Notes on Hopenhayn (1992) MATLAB Code

Yijiang Zhou *

March 13, 2021

1 Introduction

In this note, I briefly introduce the MATLAB program that solves the firm dynamics model in Hopenhayn (1992). The program is composed by Professor Alessandro Ruggieri and available at his GitHub page. The method of solving Hopenhayn (1992) model generally follows the discrete state dynamic programming framework, on which Professor Chris Edmond provided some wonderful illustration in his lecture slides. Readers are strongly encouraged to go through the slides of lecture 2 and lecture 3 before reading this note[1].

There are four MATLAB code files needed to solve the Hopenhayn (1992) model: `main.m`, `solve_vfi.m`, `compute_parameters.m` and `compute_statespace.m`. To execute the program, please put the code files in one folder and run `main.m`. The execution takes less than 5 seconds and the results will be listed in the output window. In the following sections, I will provide detailed explanation on the code files one by one, and how they calculate the price, exit threshold of productivity, mass of entrants and distribution of incumbent firms in a stationary equilibrium.

2 Initiate parameters and state space

In the beginning of `main.m`, the program calls the other two code files `compute_parameters.m` and `compute_statespace.m` to initiate parameters and state space for the model. Specifically, the firm's production function is assumed as follows:

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

where z and n denote productivity and employment, respectively and $\theta = 0.64$. The log of productivity follows a $AR(1)$ process:

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

where $a = 0.37$ and $\rho = 0.93$. Given σ_ϵ^2, the variation of $\ln z$ ($\sigma_{\ln z}^2$) is calculated for later use. The market demand function is $D(p) \equiv D - p$ and D is exogenously given. The file `compute_parameters.m` also lays out other parameters in the model, including discount rate β, fixed cost of operating c^f and entry cost c^e.

Since productivity z and employment n are both continuous variables, we need to discretize them using `compute_statespace.m`. For productivity z, the author assumes the log of it follows

*Department of Economics, the Chinese University of Hong Kong. Email: yijiangzhou@link.cuhk.edu.hk

[1]The most updated version of this note can be found on my GitHub page.

a 21-state Markov chain on $\ln z_i$ and computes transition probability matrix using Tauchen's method. The grid points $\ln z_i$ are evenly spaced along the real line with $\ln z_1 < \ln z_2 < ... < \ln z_{21}$ and

$$\ln z_1 = a - m\sigma_{\ln z};\ \ln z_{21} = a + m\sigma_{\ln z}$$

Let $w = \ln z_i - \ln z_{i-1}$ and the transition probability π_{ij} is defined as follows, with Φ corresponding to the standard normal CDF:

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

Note that the subscript for transition probability π is defined in the same way as that in Professor Edmond's slide, where we have:

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

The code for defining state space for z is as follows:

```
%% 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
```

The author also assumes that initial productivity is drawn from uniform distribution $g(z)$ (the resulting probability for each grid point is collected by vector `inidis` in code file), and there are 251 grid points from 0 to 5000 for employment n. With the above settings in place, we can proceed to solve for stationary equilibrium of the model.

3 Solve for price and exit rule

In a standard Hopenhayn (1992) model, we can write firm's profit f as follows, given the price level p and employment choice n:

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

where wage is normalized to 1. The Bellman equation for incumbent firm is then

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

With the Bellman equation, we can write the free entry condition as follows, which states that the expected profit of entry equals entry cost:

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

It is easy to show that $v^e(0) < 0$ and $v^e(p)$ is monotonically increasing in p. This means when c^e is given, we can find the unique p^* that solves equation (7), relying on bisection method. To implement bisection, it is necessary that we know the value function vector $\boldsymbol{v}^*(p)$ for each p that the program iterates on. This is done by value function iteration (VFI). The code for this part is presented below.

```
%% 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
```

As can be seen from the code block, the author applies the standard bisection method by guessing an initial p_0 which lies in the middle between p_{min} and p_{max}. The function `solve_vfi` then solves value function vector $\boldsymbol{v}^*(\cdot)$ for p_0 (more on this later). Expected profit of entry $v^e(p_0)$ is calculated and compared to c^e. For $j = 0, 1, 2, ...$, the new guess p_{j+1} is then updated based on the result of comparison as:

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

This process continues until $v^e(p)$ and c^e are sufficiently close, i.e. it stops when:

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

The author-defined function `solve_vfi` employs VFI technique to solve $\boldsymbol{v}^*(p)$ for every p. To see how VFI works, we write equation (6) into matrix form as follows, where $\boldsymbol{\Pi}$ denotes the 21×21 transition matrix, and both $\boldsymbol{v}$ and $\boldsymbol{f}$ are 1×21 vectors:

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

The superscript k in equation (9) denotes number of iterations, and the initial $\boldsymbol{v}^0$ is set to be $\mathbf{0}$. Note that for every productivity draw z_i and price level p, the firm chooses the optimal number of employment n^* which maximizes the value function. That is to say, the elements of vector $\boldsymbol{v}^{k+1}(p)$ are obtained by solving the following maximization problem:

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

Solving the maximization problem would yield $\boldsymbol{v}^{k+1}(p)$ and optimal productivity-price-specific employment n^*, whose position among employment grid points is written into price-specific policy vector $\boldsymbol{n}(p)$. The exit decisions (0 for exit and 1 for continue) given p are collected into a vector $\boldsymbol{x}(p)$ with elements:

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

It can be proved that the T in equation (9) is a contraction mapping. Therefore, iterating on T given initial guess $\boldsymbol{v}^0$ yields:

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

In practice, the program iterates on T until:

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

This completes the VFI process. The code for function `solve_vfi` is presented as follows, which returns value function $\boldsymbol{v}^*(p)$, policy vector $\boldsymbol{n}(p)$ and exit rule $\boldsymbol{x}(p)$.

```
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
```

The above VFI technique, incorporated into bisection on p, pins down the equilibrium price level p^*. Substituting it into $\boldsymbol{x}(p)$ gives optimal exit rule, which would help us calculate the exit threshold of productivity later.

4 Solve for firm distribution and entry mass

With equilibrium price level p^* and optimal exit rule $\boldsymbol{x}(p^*)$ at hand, we are ready to solve for distribution of incumbent firms $\boldsymbol{\mu}^*$ and mass of entry m^* in the stationary equilibrium. One particularly important thing to note before we proceed, is that $\boldsymbol{\mu}$ is NOT a probability distribution vector. The elements of $\boldsymbol{\mu}$ are simply mass of firms at each productivity grid point. This also means that the sum of all elements in $\boldsymbol{\mu}$ does not necessarily equal to 1.

Let $\mu_{it} = \mu_t(z_i)$ denote measure of firms with productivity z_i at t. It is also the element in firm distribution vector $\boldsymbol{\mu}_t$. The 1×21 vector $\boldsymbol{\mu}_t$ evolves according to the following equation:

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

where $\mathbf{g}$ collects $g_i = g(z_i)$, the initial distribution of productivity. The 21×21 coefficient matrix $\Psi\left(p^*\right)$ has elements

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

and it is easy to see $\psi_{ij}\left(p^*\right) = 0$ if firm exits with productivity draw z_j at t.

Equation (13) indicates that 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. Suppose m is given, we can calculate the stationary distribution $\boldsymbol{\mu}$ as follows:

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

This can also be done by iterating on mapping U, assuming initial distribution $\boldsymbol{\mu}^0 = \mathbf{g}$. Either way, the $\boldsymbol{\mu}$ we obtained is linear in m can be written into $\boldsymbol{\mu}(m, p^*) = m \times \boldsymbol{\mu}(1, p^*)$. How do we know the value of m in a stationary equilibrium? The answer is to rely on goods market clearing condition below:

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

where y_i is the output defined in equation (1) with productivity z_i. We can derive from equation (16) that:

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

The above equation delivers the mass of entry m^* in a stationary equilibrium. Substituting it into equation (15) gives equilibrium distribution of firms $\boldsymbol{\mu}^*$. Using the policy function $\boldsymbol{x}(p)$ derived from the last section, we can calculate the exit threshold of productivity as follows:

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

Firms whose productivity draws are below z_{i^*} exit the market in the equilibrium. This completes the characterization of a full stationary equilibrium, consisting of $(p^*, z^*, \boldsymbol{\mu}^*, m^*)$. It corresponds to a steady-state of the dynamic system implied by the perfect foresight equilibrium $\{p_t, z_t, \boldsymbol{\mu}_t, m_t\}_{t=0}^{\infty}$ in Hopenhayn (1992).

The origin code for calculating $\boldsymbol{\mu}\left(1, p^*\right)$ is presented in the following. It iterates on mapping U in equation (13) and assumes initial distribution $\boldsymbol{\mu}^0 = \mathbf{g}$. Note that the author made a **mistake** by scaling `muentry`, which is $\boldsymbol{\mu}_{t+1}$, using the sum of its elements. This makes no sense, as $\boldsymbol{\mu}$ is not a probability distribution vector and there is no need to scale it before calculating "distance" between $\boldsymbol{\mu}_{t+1}$ and $\boldsymbol{\mu}_t$ after each iteration.

```
%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 corrected code for calculating $\boldsymbol{\mu}\left(1, p^*\right)$ is presented below. It has been tested, that the $\boldsymbol{\mu}\left(1, p^*\right)$ we obtained from the mistake-free iteration below is identical to that produced using equation (15) and setting m equal to 1.

```
% 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
```

Finally, the code listed below computes m^* using equation (17). It also presents exit threshold z^*, average size of firms and exit rate in the equilibrium. The author made a second **mistake** by adding $\sum_{i=1}^{21} y_i(p^*)g_i$ in the denominator when calculating m^* according to equation (17). As successful entrants would only start to produce in the next period, including their output $\sum_{i=1}^{21} y_i(p^*)g_i$ in the denominator (total output in this period) is clearly wrong. This mistake and other mistakes associated with using falsely-calculated $\boldsymbol{\mu}\left(1, p^*\right)$ have been corrected in the code below.

```
%% 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;
```

As a brief summary, the breakthrough step of solving a standard Hopenhayn (1992) model is using the monotonicity of $v^e(p)$ in equation (7), the free entry condition, to construct a bisection strategy that approaches the equilibrium price p^*. In this process, we rely on VFI to compute value function vector $\boldsymbol{v}^*(p)$, employment choice $\boldsymbol{n}(p)$ and exit rule $\boldsymbol{x}(p)$ for each p. Given p^* and $\boldsymbol{x}(p^*)$, we utilize the mapping U in equation (13) to compute $\boldsymbol{\mu}(1, p^*)$, and equation (17) for m^*. Taking the final trivial step of obtaining z^*, we arrive at the full stationary equilibrium $(p^*, z^*, \boldsymbol{\mu}^*, m^*)$. It definitely takes time and effort to understand and replicate all procedures of solving the Hopenhayn (1992) model, even after carefully reading Professor Edmond's slides and the above MATLAB codes. And it is my hope, that this note can be of assistance throughout the learning.

References

Hopenhayn, H. A. (1992). Entry, Exit, and Firm Dynamics in Long Run Equilibrium. *Econometrica*, *60*(5), 1127.