The Cyclical Behavior of Equilibrium Unemployment and Vacancies
Shimer (2005)

Presented by: XING Mingjie

December 5, 2023

Bellman Equation

Worker

- Unempolyment value

$$U_p = z + \delta\{f(\theta_p)\mathbb{E}_p W_{p'} + (1 - f(\theta_p))\mathbb{E}_p U_{p'}\} \qquad (1)$$

- Employment value

$$W_p = w_p + \delta\{(1 - s)\mathbb{E}_p W_{p'} + s\mathbb{E}_p U_{p'}\} \qquad (2)$$

Bellman Equation

Firm

- Hiring value

$$J_p = p - w_p + \delta(1 - s)\mathbb{E}_p J_{p'} \tag{3}$$

- Vacancy value

$$V_p = -c + \delta q(\theta_p)\mathbb{E}_p J_{p'} \equiv 0 \tag{4}$$

Productivity

The log of productivity follows AR(1) process

$$\log(p') = \rho \log(p) + \varepsilon \tag{5}$$

where

$$\log(p) \sim N(\mu_\lambda, \sigma_\lambda^2), \ \varepsilon \sim N(\mu_\varepsilon, \sigma_\varepsilon^2)$$

Optimal Control

Market tightness

- Control in this problem consists of w_p, θ_p, u_p and the state is p
- Market tightness θ_p is given by solving the following equation of hire rate from free entry condition

$$q(\theta_p) = \frac{c}{\delta \mathbb{E}_p J_{p'}} \tag{6}$$

- Employ Rate is given by

$$f(\theta_p) = \mu^{\frac{1}{\eta}} q^{\frac{\eta-1}{\eta}} \tag{7}$$

- And market tightness

$$\theta_p = \frac{f(\theta_p)}{q(\theta_p)} \tag{8}$$

Optimal Control

Continued

- Optimal wage at each productivity level is given by the Nash Bargaining:
$$W_p - U_p = \beta(W_p - U_p + J_p) \tag{9}$$
- Note Bellman Equation of W_p given by 2, U_p given by 1, J_p given by 3
- Following the algebra given in slide 30, optimal wage for each p is
$$w_p = \beta p + (1-\beta)z + \beta c\theta_p \tag{10}$$
- And unemployment rate
$$u_p = \frac{\delta}{\delta + f(\theta_p)} \tag{11}$$

Calibration

Parameter	Symbol	Value
Productivity std.	σ_{logp}	0.05
Productivity mean	μ_{logp}	1
Stochastic std.	σ_{ε}	0.03
Stochastic mean	μ_{ε}	0
Separation rate	s	0.1
Discount rate	r	0.012
Value of leisure	z	0.4
Matching function	μ	1.355
Matching function	α	0.72
Bargaining Power	β	0.72
Cost of vacancy	c	0.213

Table 1: Parameter Calibration

Question a I

Discretization Algorithm

Inspired by Karen A. Kopecky 2006 Lecture Note

1. Choose a relateive error tolerance level `tol`;
2. Discretize the state space by constructing a grid for productivity

   $$p = \exp\{logp\} \text{ where } logp = \{logp_1, logp_2, \ldots, logp_n\}$$

   given by the Tauchen method. The n is chosen at 250; deviation step is 35.
3. Start with an initial guess of the value function $V^{(0)}(p)$ is a vector of length n, i.e., $V^{(0)}(p) = \{V_i^{(0)}\}_{i=1}^{n}$, where $V_i^{(0)} = V^{(0)}(p_i)$. V here represents U, W, J. The initial guess is ones.

Question a II

Discretization Algorithm

4. Update the value function using eqautions 1 to 10, specifically
   4.1 Fix the current productivity level at one of the grid points, p_i from $i = 1$
   4.2 For each possible choice of productivity next period, calculate optimal control in the following order:

$$q(\theta_{p_i}) = \frac{c}{\delta \sum_{j=1}^{n} p_{i,j} J^{(0)}(p_j)}$$

$$f(\theta_{p_i}) = \mu^{\frac{1}{\eta}} q^{\frac{\eta-1}{\eta}}$$

$$\theta_{p_i} = (\frac{q(\theta_{p_i})}{\mu})^{-\frac{1}{\eta}}$$

$$w_{p_i} = \beta p_i + (1-\beta)z + \beta c \theta_{p_i}$$

   4.3 and update the value function system with

Question a III

Discretization Algorithm

$$U^{(1)}_{p_i} = z + \delta\{f(\theta_{p_i})\sum_{j=1}^{n} p_{i,j} W^{(0)}(p_j) + (1 - f(\theta_{p_i}))\sum_{j=1}^{n} p_{i,j} U^{(0)}(p_j)\}$$

$$W^{(1)}_{p_i} = w_{p_i} + \delta\{(1-s)\sum_{j=1}^{n} p_{i,j} W^{(0)}(p_j) + s\sum_{j=1}^{n} p_{i,j} U^{(0)}(p_j)\}$$

$$J^{(1)}_{p_i} = p_i - w_{p_i} + \delta(1-s)\sum_{j=1}^{n} p_{i,j} J^{(0)}(p_j)$$

4.4 Choose a new grid point for productivity, go through 4.1 to 4.3. Once we have done the update for all productivity grid, we have new system of value function $V^{(1)}_p$

4.5 Compute distance between the two systems of value functions following the sup norm

$$d = \max_{i\in\{1,\ldots,n\}} |V^{(0)}_i - V^{(1)}_i|$$

Question a IV

Discretization Algorithm

4.6 If distance is within the error tolerance level, $d \leq tol * ||V_1^{(1)}||$, the functions have converged and go to step 5, or else go back to step 4.

5. Calculate the optimal control for each productivity level:

$$q(\theta^*_{p_i}) = \frac{c}{\delta \sum_{j=1}^{n} p_{i,j} J^*(p_j)}$$

$$f(\theta^*_{p_i}) = \mu^{\frac{1}{\eta}} q^{\frac{\eta-1}{\eta}}$$

$$\theta^*_{p_i} = (\frac{q(\theta^*_{p_i})}{\mu})^{-\frac{1}{\eta}}$$

$$w^*_{p_i} = \beta p_i + (1-\beta) z + \beta c \theta^*_{p_i}$$

$$u^*_p = \frac{\delta}{\delta + f(\theta^*_p)}$$

where J^* is the converged value function.

Tauchen Method

Use `discretizeAR1_Tauchen` function from the Matlab Toolbox of Kirkby (2023) .

Matlab Code I

Discretization Method

```
%% 1) Discretization method
% Initialize
[U_d, W_d, J_d] = deal(ones(max_iter,n)*tol); %Value function
[fstar_d, qstar_d, thetastar_d, wstar_d, ustar_d] = deal(ones(n,1)); % Optimal control
[f, q, theta, w] = deal(ones(n,1)); % Working control
[diffU, diffW, diffJ] = deal(ones(n,1)); % Convergence check

for I = 1:max_iter
    % iterate over productivity grids
    for i = 1:n
```

Matlab Code II

Discretization Method

```matlab
        % Working maximized control
        q(i) = c / (delta_ * expec(J_d,P,I,i));
        f(i) = m^(1/alpha) * q(i)^(1 - 1/alpha);
        theta(i) = f(i)/q(i);
        w(i) = beta_ * p(i) + (1-beta_) * z + beta_ * c * theta(i);

        % Value function iteration
        U_d(I+1,i) = z + delta_ * (f(i) * expec(W_d,P,I,i) + ...
                               (1-f(i)) * expec(U_d,P,I,i));
```

Matlab Code III

Discretization Method

```
        W_d(l+1,i) = w(i) + delta_ * ((1-s) *
            expec(W_d,P,l,i) + ...
                                    s * expec(U_d,
                                        P,l,i));
        J_d(l+1,i) = p(i) - w(i) + delta_ *
            (1-s) * expec(J_d,P,l,i);

        % Convergence
        diffU(i) = abs(U_d(l+1,i) - U_d(l,i));
        diffW(i) = abs(W_d(l+1,i) - W_d(l,i));
        diffJ(i) = abs(J_d(l+1,i) - J_d(l,i));
    end

    % Check convergence
    diff_U = max(diffU);
```

Matlab Code IV

Discretization Method

```
    diff_W = max(diffW);
    diff_J = max(diffJ);
    diff = max([diff_J, diff_W, diff_U]);
    if diff < tol
        break;
    end
end

if I == max_iter
    error('NotConverge');
else
    % For the converged value function, derive the optimal control given p
    for i = 1:n
```

Matlab Code V

Discretization Method

```
            qstar_d(i) = c / (delta_ * expec(J_d,P,l+1,i));
            thetastar_d(i) = (qstar_d(i) / m)^(-1/alpha);
            fstar_d(i) = m^(1/alpha) * qstar_d(i)^(1 - 1/alpha);
            wstar_d(i) = beta_ * p(i) + (1-beta_) * z + beta_ * c * thetastar_d(i);
            ustar_d(i) = delta_ / (delta_ + fstar_d(i));
        end
end

figure
hold on
```

Matlab Code VI

Discretization Method

```
plot(p,fstar_d);
plot(p,wstar_d);
plot(p,ustar_d);
xlabel("Productivity");
legend("Job finding rate", "Wage", "
    Unemployment rate");
title('Job Finding Rate, Wage and Unemployment
     Rate on Productivity by Discretization');
hold off
saveas(gcf,"Discretization.jpg");
```

Question a

Parametric Approximation

0. Choose Hermite interpolation polynomials to approximate $\hat{V}(p; \mathbf{coefs})$ in the form of $f(x) = a(x - x_1)^3 + b(x - x_1)^2 + c(x - x_1) + d$ with Matlab code `pchip`. Report initial paramters with `pp.coefs` for each value function, save as `old`
1. Maximize control and calculate value function at each productivity level as done in Discretization method
2. Fit for new value function system and report parameters and save as `new`
3. If $||\hat{V}(p; \mathbf{coefs\_old}) - \hat{V}(p; \mathbf{coefs\_new})|| < tol$, stop; else go to step 1.

Matlab Code I

Approximation Method

```
%% 2) Parametric Approximation
% Initialize
[U_a, W_a, J_a] = deal(ones(max_iter,n)*tol);
[fstar_a, qstar_a, thetastar_a, wstar_a, ustar_a] = deal(ones(n,1)); % Optimal control
[f, q, theta, w] = deal(ones(n,1));                    % Working control
[diffU, diffW, diffJ] = deal(ones(n,1));             % Convergence check

% We try Hermite Interpolation with pchip
ppU_old = pchip(p, U_a(1,:));
ppW_old = pchip(p, W_a(1,:));
ppJ_old = pchip(p, J_a(1,:));
```

Matlab Code II

Approximation Method

```
for l = 1:max_iter
    U_old = ppval(ppU_old,p);
    W_old = ppval(ppW_old,p);
    J_old = ppval(ppJ_old,p);
    % Maximization
    for i = 1:n
        % Working maximized control
        q(i) = c / (delta_ * expec(J_a,P,l,i))
            ;
        f(i) = m^(1/alpha) * q(i)^(1 - 1/alpha
            );
        theta(i) = f(i)/q(i);
        w(i) = beta_ * p(i) + (1-beta_) * z +
            beta_ * c * theta(i);
```

Matlab Code III

Approximation Method

```
        % Value function iteration
        U_a(l+1,i) = z + delta_ * (f(i) * expec(W_a,P,l,i) + ...
                                     (1-f(i)) * expec(U_a,P,l,i));
        W_a(l+1,i) = w(i) + delta_ * ((1-s) * expec(W_a,P,l,i) + ...
                                     s * expec(U_a,P,l,i));
        J_a(l+1,i) = p(i) - w(i) + delta_ * (1-s) * expec(J_a,P,l,i);
    end
    % Fit new parameter
    ppU_new = pchip(p, U_a(l+1,:));
```

Matlab Code IV

Approximation Method

```matlab
    ppW_new = pchip(p, W_a(l+1,:));
    ppJ_new = pchip(p, J_a(l+1,:));

    % Convergence
    U_new = ppval(ppU_new,p);
    W_new = ppval(ppW_new,p);
    J_new = ppval(ppJ_new,p);
    diffU = norm(U_new - U_old);
    diffW = norm(W_new - W_old);
    diffJ = norm(J_new - J_old);
    diff_a = max([diffU,diffW,diffJ]);
    if diff_a <= tol
        break;
    else
        ppU_old = ppU_new;
```

Matlab Code V

Approximation Method

```
            ppW_old = ppW_new;
            ppJ_old = ppJ_new;
        end
    end

    if l == max_iter
        error('NotConverge');
    else
        % For the converged value function, derive
            the optimal control given p
        coefs = [ppU_new.coefs, ppW_new.coefs,
            ppJ_new.coefs];
        for i = 1:n
            qstar_a(i) = c / (delta_ * expec(J_a,P
                ,l+1,i));
```

Matlab Code VI

Approximation Method

```
            thetastar_a(i) = (qstar_a(i) / m)^(-1/
                alpha);
            fstar_a(i) = m^(1/alpha) * qstar_a(i)
                ^(1 - 1/alpha);
            wstar_a(i) = beta_ * p(i) + (1-beta_)
                * z + beta_ * c * thetastar_a(i);
            ustar_a(i) = delta_ / (delta_ +
                fstar_a(i));
        end
    end

    % print("For the approximation method, the
        coeffecients for U, W and J are given by
        %6.4f", coefs);
    figure
```

Matlab Code VII

Approximation Method

```
hold on
plot(p,fstar_a);
plot(p,wstar_a);
plot(p,ustar_a);
xlabel("Productivity");
legend("Job finding rate", "Wage", "
    Unemployment rate");
title('Job Finding Rate, Wage and Unemployment
      Rate on Productivity by Approximation');
hold off
saveas(gcf,"Approximation.jpg");
```

Optimal Controls from two methods

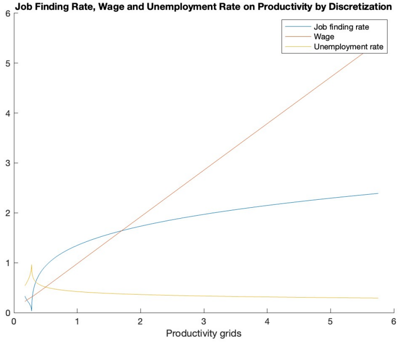

Figure 1: Discretization

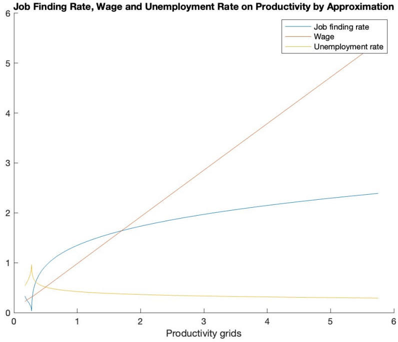

Figure 2: Approximation

Two Results

Question b

To use the polynomial interpolation function from Parametric Approximation directly

log(p)	w	u	f
0.4	1.4446	0.3865	1.5683
0.7	1.9336	0.3627	1.7362
1	2.5922	0.3409	1.9104
1.3	3.4792	0.3206	2.0936
1.6	4.6740	0.3016	2.2879

Table 2: Wage, Unemployment rate and Job Finding Rate

Two Results

Question c

	u	f	p
Data Std.	0.190	0.118	0.020
Approximation Model Std.	0.175	0.697	1.491
Discretization Model Std.	0.175	0.697	

Table 3: Model fit on Unemployment, Job finding rate and productivity

Appendix A

Optimal wage

$$
\begin{gathered}
W_p - U_p = \beta(W_p - U_p + J_p) \\
\Leftrightarrow w_p - z + \delta(1 - s - f(\theta_p))(\mathbb{E}_p W_{p'} - \mathbb{E}_p U_{p'}) = \\
\beta(p - z + \delta(1 - s - f(\theta_p))(\mathbb{E}_p W_{p'} - \mathbb{E}_p U_{p'}) + \delta(1 - s)\mathbb{E}_p J_{p'}) \\
\Leftrightarrow w_p = \beta p + (1 - \beta)z + (\beta - 1)\delta(1 - s - f(\theta_p))(\mathbb{E}_p W_{p'} - \mathbb{E}_p U_{p'}) \\
+ \frac{\beta c(1 - s)}{q(\theta_p)} \\
\Leftrightarrow w_p = \beta p + (1 - \beta)z - \frac{\beta c \delta(1 - s - f(\theta_p))}{q(\theta_p)} + \frac{\beta c(1 - s)}{q(\theta_p)} \\
\Leftrightarrow w_p = \beta p + (1 - \beta)z + \beta c \theta_p
\end{gathered}
$$

where we use the fact that $\mathbb{E}_p W_{p'} - \mathbb{E}_p U_{p'} = \frac{\beta}{1-\beta}\mathbb{E}_p J_{p'}$ and $f(\theta_p)/q(\theta_p) = \theta_p$

Reference I

Kirkby, R. (2023), 'Value function iteration (vfi) toolkit for matlab', `https://github.com/vfitoolkit/VFIToolkit-matlab`. Github.

Shimer, R. (2005), 'The cyclical behavior of equilibrium unemployment and vacancies', *American Economic Review* **95**(1), 25–49.
URL: *https://www.aeaweb.org/articles?id=10.1257/0002828053828572*