$.
  - $\ln z_i$ are evenly spaced along the real line with $\ln z_1 < \ln z_2 < \ldots < \ln z_{21}$ and $\ln z_1 = a - m\sigma_{\ln z}$, $\ln z_{21} = a + m\sigma_{\ln z}$.
- Calculate transition probability matrix for $\ln z_i$ using Tauchen's method: let $w = \ln z_i - \ln z_{i-1}$ and the transition probability π_{ij} is defined by:

$$\pi_{ij} = \begin{cases} \Phi\left(\frac{\ln z_1 + w/2 - \rho \ln z_i}{\sigma_\epsilon}\right) & \text{for } j = 1 \\ \Phi\left(\frac{\ln z_j + w/2 - \rho \ln z_i}{\sigma_\epsilon}\right) - \Phi\left(\frac{\ln z_j - w/2 - \rho \ln z_i}{\sigma_\epsilon}\right) & \text{for } 1 < j < 21 \\ 1 - \Phi\left(\frac{\ln z_1 - w/2 - \rho \ln z_i}{\sigma_\epsilon}\right) & \text{for } j = 21 \end{cases} \tag{3}$$

with Φ corresponding to the standard normal CDF.

Initialize state space

- Transition probability π defined as:

$$\pi_{ij} \equiv \mathrm{Prob}(z_t = z_j | z_{t-1} = z_i) \tag{4}$$

- Initial productivity drawn from uniform distribution $g(z)$
- Discretize n: assume there are 251 grid points from 0 to 5000 for employment n.

Code for initializing z space

```
%% Grid for tech shock, z,
% by using Tauchen's method of finite state Markov approximation
m=3;
w=2*m*sigma/(Z-1);
lnz=(-m*sigma+a):w:(m*sigma+a);
z=exp(lnz);

%Markov transition matrix
p=zeros(Z);

%See formula on notes
p(:,1)=normcdf(((lnz(1)+w/2)*ones(Z,1)-ro*lnz')/stde,0,1);
p(:,Z)=ones(Z,1)-normcdf(((lnz(Z)-w/2)*ones(Z,1)-ro*lnz')/stde,0,1);
for j=2:(Z-1)
    p(:,j)=normcdf(((lnz(j)+w/2)*ones(Z,1)-ro*lnz')/stde,0,1)-...
        normcdf(((lnz(j)-w/2)*ones(Z,1)-ro*lnz')/stde,0,1);
end
```

Bisection on p

- Given price level p and employment choice n, we can write firm's profit f as:

$$f_i(p) = f(z_i, p, n) = z_i n^\theta p - n - c^f \tag{5}$$

where wage is normalized to 1.

- Bellman equation for incumbent firm is then:

$$v_i(p) = f_i(p) + \beta \max \left[0, \sum_{j=1}^{21} v_j(p) \pi_{ij} \right] \tag{6}$$

Bisection on p

- Free entry condition:

$$v^e(p) \equiv \sum_{i=1}^{21} v_i(p) g_i = c^e \tag{7}$$

- Easy to show $v^e(0) < 0$ and $v^e(p)$ is monotonically increasing in p.
- Given c^e, we can find the unique solution p^* using bisection.
  - Need to know $\boldsymbol{v}^*(p)$ for each p. This is done by VFI.
  - Guess an initial p_0 between p_{min} and p_{max}, solve $v^*(p)$ with VFI, calculate $v^e(p_0)$ and compare to c^e.
  - Update new guess p_{j+1} as:

$$p_{j+1} = \begin{cases} \frac{p_j + p_{max}}{2} & \text{if } v^e(p_j) < c^e \\ \frac{p_{min} + p_j}{2} & \text{if } v^e(p_j) \geq c^e \end{cases} \tag{8}$$

  - Stop when $\frac{|v^e(p) - c^e|}{c^e} \leq 10^{-6}$.

Code for bisection

```
%% Iterate over price of goods
d=1;
% Boundaries for price
pmin=0.01;
pmax=100;

while d>toler
    % Guess prices
    price=(pmin+pmax)/2;

    % Solve firm value function iteration
    [vinitial,dr,exit] = solve_vfi(price,z,Z,n,N,theta,beta,cf,p);

    %Compute the decision rule for labor
    decrule=zeros(1,Z);
    for i=1:Z
        decrule(i)=n(dr(i));
    end

    %Define expected value of entrant
    value=inidis*vinitial';

    % Update price till EV=ce
    if value<ce
     pmin=price;
    else
     pmax=price;
    end

    % Check convergence
    d=abs(value-ce)/ce;
end
```

Value function iteration

- Write equation (6) into matrix form as:

$$\boldsymbol{v}^{k+1}(p)^\top = \boldsymbol{f}(p)^\top + \beta \max\left[\mathbf{0}, \mathbf{\Pi} \cdot \boldsymbol{v}^k(p)^\top\right] \equiv T\left((\boldsymbol{v}^k)^\top, p\right) \quad (9)$$

  k denotes number of iterations. Initial $\boldsymbol{v}^0$ is set to be $\mathbf{0}$.

- For every z and p, firm chooses optimal n^* that maximizes value function, i.e., elements of vector $\boldsymbol{v}^{k+1}(p)$ are obtained by solving:

$$v_i^{k+1}(p) = \max_n \left[z_i n^\theta p - n - c^f + \beta \max\left(0, \sum_{j=1}^{21} v_j^k(p)\pi_{ij}\right)\right] \quad (10)$$

Value function iteration

- Solving it yields $\boldsymbol{v}^{k+1}(p)$, policy $\boldsymbol{n}(p)$ and exit rule $\boldsymbol{x}(p)$ with elements (0 for exit and 1 for continue):

$$x_i(p) = 1 - \mathbb{1}\left(\sum_{j=1}^{21} v_j^k(p)\pi_{ij} < 0\right) \tag{11}$$

- T in equation (9) is a contraction mapping. Iterating on T given initial guess $\boldsymbol{v}^0$ yields:

$$T\left((\boldsymbol{v}^k)^\top, p\right) = \boldsymbol{v}^{k+1}(p)^\top \to \boldsymbol{v}^*(p)^\top \quad \text{as } k \to \infty \tag{12}$$

- The program iterates on T until

$$\frac{||\boldsymbol{v}^{k+1} - \boldsymbol{v}^k||}{||\boldsymbol{v}^k||} \leq 10^{-8}$$

Code for VFI

```
function [vrevised,dr,exit] = solve_vfi(price,z,Z,n,N,theta,beta,cf,p)

%Value function iteration
d=1;
toler=1e-08;

% Store policy function
dr   =zeros(1,Z);     %record for policy function
exit=zeros(1,Z);     %record for exit decision

% Guess value functions
vinitial=zeros(1,Z);
vrevised=zeros(1,Z);

% Fixed cost matrix
cost=cf*ones(N,1)';

while d>toler
    for i=1:Z
        fi = z(i)*n.^theta.*price - n - cost;
        [vrevised(i),dr(i)]=max(fi+beta*max(p(i,:)*vinitial',0)*ones(N,1)');
        exit(i)=1-1*(p(i,:)*vinitial'<0);
    end
    d=norm(vrevised-vinitial)/norm(vrevised);
    vinitial=vrevised;
end

end
```

Distribution of incumbent firms

- In the previous section, we have solved p^* and $\boldsymbol{x}(p^*)$.
- $\mu_{it} = \mu_t(z_i)$ denotes measure of firms with productivity z_i at t. It is also the element in $\boldsymbol{\mu}_t$, which evolves according to:

$$\boldsymbol{\mu}_{t+1}^\top = \Psi\left(p^*\right)\boldsymbol{\mu}_t^\top + m\mathbf{g}^\top \equiv U(\boldsymbol{\mu}_t^\top, p^*), \quad t = 0, 1, \ldots \tag{13}$$

- $\boldsymbol{\mu}$ is NOT a probability distribution vector.
- $\Psi\left(p^*\right)$ has elements

$$\psi_{ij}\left(p^*\right) = x_j\left(p^*\right)\pi_{ji}, \quad i, j = 1, \ldots, n \tag{14}$$

- Easy to see $\psi_{ij}\left(p^*\right) = 0$ if firm exits with productivity draw z_j at t.
- Measure of firms at grid point z_i at $t+1$ depends on transition probabilities and exit decisions of incumbents at t and flow of new entrants.

Distribution of incumbent firms

- Given m, we can calculate the stationary distribution $\boldsymbol{\mu}$ as:

$$\boldsymbol{\mu}^\top = m \left(I - \Psi\left(p^*\right)\right)^{-1} \mathbf{g}^\top \equiv \boldsymbol{\mu}\left(m, p^*\right)^\top \tag{15}$$

- Or: assume initial distribution $\boldsymbol{\mu}^0 = \mathbf{g}$ and iterate on mapping U.
- Either way, $\boldsymbol{\mu}$ is linear in m can be written into

$$\boldsymbol{\mu}(m, p^*) = m \times \boldsymbol{\mu}(1, p^*)$$

Mass of entry

- How do we know m?
- Market clearing condition:

$$Y\left(m,p^{*}\right)=\sum_{i=1}^{21}y_{i}\left(p^{*}\right)\mu_{i}\left(m,p^{*}\right)=D\left(p^{*}\right) \tag{16}$$

- Hence:

$$m^{*}=\frac{D\left(p^{*}\right)}{Y\left(1,p^{*}\right)}=\frac{D\left(p^{*}\right)}{\sum_{i=1}^{21}y_{i}\left(p^{*}\right)\mu_{i}\left(1,p^{*}\right)} \tag{17}$$

- Substituting m^{*} into equation (15) gives $\boldsymbol{\mu}^{*}$.

Exit threshold

- Using policy function $\boldsymbol{x}(p)$ derived from the last section, we can calculate the exit threshold of productivity as:

$$z\left(p^{*}\right)=z_{i^{*}}, \quad i^{*} \equiv \min_{i}\left[\sum_{j=1}^{n} v_{j}\left(p^{*}\right) \pi_{ij} \geq 0\right]$$

- Firms whose productivity draws are below z_{i^*} exit the market in the equilibrium.

Full stationary equilibrium

- Full stationary equilibrium: $(p^*, z^*, \boldsymbol{\mu}^*, m^*)$
  - p^*: price level for goods
  - z^*: exit threshold of productivity
  - $\boldsymbol{\mu}^*$: distribution of incumbent firms
  - m^*: mass of entrants
- They correspond to a steady-state of the dynamic system implied by the perfect foresight equilibrium $\{p_t, z_t, \boldsymbol{\mu}_t, m_t\}_{t=0}^{\infty}$.

Code for calculating $\mu(1, p^*)$

```
%Given the value function and policy function, iterate on industry structure
%until it converges
d=1;
muinitial=inidis;
while d>toler
    muexit=muinitial.*exit;                     %exit decision
    mustay=muexit*p;                            %update for the incumbents stay
    muentry=mustay+inidis;                       %entry
    murevised=muentry./sum(muentry);
    d=norm(murevised-muinitial)/norm(murevised);
    muinitial=murevised;
end
```

The author made a mistake...

- ...by scaling `muentry`, which is $\boldsymbol{\mu}_{t+1}$, using the sum of its elements.
- This makes no sense.
  - $\boldsymbol{\mu}$ is not a probability distribution vector.
  - No need to scale it before calculating "distance" between $\boldsymbol{\mu}_{t+1}$ and $\boldsymbol{\mu}_t$ after each iteration.
- It has been tested, that the corrected code (next page) yields the same $\boldsymbol{\mu}(1, p^*)$ as that produced by equation (15).

Corrected code for calculating $\mu(1, p^*)$

```
% mistake-free iteration
while d>toler
    muexit=muinitial.*exit;                      %exit decision
    mustay=muexit*p;                             %update for the incumbents stay
    muentry=mustay+inidis;                        %entry
    murevised=muentry;
    d=norm(murevised-muinitial)/norm(murevised);
    muinitial=murevised;
end
```

Corrected code for computing m^*

```
%% Calculating the entry mass M
% Using equilibrium condition in goods market
% mistake-free
murevised_sc = murevised ./ sum(murevised);

y=D-price;
Xstar=z(Z-sum(exit));
Pstar=price;
Size = (decrule)*murevised_sc';
Y=(decrule.^theta.*z)*murevised';
Mstar=y/Y;
Exrate=sum(murevised_sc(1:Z-sum(exit)))*100;
```

Summary

Solve Hopenhayn model

- Step 1: use free entry condition (7) and monotonicity of $v^e(p)$ to construct a bisection on price p.
  - Rely on VFI to compute $\boldsymbol{v}^*(p)$, $\boldsymbol{n}(p)$ and $\boldsymbol{x}(p)$ for each p.
- Step 2: given p^* and $\boldsymbol{x}(p^*)$, use mapping U in equation (13) to compute $\boldsymbol{\mu}(1, p^*)$, and equation (17) for m^*.
- Step 3: obtain z^* using exit decision $\boldsymbol{x}(p^*)$.

Extras

- Changing initial distribution $g(z)$: only $\boldsymbol{\mu}^*$ changes.
- Changing transition probability matrix to identity matrix: failed to solve $\boldsymbol{\mu}^*$.
- Increase entry cost c^e: price p^* increases and productivity threshold z^* decreases. Weaker election effect.
- Decrease fixed cost c^f: p^* decreases. How about z^*?