# Problem Set 2

Zhiheng You

## 1 Projection

The model can be written as:

$$\max \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \left( \log c_t - \frac{l_t^2}{2} \right)$$
$$\text{s.t. } c_t + k_{t+1} = e^{z_t} k_t^{\alpha} l_t^{1-\alpha} + (1-\delta) k_t, \forall t > 0$$
$$z_t = \lambda z_{t-1} + \sigma \varepsilon_t, \varepsilon_t \sim \mathcal{N}(0,1),$$

where $\beta = 0.97$, $\alpha = 0.33$, $\delta = 0.1$, $\lambda = 0.95$, and $\sigma = 0.007$.

I discretize the AR(1) process of $z_t$ into a three-point approximation $\{z^l, z^m, z^h\}$ using Tauchen's method. For each $z^j \in \{z^l, z^m, z^h\}$, I approximate the decision rules for labor $l\left(k, z^j\right) = \sum_{i=1}^{6} \theta_i^j \psi_i(k)$ using six Chebyshev polynomials $\{\psi_i(.)\}$, and solve for consumption policy function $c\left(k_t, z_t\right)$ from FOC (2) and next period capital from BC (3).

$$c\left(k_t, z_t\right)^{-1} = \beta \sum_{z_{t+1}} \pi(z_{t+1}|z_t) c\left(k_{t+1}, z_{t+1}\right)^{-1} \left[1 - \delta + \alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} l\left(k_{t+1}, z_{t+1}\right)^{1-\alpha}\right], \quad (1)$$
$$c\left(k_t, z_t\right) = (1-\alpha) e^{z_t} k_t^{\alpha} l\left(k_t, z_t\right)^{-\alpha-1}, \quad (2)$$
$$k_{t+1} = e^{z_t} k_t^{\alpha} l\left(k_t, z_t\right)^{1-\alpha} + (1-\delta) k_t - c\left(k_t, z_t\right). \quad (3)$$

To solve the coefficients $\theta = \{\theta_i^j\}$, I use the collocation method and solve the Euler residual function $R(k_c, z^j, \theta) = 0$ at collocation points $\{k_c\}$ and Markov states $\{z^j\}$. In total, there are 18 equations and 18 unknowns. To determine the collocation points $\{k_r\}$, Chebyshev Interpolation Theorem suggests that we only need to evaluate the function at the roots of the 6th-order Chebyshev polynomials. Hence, we map the roots $\{r_c\}$ on $[-1, 1]$ into the capital space $[0.75k_{ss}, 1.25k_{ss}]$ by $k_c = 0.5(r_c + 1)(1.25k_{ss} - 0.75k_{ss}) + 0.75k_{ss}$.

Figure 1 plots the consumption, labor, and (next period) capital policy functions computed using the projection method. We see that the policy functions are nearly linear. The consumption and capital policy functions are strictly increasing in current period capital $k$ given productivity $z$, while the labor policy function is strictly decreasing in $k$. Besides, the higher productivity level $z$ is, the higher the consumption, labor supply, and next period capital levels are.

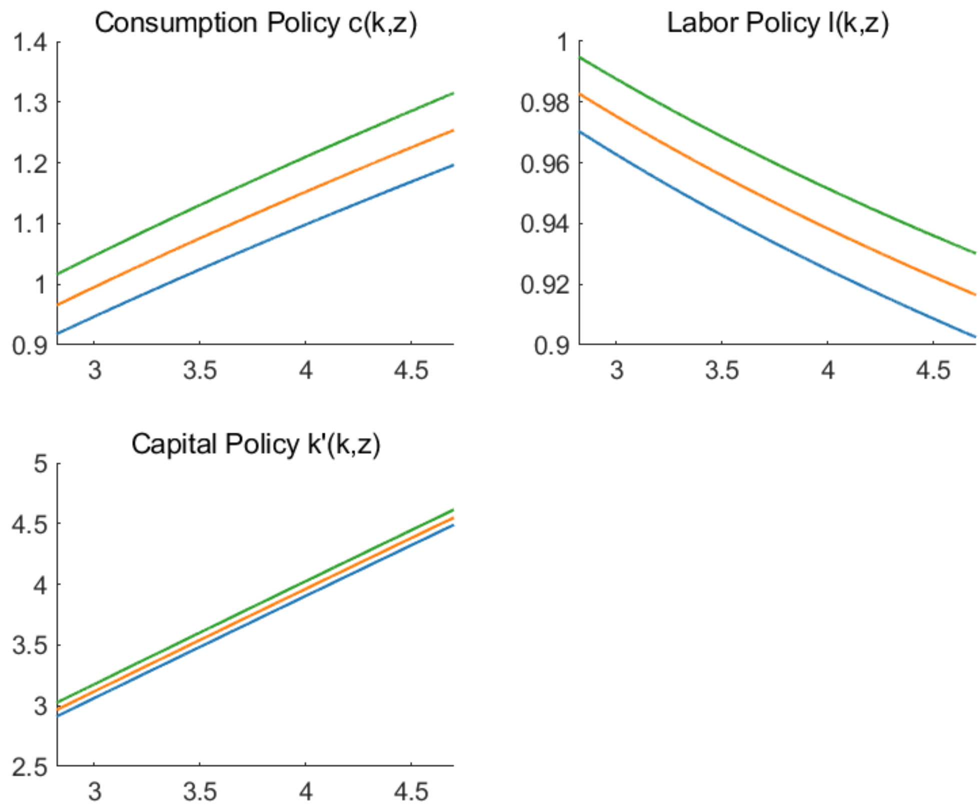


Figure 1: Policy Functions: Projection

*Notes*: Policy functions for each productivity level $z$. x-axis represents current period capital level $k$. Green line: $z = 0.067$, orange line: $z = 0$, blue line: $z = -0.067$.

Figure 2 shows the log 10 Euler error over the capital space for different productivity levels using the computed policy functions. The Euler error is defined as

$$\mathcal{EE}(k,z) = \left|1 - c\left(k,z\right) \cdot \beta \sum_{z'} \pi(z'|z) c\left(k'(k,z), z'\right)^{-1} \left[1 - \delta + \alpha e^{z'} k'(k,z)^{\alpha-1} l(k,z) \left(k'(k,z), z'\right)^{1-\alpha}\right]\right|.$$

We can see that overall the projection method performs well, with the log 10 Euler error being less than $-7$. The solution is especially accurate at collocation points, where the log 10 errors are around $-10$.

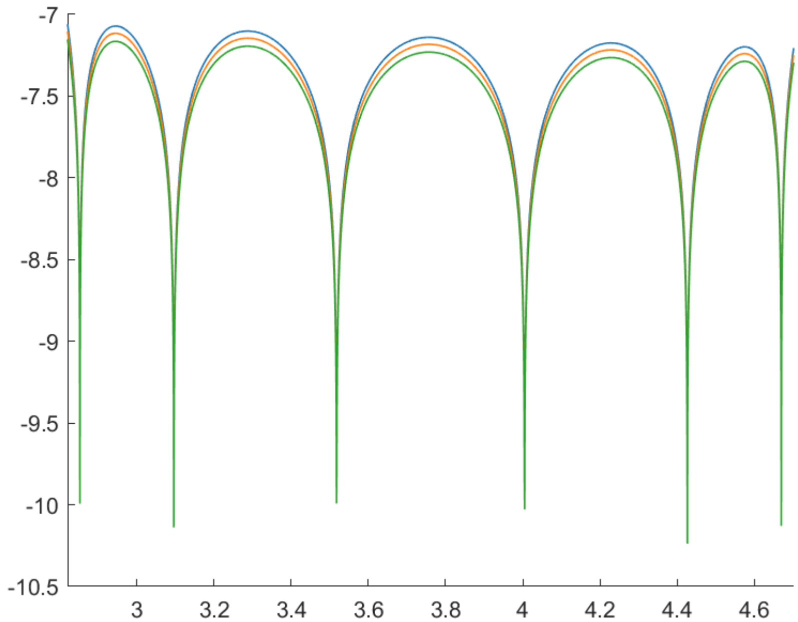


Figure 2: Log10 Euler Error Over Capital: Projection

*Notes*: Log 10 Euler error for each productivity level $z$. x-axis represents current period capital level $k$. Green line: $z = 0.067$, orange line: $z = 0$, blue line: $z = -0.067$.

# 2 Finite Element

I equally divide the capital state space $[0.75k_{ss}, 1.25k_{ss}]$ into eight subintervals $[k_{i-1}, k_i]$, $i = 2, \ldots, 9$. I discretize the AR(1) process of $z_t$ into a three-point approximation $\{z^l, z^m, z^h\}$ using Tauchen's method. For each $z^j \in \{z^l, z^m, z^h\}$, I approximate the decision rules for labor $l\left(k, z^j\right) = \sum_{i=1}^{9} \theta_i^j \Psi_i(k)$ using nine linear basis functions: for $i = 1$,

$$\Psi_1(k) = \begin{cases} \frac{k_2 - k}{k_2 - k_1} & \text{if } k \in [k_1, k_2], \\ 0 & \text{elsewhere.} \end{cases} \tag{4}$$

For $i = 2, \ldots, 8$,

$$\Psi_i(k) = \begin{cases} \frac{k - k_{i-1}}{k_i - k_{i-1}} & \text{if } k \in [k_{i-1}, k_i], \\ \frac{k_{i+1} - k}{k_{i+1} - k_i} & \text{if } k \in [k_i, k_{i+1}], \\ 0 & \text{elsewhere.} \end{cases} \tag{5}$$

For $i = 9$,

$$\Psi_9(k) = \begin{cases} \frac{k-k_8}{k_9-k_8} & \text{if } k \in [k_8, k_9], \\ 0 & \text{elsewhere.} \end{cases} \tag{6}$$

To solve the coefficients $\theta = \{\theta_i^j\}$, I use a Galerkin weighting scheme: for each $i$ and $z^j$,

$$\int_{\text{supp}(\Psi_i(k))} \Psi_i(k) R(k, z^j; \theta) dk = 0, \tag{7}$$

where

$$R(k, z^j; \theta) = c\left(k, z^j\right)^{-1} - \beta \sum_{z'} \pi(z'|z^j) c\left(k', z'\right)^{-1} \left[1 - \delta + \alpha e^{z'} k'^{\alpha-1} l\left(k', z'\right)^{1-\alpha}\right], \tag{8}$$

$$c\left(k, z\right) = (1-\alpha) e^z k^\alpha l\left(k, z\right)^{-\alpha-1}, \tag{9}$$

$$k' = e^{z^j} k^\alpha l\left(k, z^j\right)^{1-\alpha} + (1-\delta)k - c\left(k, z^j\right). \tag{10}$$

I use Gauss-Legendre quadrature when computing the integral. In total, there are 27 equations and 27 unknowns.

Figure 3 shows the consumption, labor, and capital policy functions computed from the finite element method. The results look almost the same as in Figure 1.

Figure 4 plots the log 10 Euler error over the capital space for different productivity levels. We can see that the finite element method is less accurate than the projection method, with log 10 Euler errors varying from $-3.8$ to $-5$. Also, the error plots look different when evaluated at different productivity levels: for $z = 0$, the plot is V-shape and reaches its minimum at steady state capital level $k_{ss}$, while for the other two productivity levels, the plots reach the minimum at different capital levels.

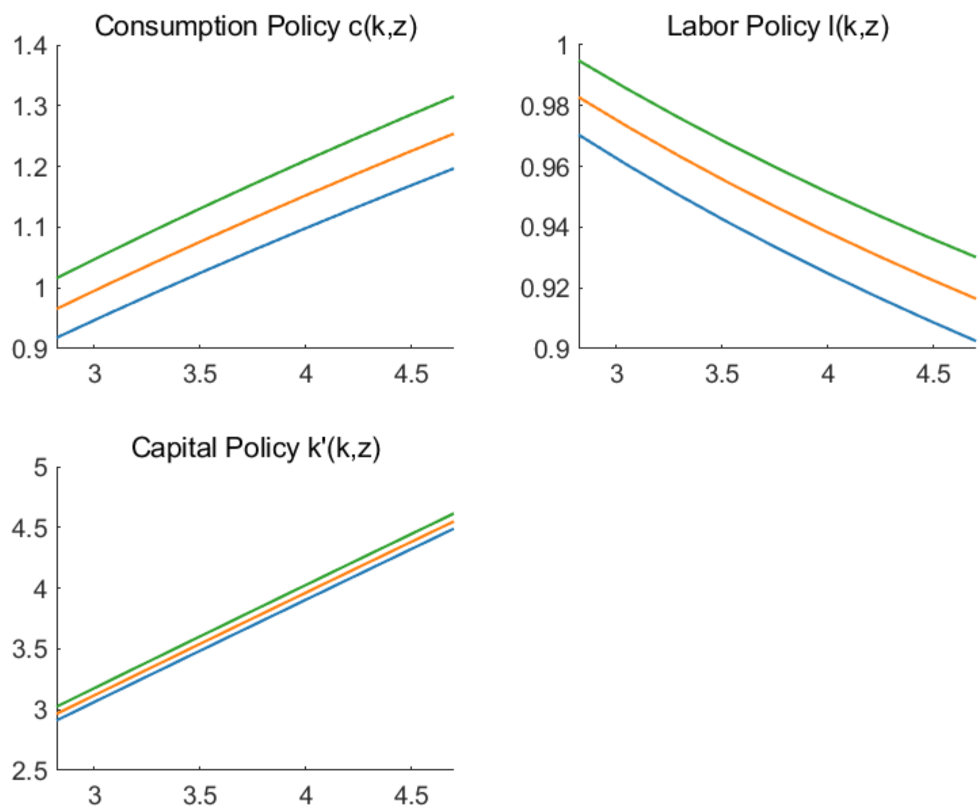


Figure 3: Policy Functions: Finite Element

*Notes*: Policy functions for each productivity level $z$. x-axis represents current period capital level $k$. Green line: $z = 0.067$, orange line: $z = 0$, blue line: $z = -0.067$.

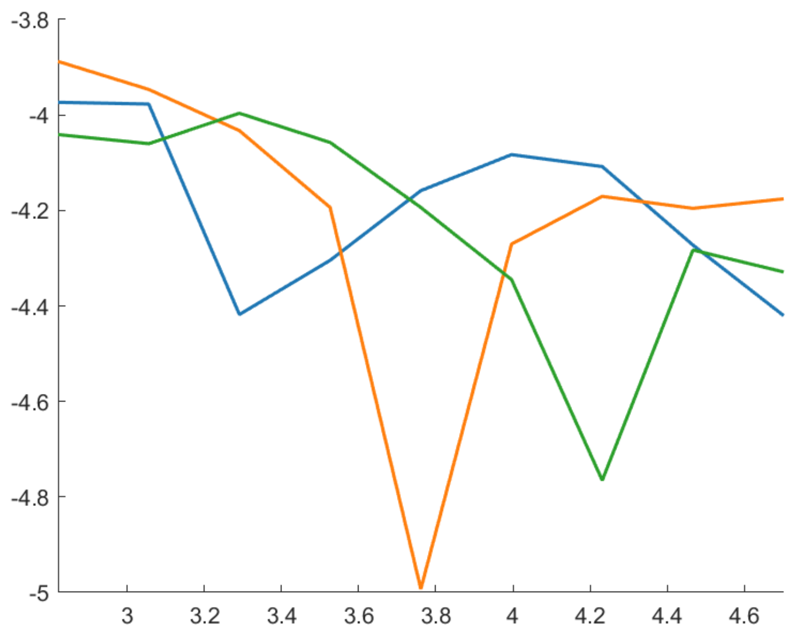


Figure 4: Log10 Euler Error Over Capital: Finite Element

*Notes*: Log 10 Euler error for each productivity level $z$. x-axis represents current period capital level $k$. Green line: $z = 0.067$, orange line: $z = 0$, blue line: $z = -0.067$.

# 3 Perturbation

The equilibrium system can be characterized by:

$$F\left(k_t, z_t; \lambda\right) = \mathbb{E}_t \begin{bmatrix} c\left(k_t, z_t; \lambda\right)^{-1} - \beta \frac{1-\delta+\alpha e^{\rho z_t+\lambda\sigma\epsilon_{t+1}} k(k_t,z_t;\lambda)^{\alpha-1} l(k(k_t,z_t;\lambda),\rho z_t+\lambda\sigma\epsilon_{t+1};\lambda)^{1-\alpha}}{c(k(k_t,z_t;\lambda),\rho z_t+\lambda\sigma\epsilon_{t+1};\lambda)} \\ c\left(k_t, z_t; \lambda\right) - (1-\alpha) e^{z_t} k_t^{\alpha} l\left(k_t, z_t; \lambda\right)^{-\alpha-1} \\ k\left(k_t, z_t; \lambda\right) + c\left(k_t, z_t; \lambda\right) - e^{z_t} k_t^{\alpha} l\left(k_t, z_t; \lambda\right)^{1-\alpha} - (1-\delta) k_t \end{bmatrix} = \mathbf{0}.$$

Dynare can solve the system in third-order around the steady state and return the derivatives of policy functions. I then use Matlab to back out the consumption policy function:

$$\begin{aligned} c(k,z) = c_{ss} &+ \frac{\partial c}{\partial k}(k_{ss}, z_{ss})\left(k-k_{ss}\right) + \frac{\partial c}{\partial z}(k_{ss}, z_{ss})\left(z-z_{ss}\right) \\ &+ \tfrac{1}{2}\left[\frac{\partial^2 c}{\partial k^2}(k_{ss}, z_{ss})(k-k_{ss})^2 + \frac{\partial^2 c}{\partial k \partial z}(k_{ss}, z_{ss})(k-k_{ss})(z-z_{ss}) + \frac{\partial^2 c}{\partial z^2}(k_{ss}, z_{ss})(z-z_{ss})^2\right] \\ &+ \tfrac{1}{6}\left[\frac{\partial^3 c}{\partial k^3}(k_{ss}, z_{ss})(k-k_{ss})^3 + \frac{\partial^3 c}{\partial k^2 \partial z}(k_{ss}, z_{ss})(k-k_{ss})^2(z-z_{ss}) \right. \\ &\qquad \left. + \frac{\partial^3 c}{\partial k \partial z^2}(k_{ss}, z_{ss})(k-k_{ss})(z-z_{ss})^2 + \frac{\partial^3 c}{\partial z^3}(k_{ss}, z_{ss})(z-z_{ss})^3\right]. \end{aligned}$$

Similar for the labor and next-period capital policy functions.

Figure 5 plots the consumption, labor, and capital policy functions computed from the perturbation method. Again, the results look almost the same as those computed before.

Figure 6 shows the log 10 Euler error over the capital space [1] for different productivity levels [2]. We observe two distinct features: first, the error level at $z = 0$ is well below those for $z = 0.067$ and $z = -0.067$; second, the plots generally exhibit a bowl-shape pattern, with the minimum error occurring near the steady-state capital level when $z = 0$. This behavior is expected since the perturbation method, being a local solution approach, provides more accurate solutions around the steady state.

[1] As the perturbation method is a local solution method, there is no such notion called "capital state space". However, I maintain the same definition of capital space for comparison purposes.

[2] When solving the model, I use the original AR(1) process for $z_t$. I evaluate the Euler error at the discretized productivity levels for comparison purposes.

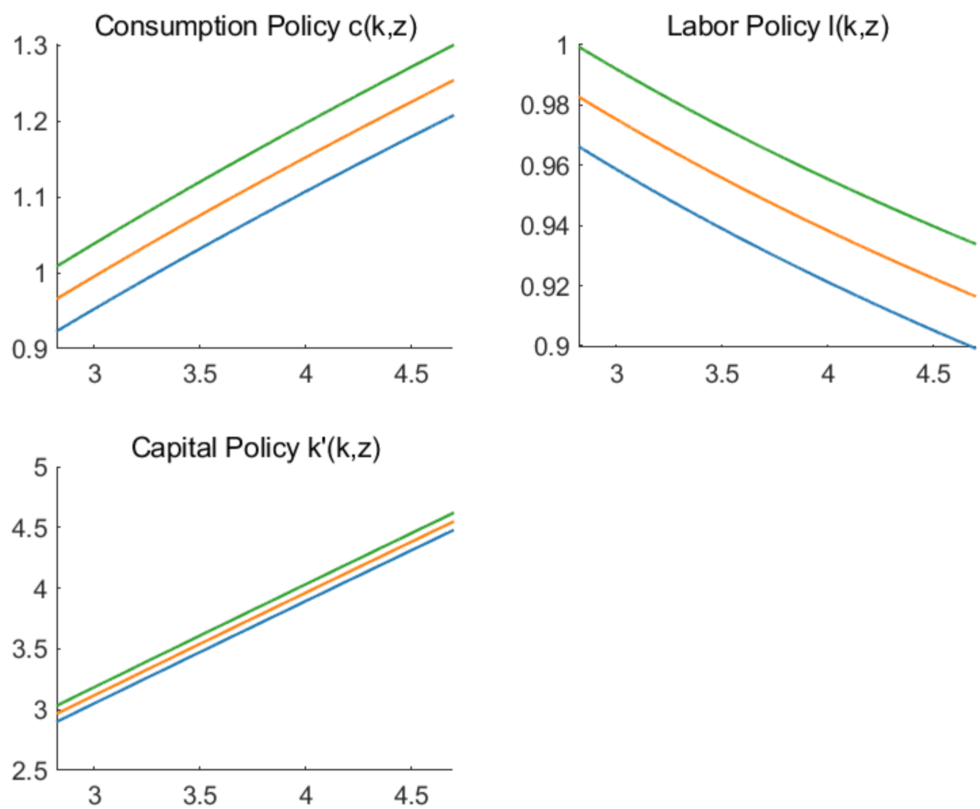


Figure 5: Policy Functions: Perturbation

*Notes*: Policy functions for each productivity level $z$. x-axis represents current period capital level $k$. Green line: $z = 0.067$, orange line: $z = 0$, blue line: $z = -0.067$.

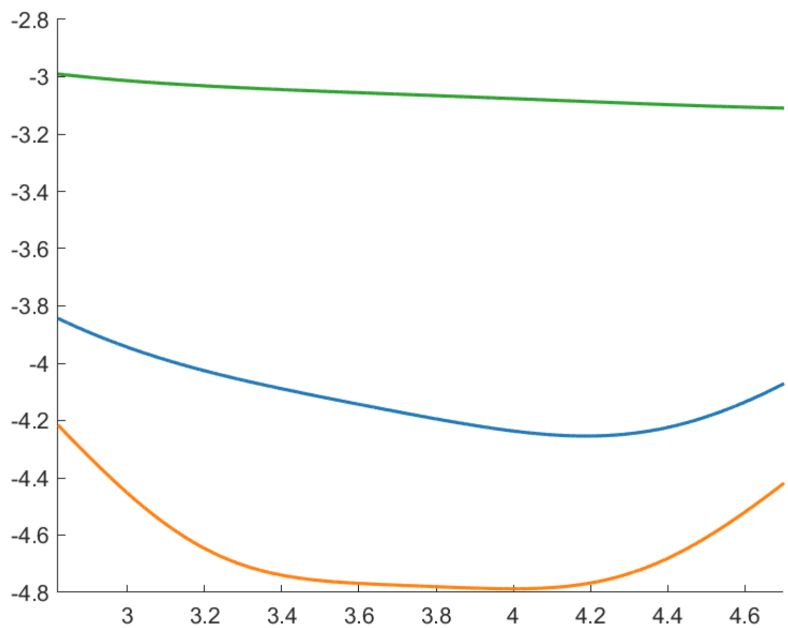


Figure 6: Log10 Euler Error Over Capital: Perturbation

*Notes*: Log 10 Euler error for each productivity level $z$. x-axis represents current period capital level $k$. Green line: $z = 0.067$, orange line: $z = 0$, blue line: $z = -0.067$.

# 4 Machine learning

I discretize the AR(1) process of $z_t$ into a three-point approximation $\{z^l, z^m, z^h\}$ using Tauchen's method. I use deep neural networks (DNNs) to approximate the labor policy function $l(k, z; \Theta^l)$: $R^2 \to R$, where $\Theta^l$ represents the parameters of the DNN. Similar for the consumption policy $c(k, z; \Theta^c)$: $R^2 \to R$ and next-period capital policy $k'(k, z; \Theta^k)$: $R^2 \to R$. Denote $\Theta = (\Theta^l, \Theta^c, \Theta^k)$.

Define the following errors:

$$\epsilon_{EE}(k, z; \Theta) = c\left(k, z\right)^{-1} - \beta \sum_{z'} \pi(z'|z) c\left(k'(k, z), z'\right)^{-1} \left[1 - \delta + \alpha e^{z'} k'(k, z)^{\alpha - 1} l\left(k'(k, z), z'\right)^{1-\alpha}\right],$$

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

$$\epsilon_{BC}(k, z; \Theta) = k'(k, z) - e^z k^\alpha l\left(k, z\right)^{1-\alpha} - (1-\delta)k + c\left(k, z\right).$$

Then the error criterion is

$$\mathcal{E}(k, z; \Theta) = ||\epsilon_{EE}(k, z; \Theta)||_2^2 + ||\epsilon_L(k, z; \Theta)||_2^2 + ||\epsilon_{BC}(k, z; \Theta)||_2^2.$$

Regarding the design of the DNN, I set the activation functions of the first two layers to Tahn and the third layer to be Softplus to ensure the non-negativity of $c$, $l$, and $k'$. The optimization of the DNN parameters is performed using ADAM. I set the batch size to 2000 and the number of training iterations to 50000.

Figure 5 plots the consumption, labor, and capital policy functions computed from DNN. As usual, the results look almost the same as those computed from previous methods.

Figure 8 demonstrates the mean log 10 Euler error over iterations. As iterations go on, the solution generally improves in accuracy, despite substantial noise during the training process.

Figure 9 displays the log 10 Euler error across the capital space for different productivity levels. Despite the flexibility of DNNs in approximating various functional forms, their accuracy in solving (nearly) linear policy functions is inferior to that of the projection method. The log 10 Euler errors range from -3 to -7, and the accuracy at $z = 0$ is generally better than the other two productivity levels.

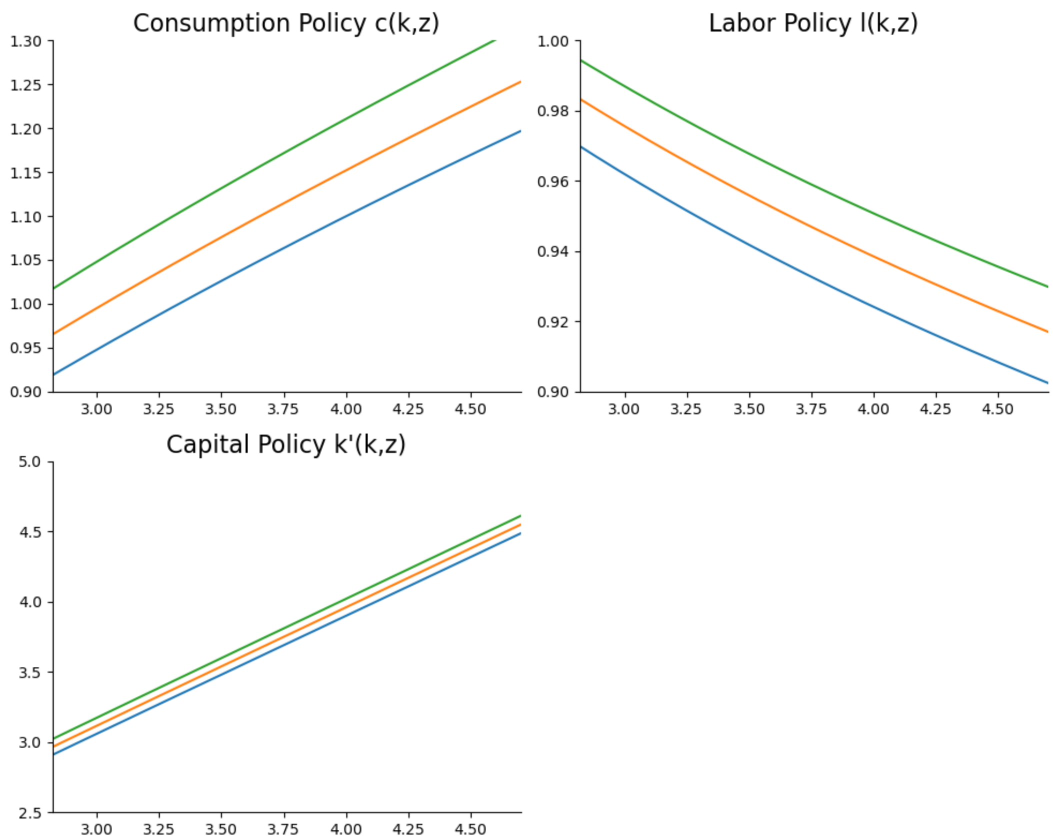


Figure 7: Policy Functions: DNN

*Notes*: Policy functions for each productivity level $z$. x-axis represents current period capital level $k$. Green line: $z = 0.067$, orange line: $z = 0$, blue line: $z = -0.067$.

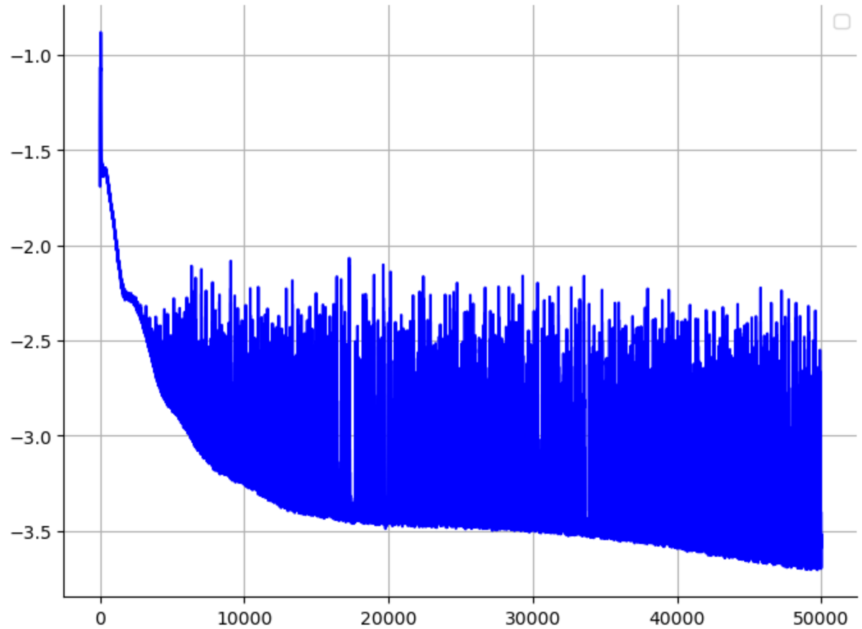


Figure 8: Log10 Euler Error Over History: DNN

*Notes*: Mean log 10 Euler error across the batch sample in each iteration. x-axis represents the number of iterations.

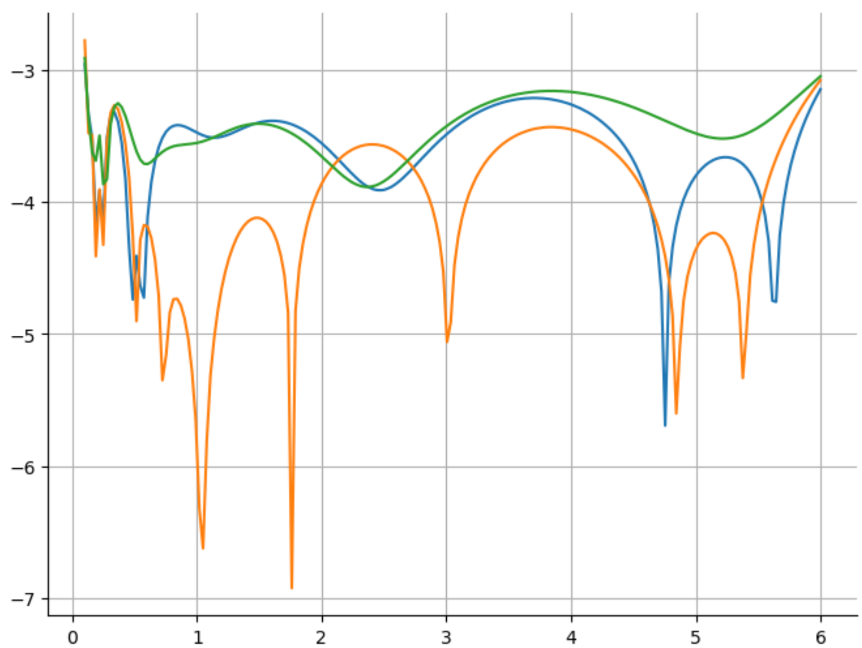


Figure 9: Log10 Euler Error Over Capital: DNN

*Notes*: Log 10 Euler error for each productivity level $z$. x-axis represents current period capital level $k$. Green line: $z = 0.067$, orange line: $z = 0$, blue line: $z = -0.067$.

# 5 Comparison

As shown in Figures 1, 3, 5, and 7, four methods generate almost identical policy functions.

Comparing the log 10 Euler error plots (Figures 2, 4, 6, and 9), we observe that in terms of accuracy: Projection > DNN > Finite Element > Perturbation. The perturbation method ranks last, as expected, since it is the only local solution method among the four and thus provides accurate solutions only near the steady state.

The projection method delivers the best performance in this setting. Given that the "true solution" in this simple model is (almost) linear, six Chebyshev polynomials are sufficient to approximate it effectively. By solving for the projection coefficients at the collocation points, the global solution's accuracy is ensured by the Chebyshev Interpolation Theorem. Another advantage of the projection method (as well as finite element) is that it is only take seconds to run, since it only involves solving a linear system of equations.

It may seem surprising that the DNN is not the most accurate method here. Generally, machine learning methods can provide highly accurate solutions for nonlinear and complex problems. However, in this case, the true solution is simple, making the DNN both unnecessary and inefficient. The DNN spends excessive time exploring the nonlinear solution space, which does not offer advantages in this context.

Lastly, the finite element method ranks between the perturbation method and the other two global solution methods. Finite element method is similar to projection as both represent the solution as a linear combination of basis functions. However, the underlying idea is different: finite element method divides the state space into different subspaces where one can use simple linear piecewise functions for approximation, while projection method directly approximates the global solution with a linear combination of nonlinear basis functions. In our setting, the projection method outperforms the finite element method, though this advantage is context-dependent.