

# Aiyagari and equilibrium effects and K-S Algorithm

Solve a version of the Aiyagari model and Krusell-Smith algorithm

Jin Zhang<sup>1</sup>

<sup>1</sup>Peking University HSBC Business School

September 2, 2021

## Basic Model

Aiyagari Model

Main File

Krusell and Smith Model

Setting of Law of Motion

# Model Setting

- In the model, there is no aggregate risk and no aggregate dynamics.
- Capital is fixed at some (endogenous) level  $K$  and labour is normalized to 1.
- A representative firm that pays competitive prices for capital and labour, with rental rate and the wage rate are given by

$$\begin{aligned} r &= \alpha K^{\alpha-1} \\ w &= (1 - \alpha) K^{\alpha} \end{aligned} \tag{1}$$

With no aggregate dynamics,  $w$  and  $r$  are constant.

- Each individual is subject to a labour productivity shock  $z_{i,t}$ , and markets are incomplete, so that the only way to insure against a negative shock is by investing and disinvesting in capital.

## Model Setting(cont.)

- An individual takes the wage and the capital rental rate as given and solves:

$$\max_{\{c_{i,t}, k_{i,t}\}} E \sum_{t=1}^{\infty} \beta^t \left( \frac{c_{i,t}^{1-\nu}}{1-\nu} - \frac{\zeta_1}{\zeta_0} \exp(-\zeta_0 k_{i,t}) - \zeta_2 k_{i,t} \right), \text{ s.t.} \quad (2)$$

$$c_{i,t} + k_{i,t} = rk_{i,t-1} + wz_{i,t} + (1 - \delta)k_{i,t-1}$$

$$z_{i,t} = (1 - \rho) + \rho z_{i,t-1} + e_{i,t}$$

- The penalty function in the utility implements ensures that the problem is well behaved and prevents Ponzi schemes.
- We will focus on low values of  $\zeta_0$ .
  - First, the higher the value of  $\zeta_0$  the more nonlinear the problem and the harder it would be to obtain a solution with perturbation techniques.
  - Second, the inequality constraint is quite extreme and a gradual formulation may very well be more realistic.

## F.O.C and Steady State

- The first-order condition for the individual's problem is:

$$c_{i,t}^{-\nu} = -\zeta_2 + \zeta_1 \exp(-\zeta_0 k_{i,t}) + \beta c_{i,t+1}^{-\nu} (r + 1 - \delta) \quad (3)$$

- The penalty parameter is set such that

$$\zeta_2 = \zeta_1 \exp(-\zeta_0 k_{ss}) \quad (4)$$

otherwise there won't be steady state.

- We can get steady state interest rate

$$r = \frac{1}{\beta} - (1 - \delta) \quad (5)$$

- We can solve the steady state capital

$$\bar{K} = \left[ \frac{\beta \alpha}{1 - \beta(1 - \delta)} \right]^{\frac{1}{1-\alpha}} \quad (6)$$

Basic Model

Aiyagari Model

Main File

Krusell and Smith Model

Setting of Law of Motion

## Programs: mod| file

- In the Aiyagari.mod file, we fill in the parameters and model part

```

1  load parametervalues;      % Already saved in diseq.m
2  set_param_value('betta', par.betta);
3  set_param_value('alpha', par.alpha);
4  set_param_value('Δ', par.Δ);
5  set_param_value('zeta0', par.zeta0);
6  set_param_value('zeta1', par.zeta1);
7  set_param_value('zeta2', par.zeta2);
8  set_param_value('rho', par.rho);
9  set_param_value('sigshock', par.sigshock);
10 set_param_value('nu', par.nu);
11 set_param_value('k_init', par.kini);
12 set_param_value('r', r); % Taken as given for individual

```

```

1  % Euler Equation
2  c^(-nu) = -zeta2 + zeta1*exp(-zeta0*k) + betta*c^(+1)^(-nu)*(r+1-Δ);
3  z      = (1-rho)+rho*z(-1)+e;      % Shock Process
4  c + k  = r*k(-1)+w*z+(1-Δ)*k(-1); % Resource Constraint

```

## Programs: diseq.m | file

- In diseq.m file, it computes the difference between the capital supply and capital demand given the interest rate  $r$ .

```

1 function [diff,ks] = diseq(r,par,z,shocks)% r enter as a param
2     k = zeros(par.T,1);
3     save parametervalues r par    %Saves parameter values ...
        to a file that will be read by Dynare
4     dynare Aiyagari_full noclearall % Runs Dynare
5     % Change the file name after filling mod file!
6     load dynarerocks;
7     % Loads the decision rules to a matrix called 'decision'.
8     % Requires displace disp_dr.m file in the Dynare path.
9     ...

```



## Programs: diseq.m| file (cont.)

- To realize the idiosyncratic shock for a continuum of household, we can turn it into a *time-series* process for multiple periods, whose time-dimension aggregation is equivalent to the aggregation of the household!!!

```

1  ...
2  z = ones(par.T,1);
3  for t=2:par.T
4      z(t) = 1-par.rho + par.rho*z(t-1)+shocks(t);
5  end
6  k(1) = para.kini;
7  for t = 2:par.T % From the decision matrix computed by Dynare!
8      k(t)=decision(1)+decision(3)*k(t-1)+decision(4)*z(t)
9          +decision(5)*shocks(t)+decision(6)*k(t-1)^2
10         +decision(7)*k(t-1)*z(t)+decision(8)*z(t)^2
11         +decision(9)*shocks(t)^2+decision(10)*shocks(t)*k(t-1)
12         +decision(11)*shocks(t)*z(t-1);
13 end
14 ks = mean(k(par.T0:end)); % average capital supply from HH
15 kd = (r/par.alpha)^(1/(par.alpha-1)); % implied capital dmnd
16 diff = ks-kd;

```

## Krusell and Smith Model

### Setting of Law of Motion

# Main Programs: SolveAiyagari.m | file

- The whole solving process is as follows:

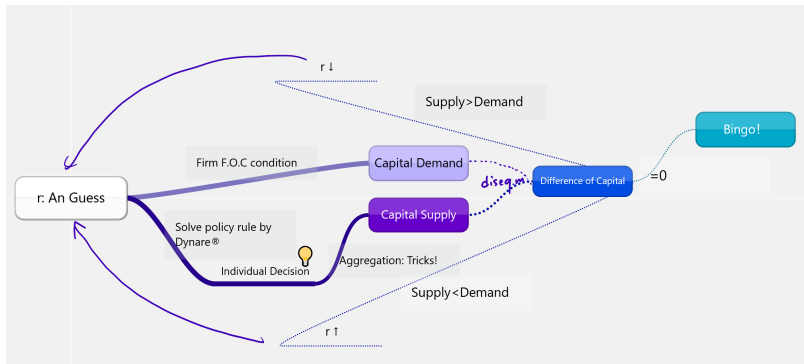


Figure: The whole solving process

- Now we have to fill the determine sentences of interest rate  $r$ .

## Main Programs: SolveAiyagari.m | file

- First an increase in volatility is considered in the equilibrium model.
- Second, using the equilibrium interest rate of the low-volatility economy, we then resolve the two economies to measure the partial equilibrium response of the increase in volatility.

```

1  %-----
2  % 1. Parameters
3  %-----
4  par.T = 100000;% length of simulation
5  par.T0 = 10001;% start of sample
6  ...
7  % allocating memory and shock values
8  z      = ones(par.T,1); % idiosyncratic shocks
9  k      = zeros(par.T,1); % Intial capital for individual
10 ks_partial = zeros(2,1);
11 ks_ge      = zeros(2,1);
12 r_ge       = zeros(2,1);
13 sigs = [0.001;0.3]; % values for standard deviations of shocks

```

## Loop over sigma for equilibrium $r$

- The algorithm for root finding is

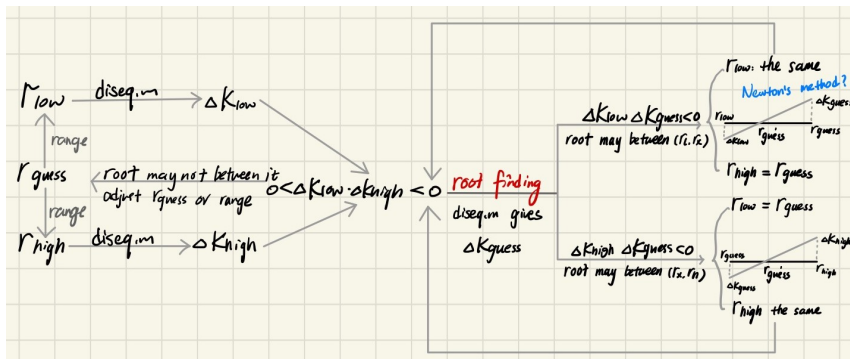


Figure: Equilibrium Root Finding

## Loop over sigma for equilibrium $r$

```
1  for i = 1:2 % defining shocks
2  par.sigshock = sigs(i); % choosing standard deviation
3  shocks      = par.sigshock*innovations;
4      for t = 2:par.T
5          z(t) = 1-par.rho + par.rho*z(t-1) + shocks(t);
6      end
7  % Solve for equilibrium r
8  r_l      = r-0.0001; % lower bound for r at which ks < kd
9  r_u      = r+0.0001; % upper bound for r at which ks > kd
10 err      = 100; % initial error
11 r_x      = r_u;
12 [diseq_r_l,ks_l] = diseq(r_l,par,z,shocks);
13 [diseq_r_u,ks_u] = diseq(r_u,par,z,shocks);
14 [diseq_r_x,ks_x] = diseq(r_x,par,z,shocks);
15 prodt     = diseq_r_l*diseq_r_u;
16
17 if prodt > 0 % A warning for a narrow initial interval
18     disp(' root not in between r_l and r_u')
19     disp(' adjust r_l or r_u')
20     stop
21 end
```

## Loop over sigma for equilibrium $r$

```
1      disp(' root in between r_l and r_u')
2      disp([r_l r_x r_u diseq_r_l diseq_r_x diseq_r_u])%pause
3
4  while err > par.tol
5      r_x_old = r_x;
6      r_x      = r_u - ...
          diseq_r_u*(r_u-r_l)/(diseq_r_u-diseq_r_l) % Newton
7      [diseq_r_x,ks_x] = diseq(r_x,par,z,shocks);
8
9      disp(diseq_r_x)
10     disp([r_l r_x r_u diseq_r_l diseq_r_x diseq_r_u])      ...
        %pause
11     if r_x ≠ 0
12         err = abs((r_x - r_x_old)/r_x) * 100;
13     else
14         err = abs(r_x - r_x_old);
15     end
16     test = diseq_r_l*diseq_r_x;
```

## Loop over sigma for equilibrium $r$

```
1      if test == 0
2          err = 0; % root is at r_x
3      elseif test < 0
4          r_u = r_x; %root is below r_x
5          diseq_r_u = diseq_r_x;
6      else
7          r_l = r_x; %root is above r_x
8          diseq_r_l = diseq_r_x;
9      end
10 end
11
12 ks_ge(i) = ks_x; % general equilibrium capital
13 r_ge(i) = r_x; % general equilibrium interest rate
14 end
15 % Save values for proj_PE.m This just saves the Dynare's ...
    results. They will be used later as starting values for ...
    projection.
16 save r_ge
17 save ks_ge
18 save par
```



# Partial Equilibrium Result

- Fill in the empty part by your own:)
- The effects of volatility are larger in partial than in general equilibrium.
- The reason that causes the dampening is
  - In the general equilibrium, people save more for precautionary  $\rightarrow r \downarrow$  due to equilibrium effect  $\rightarrow K \downarrow$  due to the price rise.
  - However, in the partial equilibrium, the precautionary saving doesn't affect interest rate, which obliterates the second effect.

## Basic Model

Aiyagari Model  
Main File

## Krusell and Smith Model

### Setting of Law of Motion

## Krusell and Smith(1998): idiosyncratic+agg. shock

- Krusell and Smith(1998) is the same as Aiyagari assignment except that there is not only idiosyncratic, *but also* aggregate productivity shocks.
- The presence of aggregate shocks makes the problem much harder because now the cross-sectional distribution is time-varying which means that we have a (time-varying) state variable that is infinite-dimensional.
- The algorithm takes three steps
  1. Start with the (assumed!) following law of motion for the capital stock:

$$K_t^a = b_0 + b_K K_{t-1}^a + b_Z (z_t - z) \quad (7)$$

Compared with Aiyagari, we have an extra state variable,  $K_{t-1}^\alpha$ .

2. Given the policy rules of the individuals obtained in (1), conduct a Monte Carlo simulation for a panel of  $I$  agents. Each period aggregate the individual capital stocks to obtain a time series for aggregate capital  $K_t^\alpha$ .
3. Run a regression to update the coefficients of the law of motion of  $K_t^a$  given in (7). Go back to (1) until the law of motion obtained in (3) is (basically) the same as (1).



# The Dynare file

- Fill in the parameter loading part and the equations

```

1  load parametervalues; % Delete the b_0 b_k b_z of it in m ...
    file saving part since it interrupts!
2  load aggregatelaw;    % Updated coefficients store here
3  ...
4  % Euler Equation
5  1/c = -zeta2+zeta1*exp(-zeta0*k)+beta*1/c(+1)*(r(+1)+1-Δ);
6  % Shock Process
7  z    = (1-rho)+rho*z(-1)+e2;
8  % Resource Constraint
9  c+k   = r*k(-1)+w*(1+e1)+(1-Δ)*k(-1);
10 % F.O.C 1
11 r     = alpha*z*Ka(-1)^(alpha-1);
12 % F.O.C 2
13 w     = (1-alpha)*z*Ka(-1)^(alpha);
14 % Law of motion, updated
15 Ka    = b_0+b_K*Ka(-1)+b_z*(z-1);

```

- Now we finish the mod file.

# The main file

- A quick review of the file

```

1  % settings
2  T=10000; % number of time periods in simulation step
3  ...
4  % parameter values
5  ...
6  % initial values for LoM coefficients for aggregate capital
7  ...
8  save parametervalue ...
9  % reserve memory
10 ...
11 % draw shocks and simulate productivity
12 %% Fill in the construction of shocks
13 % Loop to find the coefficients
14 ...
15 %% Fill in the step 2 and 3

```

- Now we construct the shocks(exactly the same, copy it) and fill in the loop.

## Make use of Policy Rule

```

1  while error > crit % step 1
2      save aggregatelaw b_0 b_K b_z sig_e1
3      dynare model_agg_uncertainty.mod noclearall % policy rule!
4      load dynarerocks % uploads the decision
5      for t=2:T          % step 2
6          % compute individual k using decision rules by dynare
7          Ka_ss = b_0/(1-b_K); % Update the steady state of Ka
8          k_ss = decision(1,1)-decision(2,1);
9          k_sim(:,t)=k_ss + [-ones(N,1)%Correction term take minus sign
10             (k_sim(:,t-1)-k_ss)      ones(N,1)*(z_sim(t-1)-1)
11             ones(N,1)*(Ka_sim(t-1)-Ka_ss)      e1_sim(:,t)
12             ones(N,1)*e2_sim(t)      (k_sim(:,t-1)-k_ss).^2 ...
13             (z_sim(t-1)-1)*(k_sim(:,t-1)-k_ss) ...
14             ones(N,1)*(z_sim(t-1)-1).^2 ...
15             ...
16             ones(N,1)*(Ka_sim(t-1)-Ka_ss)*e2_sim(t)]*decision(2:end,1);
17
18          Ka_sim(t)= mean(k_sim(:,t)); % compute aggregate capital
19      end

```

- The policy rule gives Taylor expansion at the **state state** of each state variable. Update according to the **deviation** from ss of each term, not ss!

- Now that we finish the simulation given the to-be-verified  $b_0, b_K, b_z$ . Redo the regression under simulation until the coefficient converge.

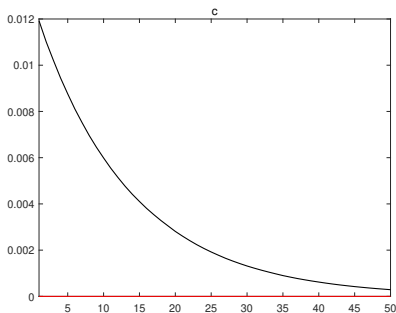
```

1  % step 3: run a regression and update law of motion for Ka
2      [b, \r]=regress(Ka_sim(2+Tdiscard:T) ,
3      [ones(T-1-Tdiscard,1)
4      Ka_sim(1+Tdiscard:T-1)
5      z_sim(2+Tdiscard:T)-1]); % Discard periods with ...
        diverging k
6      error = max(abs([b_0-b(1) b_K-b(2) b_z-b(3)])) % Update ...
        the distance of coefficient
7  % Update Coefficient
8      b_0=b(1)
9      b_K=b(2)
10     b_z=b(3)

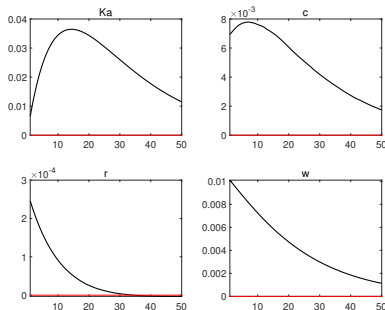
```



# Impulse Response Function



**Figure:** Impulse response to idiosyncratic shock



**Figure:** Impulse response to aggregate shock

- Positive idiosyncratic shock:  $c \uparrow$ , wage, agg. capital, return rate  $-$ .
- Positive agg. shock:  $z \uparrow \rightarrow r, w \uparrow \rightarrow c$ ;  $Y \uparrow \rightarrow k \uparrow$  through B.C.  $\rightarrow K_a \uparrow$ .

Basic Model

Aiyagari Model

Main File

Krusell and Smith Model

Setting of Law of Motion

## Linear form of LoM

- The linear form setting of the law of motion of aggregate shock induces a question mark about its accuracy.
- Before answering the inaccuracy due to the setting, one question is whether we can alter the LoM to incorporate the second moments change of aggregate capital.
  - e.g.  $K_t^a = b_0 + b_{K1}E(K_{t-1}^a) + b_{K2}std(K_{t-1}^a) + b_Z(z_t - z)$
- Unfortunately, the answer is no.

## Linear form of LoM(cont.)

- With the setting of linear form, Dynare gives policy function with form

$$k' = F(k, z, e_1, K^a - E(K^a)) \quad (8)$$

- With aggregation, we have

$$K^{a'} = \Sigma k' = \Sigma F(k, z, e_1, K^a - E(K^a)) \quad (9)$$

- Including second order, the policy function becomes

$$k' = F(k, z, e_1, K^a - E(K^a), [K^a - E(K^a)]^2) \quad (10)$$

- Consider  $[k' - E(k')]^2$ , it becomes

$$\{F(k, z, e_1, K^a - E(K^a), [K^a - E(K^a)]^2) - E[F(k, z, e_1, k - E(k), [K^a - E(K^a)]^2)]\}^2 \quad (11)$$

which induce  $[k' - E(k')]^4$ .

- During further computation, infinite moments will be generated, which can't be enumerated.

# Winberry (2018)

- The key assumption involves two elements
  - Policy function
  - Law of motion
- Winberry(2018) induce a way —— Histogram.