Github

https://github.com/mariiaelk/ECON-8210-Computational-Methods-in-Economics-HW-2

Model

Let's obtain the system equations which describe dynamics of the model. The problem of the household is:

$$\begin{aligned} &\max_{c_t, l_t, k_{t+1}} \mathbb{E}_0 \left\{ \beta^t \left(\log c_t - \frac{l_t^2}{2} \right) \right\} \\ \text{s.t.} \quad & c_t + k_{t+1} - (1-\delta) k_t \leqslant w_t l_t + r_t k_t \\ & k_0 \text{ given} \end{aligned}$$

FOCs (necessary and sufficient):

$$\begin{aligned} [c_t] : & \frac{1}{c_t} = \lambda_t \\ [l_t] : & l_t = \frac{1}{c_t} w_t \\ [k_{t+1}] : & 1 = \beta \mathbb{E}_t \left\{ \frac{c_t}{c_{t+1}} \left[r_{t+1} + (1-\delta) \right] \right\} \end{aligned}$$

Problem of a perfectly competitive firm gives us the following conditions for wage and rental rate of capital:

$$\begin{aligned} w_t &= (1-\alpha) e^{z_t} k_t^{\alpha} l_t^{-\alpha} \\ r_t &= \alpha e^{z_t} k_t^{\alpha-1} l_t^{1-\alpha} \end{aligned}$$

Overall, we have the following system for period t:

$$\begin{aligned} & l_t = \frac{1}{c_t} w_t \\ & 1 = \beta \mathbb{E}_t \left\{ \frac{c_t}{c_{t+1}} \left[r_{t+1} + (1-\delta) \right] \right\} \\ & w_t = (1-\alpha) e^{z_t} k_t^{\alpha} l_t^{-\alpha} \\ & r_t = \alpha e^{z_t} k_t^{\alpha-1} l_t^{1-\alpha} \\ & c_t + k_{t+1} - (1-\delta) k_t = e^{z_t} k_t^{\alpha} l_t^{1-\alpha} \\ & z_t = \rho z_{t-1} + \sigma \varepsilon_t \end{aligned}$$

6 equations and 6 endogenous variables: $l_t, c_t, k_t, w_t, r_t, z_t$. We can substitute out wage and

rental rate of capital:

$$
\begin{aligned}
c_t &= (1-\alpha)\, e^{z_t} k_t^{\alpha} \left(l_t\right)^{-(1+\alpha)} \\
1 &= \beta \mathbb{E}_t \left\{ \frac{c_t}{c_{t+1}} \left[\alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} l_{t+1}^{1-\alpha} + (1-\delta) \right] \right\} \\
c_t &+ k_{t+1} - (1-\delta)\, k_t = e^{z_t} k_t^{\alpha} l_t^{1-\alpha} \\
z_t &= \rho z_{t-1} + \sigma \varepsilon_t
\end{aligned}
$$

Essentially, we are looking for decision rules: $k_{t+1} = k'\left(k_t, z_t\right), c_t = c\left(k_t, z_t\right), l_t = l\left(k_t, z_t\right)$. In deterministic steady state we get:

$$
\begin{aligned}
c = (1-\alpha)\, e^{z_{ss}} k^{\alpha} \left(l\right)^{-(1+\alpha)} &\Rightarrow l = \left[(1-\alpha)\, e^{z_{ss}} \left(\frac{k}{l}\right)^{\alpha} \left(\frac{c}{l}\right)^{-1} \right]^{\frac{1}{2}} \\
1 = \beta \left[\alpha e^{z_{ss}} \left(\frac{k}{l}\right)^{\alpha-1} + (1-\delta) \right] &\Rightarrow \frac{k}{l} = \left(\frac{1}{\alpha e^{z_{ss}}} \left[\frac{1}{\beta} - (1-\delta) \right] \right)^{\frac{1}{\alpha-1}} \\
c + \delta k = e^{z_{ss}} k^{\alpha} l^{1-\alpha} &\Rightarrow \frac{c}{l} = e^{z_{ss}} \left(\frac{k}{l}\right)^{\alpha} - \delta \frac{k}{l} \qquad (1) \\
z = z_{ss} &= 0
\end{aligned}
$$

1 Chebychev

Code which produces results for this section: Q1.m.

After we obtain finite approximation of z_t our system becomes

$$
\begin{aligned}
c_t &= (1-\alpha)\, e^{z_t} k_t^{\alpha} \left(l_t\right)^{-(1+\alpha)} \\
\frac{1}{c_t} &= \beta \sum_{z_{t+1} \in Z} \Pr\left(z_{t+1} | z_t\right) \left\{ \frac{1}{c_{t+1}} \left[\alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} l_{t+1}^{1-\alpha} + (1-\delta) \right] \right\} \qquad (2) \\
c_t &+ k_{t+1} - (1-\delta)\, k_t = e^{z_t} k_t^{\alpha} l_t^{1-\alpha}
\end{aligned}
$$

We are going to approximate decision rule for labor as follows:

$$
l\left(k_t, z_j; \theta\right) = \sum_{i=1}^{n} \theta_i\left(z_j\right) \psi_i\left(k_t\right)
$$

where ψ_i is Chebychev polynomial of order i. We choose $n = 6$, so we use Chebychev polynomials of order up to 6. Once we have our labor policy function, we can compute consumption as a function of k_t and z_t from the first equation of the system (2), $c\left(k_t, z_j; \theta\right)$, and next period capital $k'\left(k_t, z_j; \theta\right)$ from last equation of the same system.

The goal is to solve for the vectors of $\theta\left(z_j\right) \forall z_j \in Z$ such that errors of the Euler equation are zeros at n collocation points given $l\left(k_t, z_j; \theta\right)$, $c\left(k_t, z_j; \theta\right)$, $k'\left(k_t, z_j; \theta\right)$. So, we have a system of

$3 \times 6 = 18$ equations in 18 coefficients.

Overall, the algorithm works as follows:

1. Solve for deterministic steady state of the model using the system (1). The grid for capital is centered around steady state level of capital with minimum value defined as $k_{min} = 0.75k_{ss}$ and maximum value defines as $k_{max} = 1.25k_{ss}$.

2. Find collocation points (zeros of Chebychev polynomials) using the following formula:

$$x_i = -\cos\frac{2i-1}{2n}\pi$$

Then Chebychev polynomials are defined recursively over the interval of $[0, 1]$:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_{i+1}(x) &= 2xT_i(x) - T_{i-1}(x) \end{aligned}$$

Collocation points for capital are found as follows:

$$k_i = \frac{(x_i + 1)(k_{max} - k_{min})}{2k_{min}}$$

This relationship allows us to map x into k and vice versa.

3. Initial guess for vector θ is based on steady state level of labor. $\theta_1(z_j) = l_{ss}, \theta_i(z_j) = 0 \forall i, j$.

4. Using this guess, we use fsolve function to solve the following system of equations:

$$\frac{1}{c(k_i, z_j; \theta)} = \beta \sum_{z' \in Z} \Pr(z'|z_j) \left\{ \frac{1}{c(k'((k_i, z_j; \theta)), z'; \theta)} \left[\alpha e^{z'} k'(k_i, z_j; \theta)^{\alpha-1} l(k'(k_i, z_j; \theta), z'; \theta)^{1-\alpha} + (1-\delta) \right] \right\} \forall i, j$$

More specifically, given some guess of θ the error of the Euler equation is computed as follows:

   (a) For each z_j and each k_i compute associated value of current labor l_t using θ, Chebychev polynomials and mapping between k and x.

   (b) Compute associated c_t and k_{t+1} using first and third equations of the system (2).

   (c) Convert k_{t+1} to x_{t+1} in order to compute l_{t+1} for each possible realization of next period technology shock. Then compute corresponding c_{t+1} for each possible realization of next period technology shock. Use transition probabilities to compute right hand side of the Euler equation.

(d) Compute the difference between left and right hand sides of the Euler equations.

(e) fsolve is used to equate errors to 0 for each pair of z_j, k_i.

5. After we obtain the vector of θ which solves our system of equations, we get our policy function for labor. We can also evaluate this policy function on a finer grid for capital with 100 points using the mapping between k and x. Policy function for consumption and next period capital are obtained using first and third equations of the system (2).

Running the algorithm takes about 0.3 seconds. Figure (1) compares policy functions obtained using approximation with Chebychev polynomials with policy functions obtained using generalized endogenous grid method (see section 6 for details on the implementation of generalized EGM).

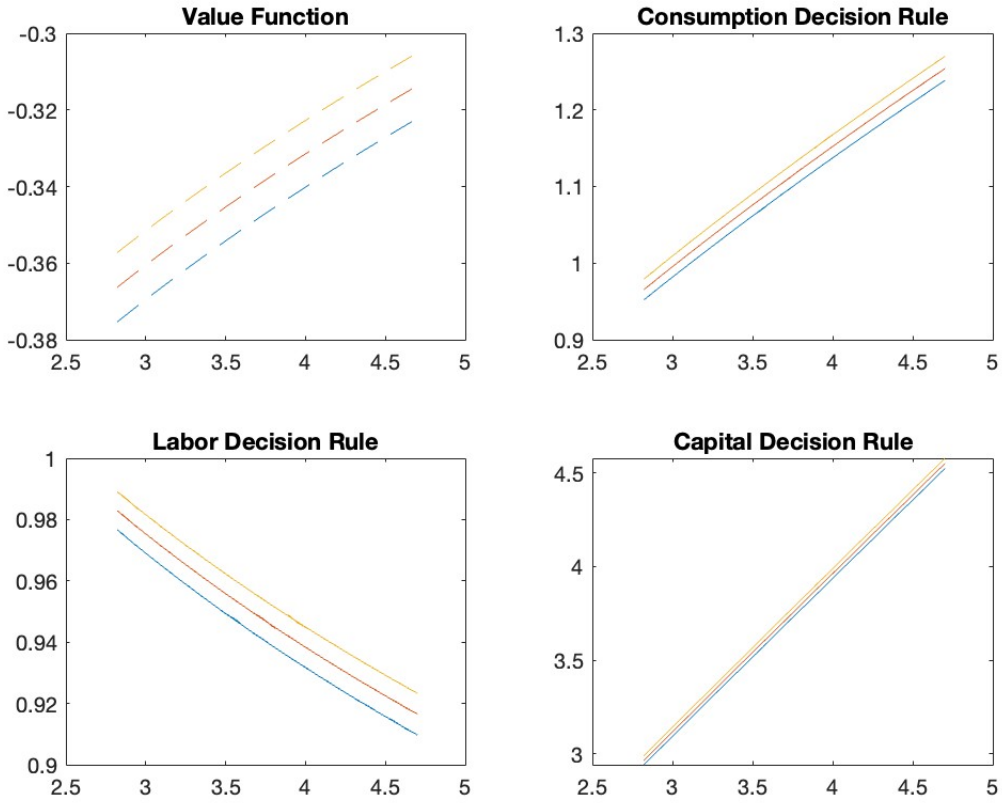


Figure 1: Comparison of policy functions obtained using Chebychev polynomials (solid lines) with policy functions from generalized endogenous grid method (dashed lines; yellow lines correspond to largest realization of current technology shock, red – to medium realization and blue – to smallest realization).

One can see that policy functions are almost identical. We can also use policy functions to simulate the model. Figure (2) shows distribution of main variables of the model obtained from simulating the model for 10000 periods (with first 1000 dropped).

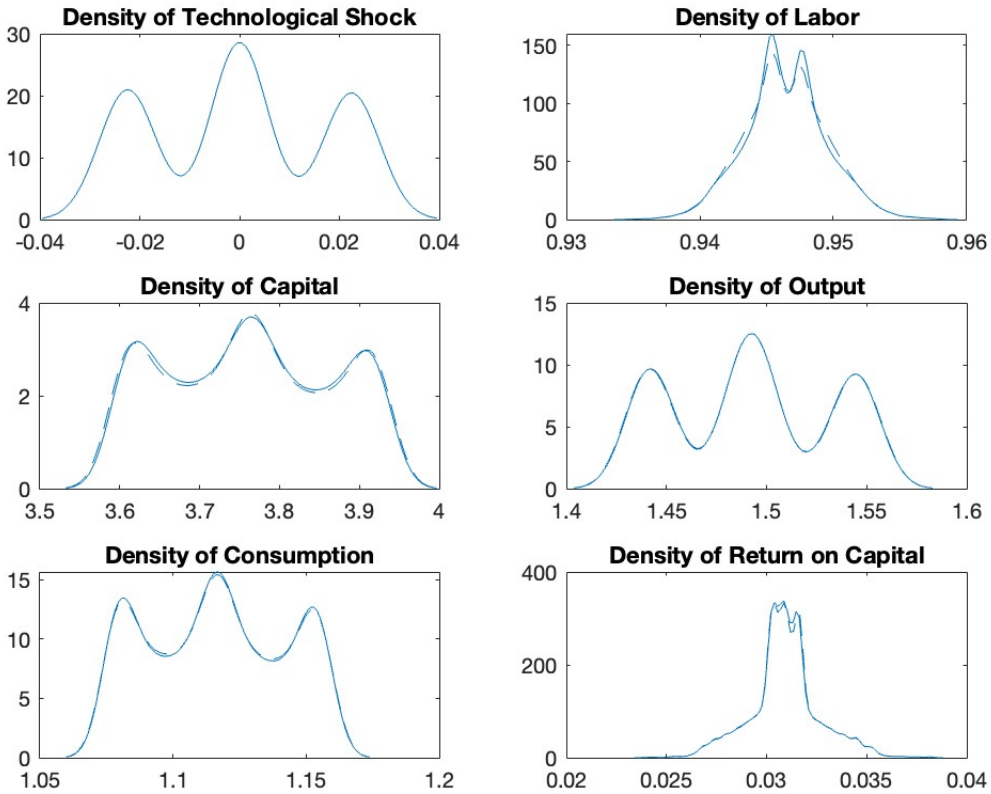


Figure 2: Comparison of distributions of model's variables obtained from simulating the model using policy functions obtained using Chebychev polynomials (solid lines) and policy functions from generalized endogenous grid method

In this case differences between two solutions are more noticeable, but overall distributions are almost identical.

2 Finite Elements

Code which produces results for this section: Q2.m.

Again, we are looking for labor policy function

$$l\left(k_t, z_j; \theta\right) = \sum_{i=1}^{n} \theta_i\left(z_j\right) \psi_i\left(k_t\right)$$

where $\psi_i\left(k_t\right)$ are finite elements. In other words, we are looking for vector $\theta = \left(\theta\left(z_1\right)', \theta\left(z_2\right)', \theta\left(z_3\right)'\right)'$. Consumption and next period capital would be defined as

$$\begin{aligned} c\left(k, z_j; \theta\right) &= (1-\alpha)\, e^{z_j} k^{\alpha} \left(l\left(k, z_j; \theta\right)\right)^{-(1+\alpha)} \\ y\left(k, z_j; \theta\right) &= e^{z_j} k^{\alpha} \left(l\left(k, z_j; \theta\right)\right)^{1-\alpha} \\ k'\left(k, z_j; \theta\right) &= y\left(k, z_j; \theta\right) + (1-\delta)k - c\left(k, z_j; \theta\right) \end{aligned} \tag{3}$$

First, we need to bind and partition the state space for capital. We will use the same k_{min} and k_{max} we used for Chebychev polynomials. Then $\Omega = [k_{min}, k_{max}]$. Then divide Ω into $n-1=7$ equally spaced intervals $[k_i, k_{i+1}]$ with $k_1 = k_{min}$ and $k_8 = k_{max}$. Then finite elements on capital are defined as follows:

$$\psi_i(k) = \begin{cases} \frac{k-k_{i-1}}{k_i-k_{i-1}} & \text{if } k \in [k_{i-1}, k_i] \\ \frac{k_{i+1}-k}{k_{i+1}-k_i} & \text{if } k \in [k_i, k_{i+1}] \\ 0 & \text{elsewhere} \end{cases}$$

Residual equation is obtained from Euler equation:

$$R(k, z_j; \theta) = \beta \sum_{z' \in Z} \Pr(z'|z_j) \left\{ \frac{1}{c\left(k'\left(k, z_j; \theta\right), z'; \theta\right)} \left[\alpha e^{z'} \left(k'\left(k, z_j; \theta\right)\right)^{\alpha-1} \cdot \right.\right.$$
$$\left.\left. \cdot \left(l\left(k'\left(k, z_j; \theta\right), z'; \theta\right)\right)^{1-\alpha} + (1-\delta) \right] - \frac{1}{c\left(k, z_j; \theta\right)} \right\}$$

Galerkin scheme implies that we weight residuals as follows:

$$\int_{k \in [k_{min}, k_{max}]} \psi_i(k) R\left(k, z_j; \theta\right) dk = 0 \, \forall i, j$$

Given our approach to defining $\psi_i(k)$,

$$\int_{k \in [k_{i-1}, k_{i+1}]} \psi_i(k) R\left(k, z_j; \theta\right) dk = 0 \, \forall i, j$$

For $i = 1$ we will have

$$\int_{k \in [k_1, k_2]} \psi_1(k) R\left(k, z_j; \theta\right) dk = 0 \, \forall j$$

Similarly, for $i = 8$ we will have

$$\int_{k \in [k_7, k_8]} \psi_8(k) R\left(k, z_j; \theta\right) dk = 0 \, \forall j$$

We will use midpoint method to evaluate integrals:

$$\int_{k \in [k_{i-1}, k_{i+1}]} \psi_i(k) R\left(k, z_j; \theta\right) dk = \sum_{m=1}^{M} (b_m - a_m) \psi_i\left(\frac{a_m + b_m}{2}\right) R\left(\frac{a_m + b_m}{2}, z_j; \theta\right)$$

for $M = 1000$.

We have a system of $8 \times 3 = 24$ equations. Solving this system numerically, we obtain the vector θ with 24 elements. We use fsolve to solve this system.

Overall, the algorithm works as follows:

1. Solve for deterministic steady state of the model using the system (1). The grid for capital is

centered around steady state level of capital with minimum value defined as $k_{min} = 0.75k_{ss}$ and maximum value defines as $k_{max} = 1.25k_{ss}$.

2. Initial guess for vector θ is based on steady state level of labor. $\theta_i(z_j) = l_{ss} \forall i, j$. So, essentially we guess that policy function for labor implies choosing steady state level of labor for any capital and technology shock.

3. Using this guess, we use fsolve function to solve the following system of equations:

$$\int_{k \in [k_{i-1}, k_{i+1}]} \psi_i(k) R\left(k, z_j; \theta\right) dk = 0 \, \forall i, j$$

   More specifically, given some guess of θ the left hand side of the equation above for some i, j is computed as follows:

   (a) For a given k the error of the Euler equation is computed $R\left(k, z_j; \theta\right)$ as follows:

      i. Evaluate 8 finite elements at this value of k. Then compute current labor l using these finite elements and the guess for θ.

      ii. Compute associated c and k' using (3).

      iii. Evaluate 8 finite elements at this value of k'. Then compute future labor l' using these finite elements and the guess for θ for each possible realization of next period technology shock. Then compute corresponding c' for each possible realization of next period technology shock. Use transition probabilities to compute right hand side of the Euler equation.

      iv. Compute the difference between left and right hand sides of the Euler equations.

   (b) Compute the integral for given i, j using midpoint method explained above.

4. After we obtain the vector of θ which solves our system of equations, we get our policy function for labor. We can also evaluate this policy function on a finer grid for capital with 100 points. Policy function for consumption and next period capital are obtained using (3).

Running the algorithm takes about 30 seconds. Figure (3) compares policy functions obtained using approximation with finite elements with policy functions obtained using generalized endogenous grid method (see section 6 for details on the implementation of generalized EGM).

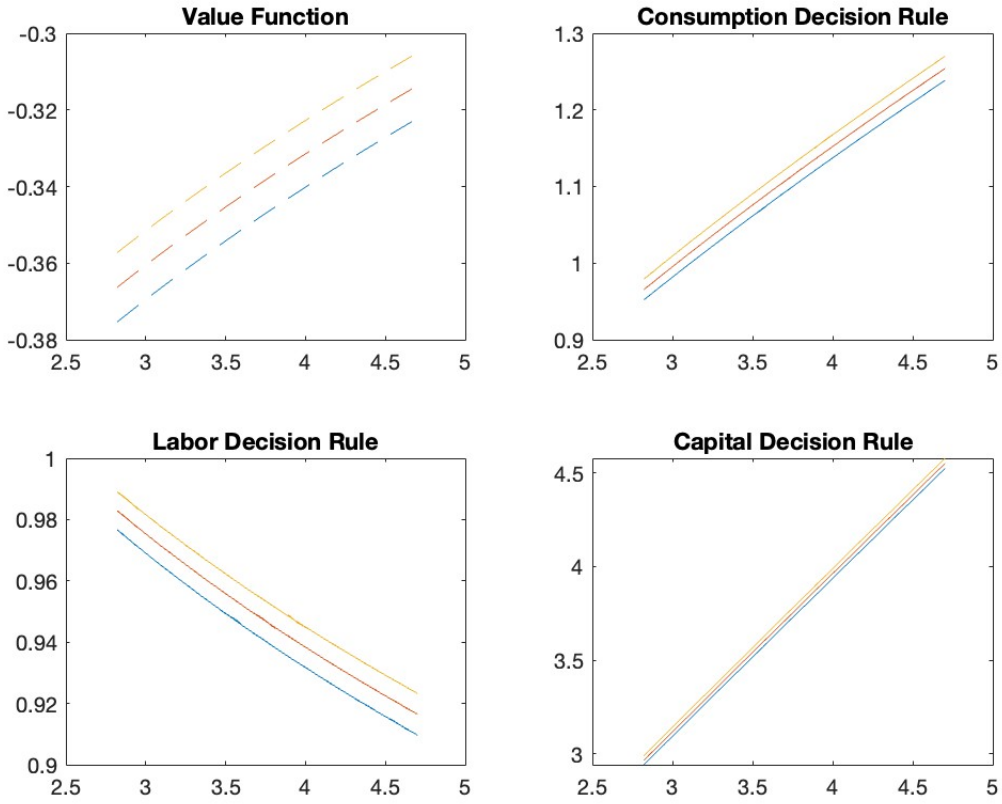


Figure 3: Comparison of policy functions obtained using finite elements (solid lines) with policy functions from generalized endogenous grid method (dashed lines; yellow lines correspond to largest realization of current technology shock, red – to medium realization and blue – to smallest realization).

One can see that policy functions are almost identical. We can also use policy functions to simulate the model. Figure (4) shows distribution of main variables of the model obtained from simulating the model for 10000 periods (with first 1000 dropped).

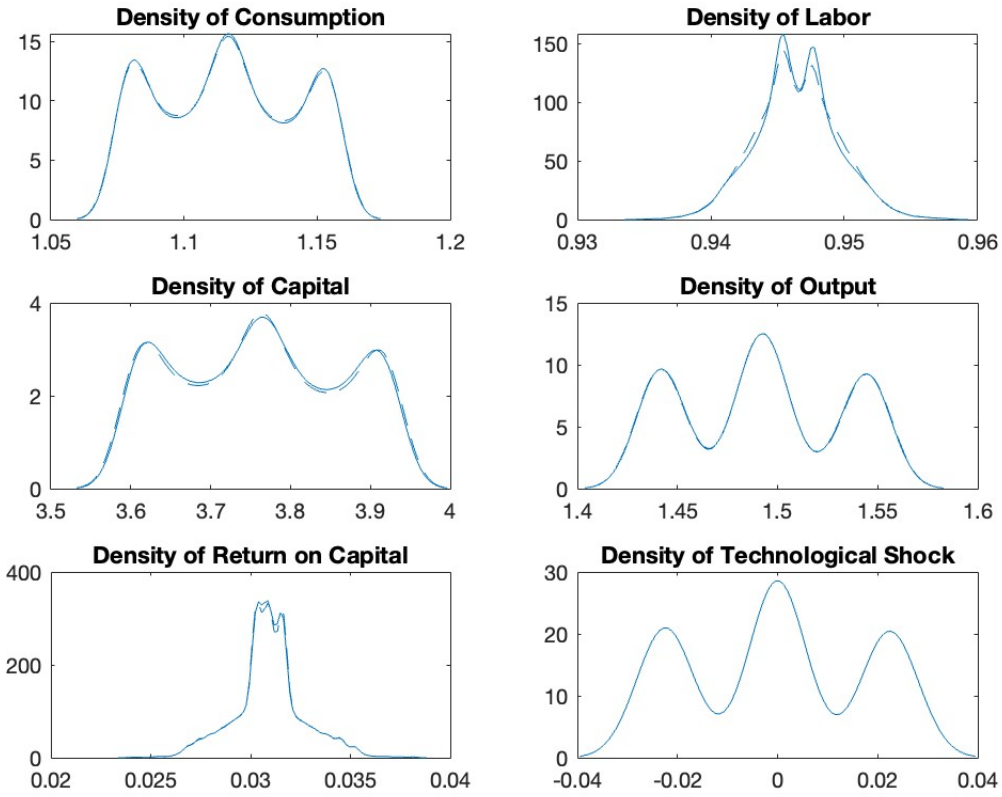


Figure 4: Comparison of distributions of model's variables obtained from simulating the model using policy functions obtained using finite elements (solid lines) and policy functions from generalized endogenous grid method

In this case differences between two solutions are more noticeable, but overall distributions are almost identical.

3 Perturbation

Codes which produces results for this section: Q3.m and Q3.mod.

We are going to use Dynare in order to obtain solution to our model, and then use its output to produce policy functions and distributions similar to the ones we computed using Chebychev polynomials and finite elements.

Our system of equations

$$
\begin{aligned}
c_t &= (1-\alpha)\, e^{z_t} k_t^{\alpha}\, (l_t)^{-(1+\alpha)} \\
\frac{1}{c_t} &= \beta \mathbb{E}_t \left\{ \frac{1}{c_{t+1}} \left[\alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} l_{t+1}^{1-\alpha} + (1-\delta) \right] \right\} \\
c_t &+ k_{t+1} - (1-\delta)\, k_t = e^{z_t} k_t^{\alpha} l_t^{1-\alpha} \\
y_t &= e^{z_t} k_t^{\alpha} l_t^{1-\alpha} \\
z_t &= \rho z_{t-1} + \sigma \varepsilon_t
\end{aligned}
$$

Using Dynare we obtain log-linearization of this model around its deterministic steady state. We rely on third order approximation when doing so. Dynare's code takes approximately 2 seconds to run.

To get policy functions comparable with policy functions obtained from other methods, we need to use the following formula:

$$
\begin{aligned}
x_t &= x^s + G_0 + G_1 \xi_t + G_2 \xi_t \otimes \xi_t + G_3 \xi_t \otimes \xi_t \otimes \xi_t \\
\xi_t &= \begin{pmatrix} \hat{k}_t \\ \hat{z}_{t-1} \\ \varepsilon_t \end{pmatrix}
\end{aligned}
$$

where G_1, G_2, G_3 are matrices with derivatives and $\hat{k}_t$ is the deviation of predetermined capital from its steady state value. To obtain 3 different levels of technology we will set $\hat{z}_{t-1} = 0$ and ε_t to one of 3 values we get from Tauchen's method of approximating z_t: $\{-0.02, 0, 0.02\}$ divided by σ.

Figure (5) compares policy functions obtained using third order perturbation with policy functions obtained using generalized endogenous grid method (see section 6 for details on the implementation of generalized EGM).

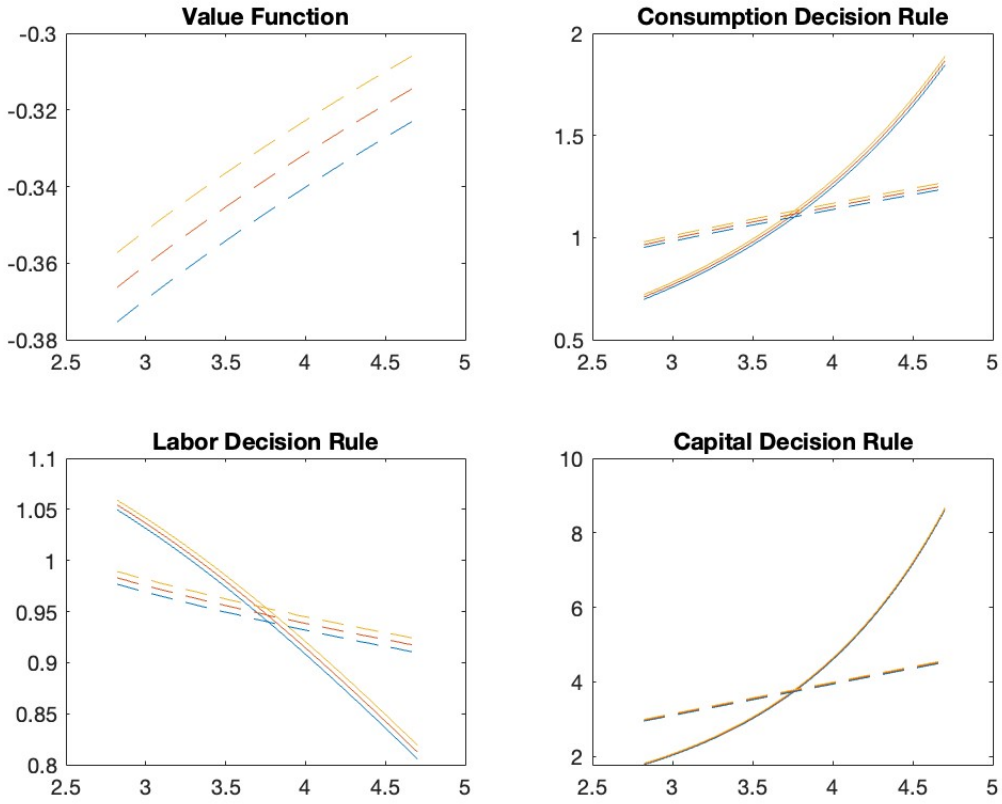


Figure 5: Comparison of policy functions obtained using third order perturbation (solid lines) with policy functions from generalized endogenous grid method (dashed lines; yellow lines correspond to largest realization of current technology shock, red – to medium realization and blue – to smallest realization).

Policy functions obtained using two different methods are close to each other around the steady state level of capital. Also the distance between the choice of labor given different realization of shocks is similar for two methods. However, as capital deviates from its steady state level we can see a growing difference between two methods. Since we used log-linearization, policy functions take exponential form when we use perturbation but are approximately linear when we use generalized EGM. Overall, this result is expected as perturbation method should work well only around the steady state (since we log-linearized the model around the steady state).

We can also use policy functions to simulate the model. Figure (6) shows distribution of main variables of the model obtained from simulating the model for 10000 periods (with first 1000 dropped).

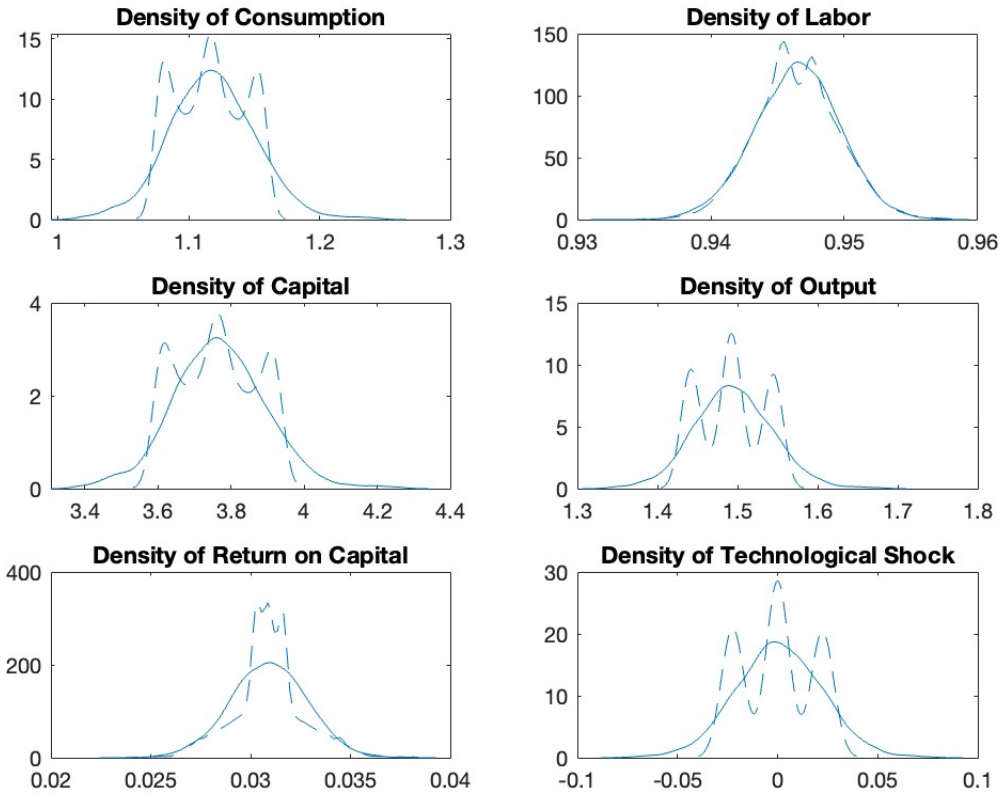


Figure 6: Comparison of distributions of model's variables obtained from simulating the model using policy functions obtained using third order perturbation (solid lines) and policy functions from generalized endogenous grid method

Distributions obtained from two different methods are similar. However, distributions obtained using perturbation are more smooth since we did not approximate technological process with a Markov chain with 3 realizations, but used the original AR(1) process.

4 Deep Learning

Code which produces results for this section: Q4.m.

As before, we are solving the following system of equations:

$$\begin{aligned} c_t &= (1-\alpha)\, e^{z_t} k_t^{\alpha} \left(l_t\right)^{-(1+\alpha)} \\ \frac{1}{c_t} &= \beta \sum_{z_{t+1} \in Z} \Pr\left(z_{t+1}|z_t\right) \left\{ \frac{1}{c_{t+1}} \left[\alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} l_{t+1}^{1-\alpha} + (1-\delta)\right] \right\} \\ c_t &+ k_{t+1} - (1-\delta)\, k_t = e^{z_t} k_t^{\alpha} l_t^{1-\alpha} \end{aligned} \tag{4}$$

We are looking for labor policy function:

$$l_t = l^z(k_t)$$

Notice that it is indexed by z. It implies that neural network will have 3 outputs: one for each realization of current technology shock. The only input is current value of capital. Given correct policy function for labor, consumption can be obtained from the first equation of the system. Then next period capital can be obtained from the last equation of the system.

Overall, the algorithm works as follows:

1. Solve for deterministic steady state of the model using the system (1). The grid for capital is centered around steady state level of capital with minimum value defined as $k_{min} = 0.75k_{ss}$ and maximum value defines as $k_{max} = 1.25k_{ss}$. We use capital grid with 100 points.

2. First, we are going to train the neural network to produce policy functions which imply choosing constant level of labor equal to the steady state level of labor for any current level of capital and technology. So, for each level of capital on the grid we want the prediction of neural network to be l_{ss}. We use mean of square root of squared error as the loss function. We use Adam optimizer when updating parameters of the neural network. Number of epochs is fixed at 10000, and the learning rate is set equal to 0.001.

3. Then we train neural network to minimize sum of square roots of squared errors of Euler equations for each value of capital on the grid and each possible value of z.

$$\frac{1}{c^z(k_i)} = \\ = \beta \sum_{z' \in Z} \Pr(z'|z) \left\{ \frac{1}{c^{z'}(k'^z(k_i))} \left[\alpha e^{z'} k'^z(k_i)^{\alpha-1} l^{z'}(k'^z(k_i))^{1-\alpha} + (1-\delta) \right] \right\} \forall i, z$$

   Number of epochs is fixed at 1500, and the learning rate is set equal to 0.001. When updating parameters of the neural network the gradient is limited by the value of 1 to increase stability. More specifically, given some k_i and z the error of the Euler equation is computed as follows:

   (a) Compute associated value of current labor l using the neural network.

   (b) Compute associated c and k' using first and third equations of the system (4).

   (c) Compute associated values of future labor $l^{z'}$ for each possible realization of next period technology shock z' using the neural network. Then compute corresponding $c^{z'}$ for each possible realization of next period technology shock. Use transition probabilities to compute right hand side of the Euler equation.

   (d) Compute the difference between left and right hand sides of the Euler equations.

4. This allows us to get a neural network which produces policy functions for labor for each possible realization of z. Then policy functions for consumption and next period capital are obtained using first and third equations of the system (4).

The algorithm takes more than an hour to run. Figure (7) compares policy functions obtained using the neural network with policy functions obtained using generalized endogenous grid method (see section 6 for details on the implementation of generalized EGM).

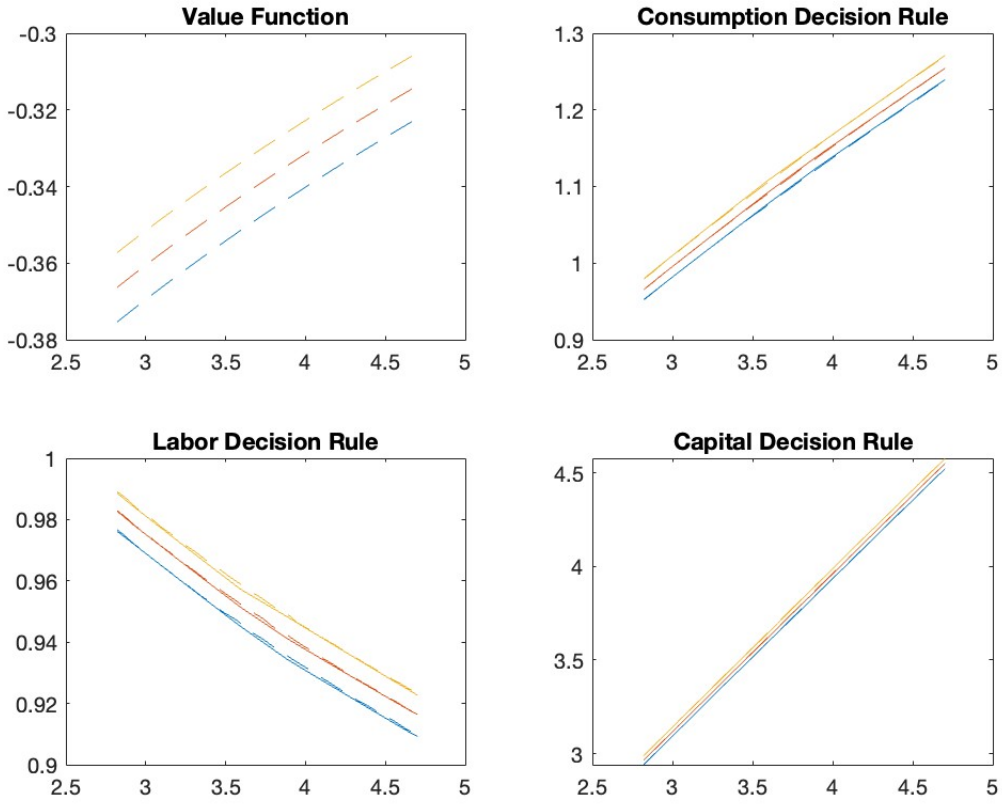


Figure 7: Comparison of policy functions obtained using neural network (solid lines) with policy functions from generalized endogenous grid method (dashed lines; yellow lines correspond to largest realization of current technology shock, red – to medium realization and blue – to smallest realization).

One can see that policy functions obtained using two methods are almost identical. So, the neural network converged to the proper solution of the model. However, we can work on the efficiency of the algorithm as it takes a long time to converge.

We can also use policy functions to simulate the model. Figure (8) shows distribution of main variables of the model obtained from simulating the model for 10000 periods (with first 1000 dropped).

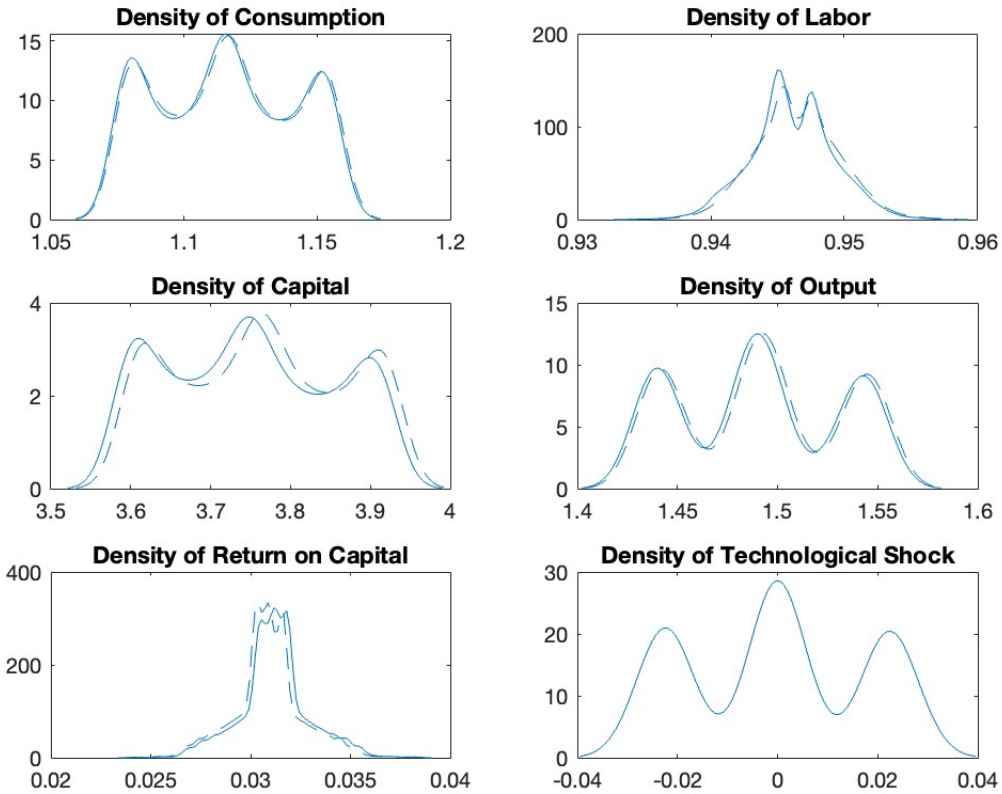


Figure 8: Comparison of distributions of model's variables obtained from simulating the model using policy functions obtained using neural network (solid lines) and policy functions from generalized endogenous grid method

Distributions obtained using two methods are almost identical. However, distribution of labor, capital and consumption obtained from neural network are shifted a little to the left as corresponding policy functions for labor are biased downwards.

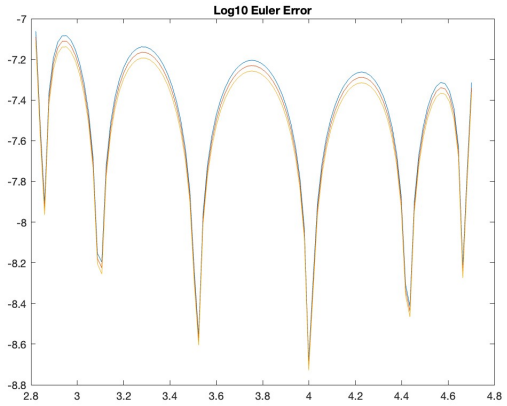
5 Comparison

In sections 1-4 we have compared policy functions obtained from each method with policy functions obtained using generalized endogenous grid method. So, approximation with Chebychev polynomials and finite elements produce identical policy functions which are very close to policy functions obtained using generalized EGM. Using third order perturbation produces policy functions which are quite different from the ones obtained using first two methods when we deviate from the steady state. This is the drawback of linearizing the model around the deterministic steady state. However, in this case we do not have to approximate technological process with a 3 state Markov chain, and the algorithm still runs very fast. Using the neural network, we are able to obtain policy functions which are very close to policy functions obtained from first two methods.

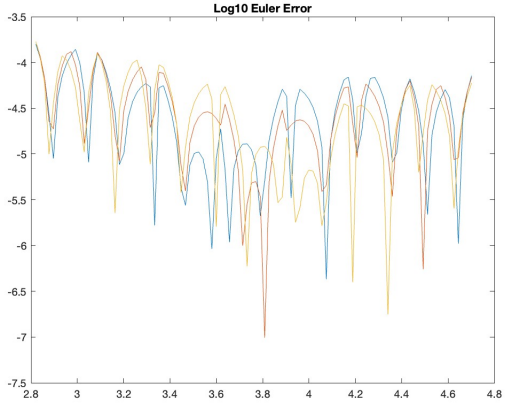
In terms of speed, approximation using Chebychev polynomials is the fastest one. The code

takes only 0.3 seconds to run. Third order perturbation obtained using Dynare takes about 2 seconds to be completed which is also very fast. Approximation using finite elements is slightly slower, and the code takes 30 seconds to run. Neural network is the slowest methods and takes more than an hour to converge to solution which is approximately correct.

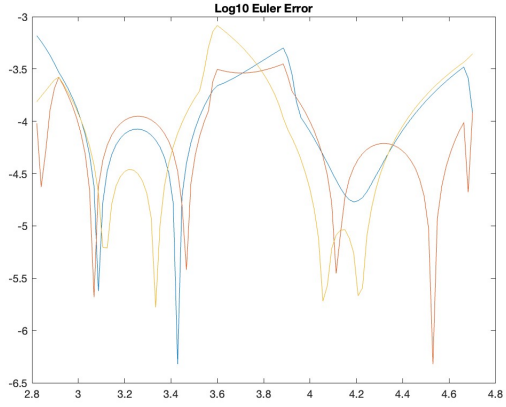
Finally, let's compare accuracy of algorithms. Figure 9 plots log10 of Euler equations for each of 100 points on the capital grid and 3 possible levels of technology shock (third order perturbation is obviously not very accurate as was demonstrated by the comparison of the policy functions in section 3).



(a) log10 of Euler equations when using Chebychev polynomials



(b) log10 of Euler equations when using finite elements



(c) log10 of Euler equations when using neural network

Figure 9: log10 of Euler equations for three methods: Chebychev polynomials, finite elements and neural network

So, approximation with Chebychev polynomials produces smallest errors of the Euler equation. Errors associated with using finite elements are slightly higher. Errors associated with using the neural network are even larger. In general, all three methods produce errors which are relatively small.

Overall, approximation with Chebychev polynomials is the fastest and the most accurate method to solve the model (among the methods we considered).

6 Generalized Endogenous Grid

Code which produces results for this section: EGM.m.

In this part we are going to use the generalized endogenous grid method proposed by Barillas, Fernandez-Villaverde (2007) to obtain the solution to our model.

Bellman equation of social planner for our model reads:

$$\begin{aligned} V(k,z) = \max_{c,k',l} & \left\{ (1-\beta)\left[\log c - \frac{l^2}{2}\right] + \beta \sum_{z'} \Pr\left(z'|z\right) V\left(k',z'\right) \right\} \\ & c + k' - (1-\delta)\,k \leqslant e^z k^\alpha l^{1-\alpha} \end{aligned}$$

FOCs:

$$\begin{aligned} [c] &: (1-\beta)\frac{1}{c} = \lambda \\ [l] &: l = (1-\alpha)\, e^z k^\alpha l^{-\alpha}\frac{1}{c} \\ [k'] &: \beta \sum_{z'} \Pr\left(z'|z\right) V'\left(k',z'\right) = (1-\beta)\frac{1}{c} \\ [Env] &: V'(k,z) = \lambda\left[\alpha e^z k^{\alpha-1} l^{1-\alpha} + 1 - \delta\right] \\ \Rightarrow & \beta \sum_{z'} \Pr\left(z'|z\right) \frac{1}{c\left(k'\left(k,z\right),z'\right)}\left[\alpha e^{z'} k'(k,z)^{\alpha-1} l\left(k'\left(k,z\right),z'\right)^{1-\alpha} + 1 - \delta\right] = \frac{1}{c\left(k,z\right)} \\ & c(k,z) = (1-\alpha)\, e^z k^\alpha l(k,z)^{-\alpha-1} \\ & k'(k,z) = e^z k^\alpha l(k,z)^{1-\alpha} + (1-\delta)\,k - c(k,z) \end{aligned}$$

We are going to solve the problem in two steps:

1. Fix labor at l_{ss} and use endogenous grid method to obtain guess for policy functions
2. Use VFI to solve the overall problem

Step 1. Endogenous grid with $l = l_{ss}$

Our problem becomes:

$$\begin{aligned} V(k,z) = \max_{c,k'} & \left\{ (1-\beta)\left[\log c - \frac{l_{ss}^2}{2}\right] + \beta \sum_{z'} \Pr\left(z'|z\right) V\left(k',z'\right) \right\} \\ & c + k' - (1-\delta)\,k \leqslant e^z k^\alpha l_{ss}^{1-\alpha} \end{aligned}$$

Let's use the change of variable and use cash-on-hand y as a state:

$$y = e^z k^\alpha l_{ss}^{1-\alpha} + (1-\delta)\,k$$

Define the following value function:

$$\tilde{V}(y,z) = \max_{k'} \left\{ (1-\beta) \left[\log\left(y - k'\right) - \frac{l_{ss}^2}{2} \right] + \beta \sum_{z'} \Pr\left(z'|z\right) \tilde{V}\left(e^{z'} k'^{\alpha} l_{ss}^{1-\alpha} + (1-\delta) k', z' \right) \right\}$$

Denote $\mathcal{V}(k',z) = \beta \sum_{z'} \Pr(z'|z) \tilde{V}\left(e^{z'} k'^{\alpha} l_{ss}^{1-\alpha} + (1-\delta) k', z' \right)$. Then the value function can be written as:

$$\begin{aligned} \tilde{V}(y,z) = & \max_{k',c} \left\{ (1-\beta) \left[\log c - \frac{l_{ss}^2}{2} \right] + \mathcal{V}\left(k',z\right) \right\} \\ & c = y - k' \end{aligned}$$

FOC:

$$\begin{aligned} [c] &: (1-\beta) \frac{1}{c} = \lambda \\ [k'] &: \mathcal{V}'\left(k',z\right) = \lambda \\ \Rightarrow & c = (1-\beta) \frac{1}{\mathcal{V}'(k',z)} \end{aligned}$$

Define a grid on capital tomorrow, $G_{k'} = \{k_1, k_2, ..., k_M\}$. Compute a grid of values of market resources in the next period implied by $G_{k'}$. $G_{y'}$ is a matrix of size $M \times N$ computed using the following formula: $y' = e^{z'} k'^{\alpha} l_{ss}^{1-\alpha} + (1-\delta) k'$.

The algorithm works as follows:

1. Set $n = 0$ and guess $\mathcal{V}^0(k',z)$. Initial guess on the value function $\mathcal{V}^0(k',z)$ has to be increasing in capital. We use $G_{y'}$ as an initial guess.

2. Compute the derivative of $\mathcal{V}^n(k',z)$ at the points on $G_{k'}$ to obtain $\mathcal{V}^{n\prime}(k',z)$ and compute the optimal level of consumption

$$c\left(k',z\right) = (1-\beta) \frac{1}{\mathcal{V}^{n\prime}(k',z)}$$

   We can use the average of the slopes of the linearly interpolated value function to obtain these derivatives. Given this optimal level of consumption, we compute the value of the endogenously determined market resources

$$y\left(k',z\right) = c\left(k',z\right) + k'$$

3. Update value function

$$\tilde{V}^{n+1}\left(y\left(k',z\right),z\right) = (1-\beta) \left[\log c\left(k',z\right) - \frac{l_{ss}^2}{2} \right] + \mathcal{V}^n\left(k',z\right)$$

Our new guess should be defined on tomorrow's market resources grid $G_{y'}$. Hence, we interpolate $\tilde{V}^n\left(y\left(k',z\right),z\right)$ so that first argument takes values from $G_{y'}$ for corresponding value of z.

4. Compute $\mathcal{V}^{n+1}\left(k',z\right)$

$$\mathcal{V}^{n+1}\left(k',z\right)=\beta\sum_{z'}\Pr\left(z'|z\right)\tilde{V}^{n+1}\left(y',z'\right)$$

5. Check convergence of $\mathcal{V}^{n+1}\left(k',z\right)$

6. After the algorithm has converged, obtain $k'\left(k,z\right)$ from $y\left(k',z\right)$. First, get $k\left(k',z\right)$ from solving this equation:

$$y\left(k',z\right)=e^{z}k\left(k',z\right)^{\alpha}l_{ss}^{1-\alpha}+\left(1-\delta\right)k\left(k',z\right)$$

Then invert it to obtain $k'\left(k,z\right)$.

Step 2. Take labor into account

Now our problem is:

$$\begin{aligned}V\left(k,z\right)=\max_{c,k'}&\left\{\left(1-\beta\right)\left[\log c-\frac{l^{2}}{2}\right]+\beta\sum_{z'}\Pr\left(z'|z\right)V\left(k',z'\right)\right\}\\c+k'-\left(1-\delta\right)k&\leqslant e^{z}k^{\alpha}l^{1-\alpha}\end{aligned}$$

Let's use the change of variable and use cash-on-hand y as a state:

$$y=e^{z}k^{\alpha}l\left(k,z\right)^{1-\alpha}+\left(1-\delta\right)k$$

$$\tilde{V}\left(y,z\right)=\max_{k',l}\left\{\left(1-\beta\right)\left[\log\left(y-k'\right)-\frac{l^{2}}{2}\right]+\beta\sum_{z'}\Pr\left(z'|z\right)\tilde{V}\left(e^{z'}k'^{\alpha}l\left(k',z'\right)^{1-\alpha}+\left(1-\delta\right)k',z'\right)\right\}$$

Denote $\mathcal{V}\left(k',z\right)=\beta\sum_{z'}\Pr\left(z'|z\right)\tilde{V}\left(e^{z'}k'^{\alpha}l\left(k',z'\right)^{1-\alpha}+\left(1-\delta\right)k',z'\right)$. Then

$$\begin{aligned}V\left(k,z\right)=\tilde{V}\left(y,z\right)=\max_{k',c,l}&\left\{\left(1-\beta\right)\left[\log c-\frac{l^{2}}{2}\right]+\mathcal{V}\left(k',z\right)\right\}\\c&=y-k'\end{aligned}$$

FOC:

$$[c] : (1-\beta)\frac{1}{c} = \lambda$$
$$[k'] : \mathcal{V}'\left(k', z\right) = \lambda$$
$$\Rightarrow c = (1-\beta)\frac{1}{\mathcal{V}'\left(k', z\right)}$$
$$[l] : l = \left[\frac{1}{c}(1-\alpha)e^{z}k^{\alpha}\right]^{\frac{1}{1+\alpha}}$$

We are going to perform value function iteration. Use $\tilde{V}(y, z)$ from step 1 (when we fixed $l = l_{ss}$) as a guess of value function. Obtain policy functions for capital, consumption, labor and updated value function.

For given k' get optimal labor from solving the following non-linear equation

$$k' + (1-\alpha)e^{z}k^{\alpha}l\left(k, z, k'\right)^{-\alpha-1} = e^{z}k^{\alpha}l\left(k, z, k'\right)^{1-\alpha} + (1-\delta)k$$

Then get corresponding value of consumption from

$$c\left(k, z, k'\right) = (1-\alpha)e^{z}k^{\alpha}l\left(k, z, k'\right)^{-\alpha-1}$$

Then compute optimal k' by maximizing value function

$$V(k, z) = \max_{k'}\left\{(1-\beta)\left[\log c\left(k, z, k'\right) - \frac{l\left(k, z, k'\right)^{2}}{2}\right] + \beta\sum_{z'}\Pr\left(z'|z\right)V\left(k', z'\right)\right\}$$

Then get policy functions for capital, consumption, labor and updated value function.
Second, perform the rest of the generalized endogenous grid method:

1. Set $n = 0$ and compute the guess $\mathcal{V}^{0}(k', z)$ as follows

$$\mathcal{V}^{0}\left(k', z\right) = \beta\sum_{z'}\Pr\left(z'|z\right)\tilde{V}^{0}\left(k', z'\right)$$

   It is defined over $G_{k'}$.

2. Compute the derivative of $\mathcal{V}^{n}(k', z)$ at the points on $G_{k'}$ to obtain $\mathcal{V}^{n\prime}(k', z)$ and compute the optimal level of consumption

$$c\left(k', z\right) = (1-\beta)\frac{1}{\mathcal{V}^{n\prime}\left(k', z\right)}$$

3. Solve for optimal labor and current capital $\tilde{k}(k', z)$ from the budget constraint using non-

linear solver

$$l\left(\tilde{k},z\right)=\left[\frac{1}{c\left(k',z\right)}\left(1-\alpha\right)e^{z}\tilde{k}^{\alpha}\right]^{\frac{1}{1+\alpha}}$$
$$k'+c\left(k',z\right)=e^{z}\tilde{k}^{\alpha}l\left(\tilde{k},z\right)^{1-\alpha}+\left(1-\delta\right)\tilde{k}$$

4. Update value function

$$V^{n+1}\left(\tilde{k}\left(k',z\right),z\right)=\tilde{V}^{n+1}\left(y\left(k',z\right),z\right)=\left(1-\beta\right)\left[\log c\left(k',z\right)-\frac{l\left(\tilde{k}\left(k',z\right),z\right)^{2}}{2}\right]+\mathcal{V}\left(k',z\right)$$

   Our new guess should be defined on grid $G_{k'}$. Hence, we interpolate $V^{n+1}\left(\tilde{k}\left(k',z\right),z\right)$ so that first argument takes values from $G_{k'}$ for corresponding value of z.

5. Compute $\mathcal{V}^{n+1}\left(k',z\right)$

$$\mathcal{V}^{n+1}\left(k',z\right)=\beta\sum_{z'}\Pr\left(z'|z\right)\tilde{V}^{n+1}\left(k',z'\right)$$

6. Check convergence of $\mathcal{V}^{n+1}\left(k',z\right)$

7. After the algorithm has converged, obtain $k'\left(k,z\right)$ and $c\left(k,z\right)$ from $\tilde{k}\left(k',z\right)$ and $y\left(k',z\right)$. First, get $c\left(k',z\right)=y\left(k',z\right)-k'$. Then invert $c\left(k',z\right)$ and $\tilde{k}\left(k',z\right)$ to obtain $k'\left(k,z\right)$ and $c\left(k,z\right)$. Then compute policy function for labor:

$$l\left(k,z\right)=\left[\frac{1}{c\left(k,z\right)}\left(1-\alpha\right)e^{z}k^{\alpha}\right]^{\frac{1}{1+\alpha}}$$

The algorithm takes about 10 seconds to converge. Figure 10 shows value function and policy functions for consumption, labor and next period capital obtained from generalized EGM.

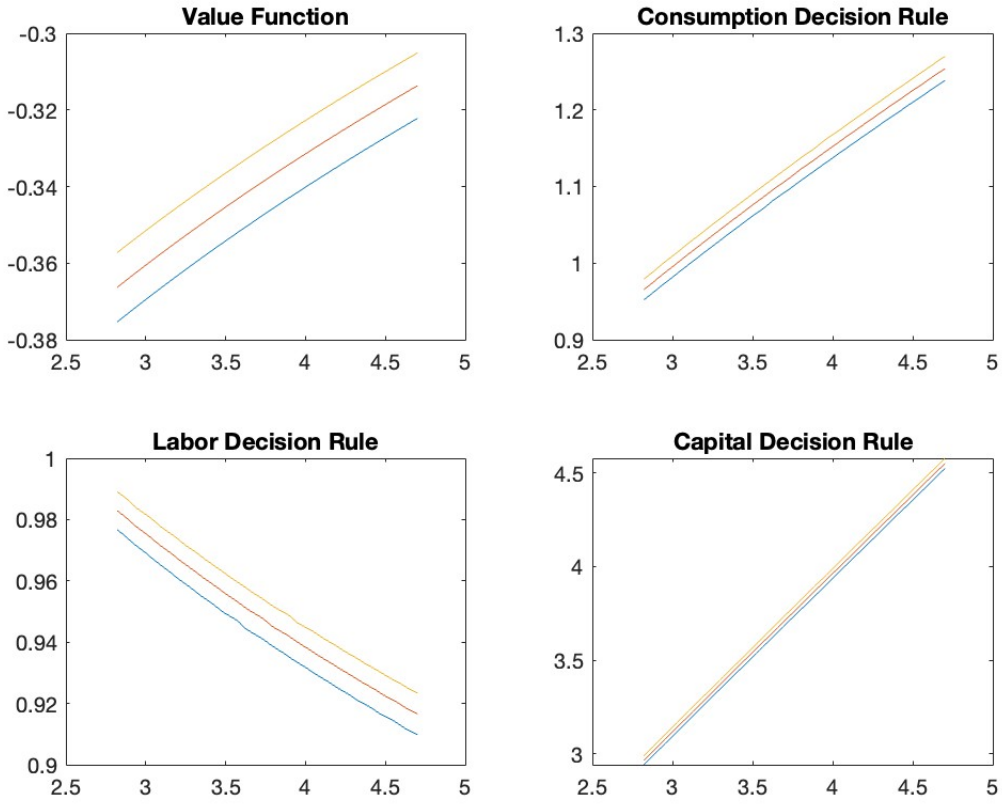


Figure 10: Value function and policy functions obtained from generalized endogenous grid method (yellow lines correspond to largest realization of current technology shock, red – to medium realization and blue – to smallest realization).