Computational Economics – Econ 714
Prof. Jesús Fernandez-Villaverde
Homework II

Felipe Ruiz Mazin

December 10^{th}, 2018

Different Approaches for Value Function Iteration

The objective of the present assignment is to find the value and policy functions of an infinitely-lived representative household with Epstein-Zin preferences in a real business cycle model using different techniques.

The representative household's utility function at a given time t is given recursively by:

$$U_t = \left[\left(\log c_t - \eta \frac{l_t^2}{2} \right)^{\rho} + \beta \left(\mathbb{E}_t \, U_{t+1}^{\psi} \right)^{\frac{\rho}{\psi}} \right]^{\frac{1}{\rho}} \tag{1}$$

where c_t denotes period consumption and l_t denotes labor.

The representative household owns the capital stock and each period decides on how much to work. He then decides on the share of the proceedings of production that will be consumed and the one that will be saved. His budget constraint is given by:

$$c_t + i_t = w_t l_t + r_t k_t \tag{2}$$

where i_t denotes investment (savings), w_t is real wages and r_t the rate of return on capital.

The capital stock evolves according to the following law of motion:

$$k_{t+1} = (1 - \delta) k_t + i_t \tag{3}$$

The production takes aggregate capital, K_t, and aggregate labor, L_t, given values for technology e^{z_t} and capital share α_t, and gives Y_t as output according to the following function:

$$Y_t = e^{z_t} K_t^{\alpha_t} L_t^{1-\alpha_t} \tag{4}$$

Variables z_t and α_t both follow Markov chains. z_t takes values in:

$$z_t \in \{-0.0673, -0.0336, 0, 0.0336, 0.0673\} \tag{5}$$

and has the transition matrix Π_z:

$$\Pi_z = \begin{pmatrix} 0.9727 & 0.0273 & 0 & 0 & 0 \\ 0.0041 & 0.9806 & 0.0153 & 0 & 0 \\ 0 & 0.0082 & 0.9836 & 0.0082 & 0 \\ 0 & 0 & 0.0153 & 0.9806 & 0.0041 \\ 0 & 0 & 0 & 0.0273 & 0.9727 \end{pmatrix}$$

α_t takes values in the vector below:

$$\alpha_t \in \{0.25, 0.3, 0.35\} \tag{6}$$

and has the following transition matrix:

$$\Pi_\alpha = \begin{pmatrix} 0.9 & 0.07 & 0.03 \\ 0.05 & 0.9 & 0.05 \\ 0.03 & 0.07 & 0.9 \end{pmatrix}$$

1 Social Planner

The social planner for this economy chooses sequences for consumption, labor and capital to maximize equation (1) subject to the resource constraint:

$$c_t + k_{t+1} - (1-\delta)k_t = e^{z_t} k^{\alpha_t} l^{1-\alpha_t} \tag{7}$$

This problem can be rewritten in a recursive fashion as:

$$V(z,\alpha,k) = \max_{c,l,k'} \left[\left(\log c - \eta \frac{l^2}{2} \right)^\rho + \beta \left(\sum_{z'} \sum_{\alpha'} \pi_{zz'} \pi_{\alpha\alpha'} V(z',\alpha',k')^\psi \right)^{\frac{\rho}{\psi}} \right]^{\frac{1}{\rho}} \tag{8}$$

subject to (7).
A solution to the social planner's problem satisfies the following first-order conditions:

$$\begin{aligned} [k']: \quad & \frac{1}{\rho} V(z,\alpha,k)^{1-\rho} \left(\rho \left(\log\left(e^z k^\alpha l^{1-\alpha} + (1-\delta)k - k'\right) - \eta \frac{l^2}{2} \right)^{\rho-1} \times \right. \\ & \times \left(-\frac{1}{e^z k^\alpha l^{1-\alpha} + (1-\delta)k - k'} \right) + \frac{\rho}{\psi} \beta \left(\sum_{z'} \sum_{\alpha'} \pi_{zz'} \pi_{\alpha\alpha'} V(z',\alpha',k')^\psi \right)^{\frac{\rho}{\psi}-1} \times \\ & \left. \times \psi \sum_{z'} \sum_{\alpha'} \pi_{zz'} \pi_{\alpha\alpha'} V(z',\alpha',k')^{\psi-1} V_k(z',\alpha',k') \right) = 0 \end{aligned} \tag{9}$$

$$\begin{aligned} [l]: \quad & \frac{1}{\rho} V(z,\alpha,k)^{1-\rho} \rho \left(\log\left(e^z k^\alpha l^{1-\alpha} + (1-\delta)k - k'\right) - \eta \frac{l^2}{2} \right)^{\rho-1} \times \\ & \times \left(\frac{(1-\alpha) e^z k^\alpha l^{-\alpha}}{e^z k^\alpha l^{1-\alpha} + (1-\delta)k - k'} - \eta l \right) = 0 \end{aligned} \tag{10}$$

The Envelope Theorem gives us:

$$\begin{aligned} V_k(z,\alpha,k) = \frac{1}{\rho} V(z,\alpha,k)^{1-\rho} \rho \left(\log\left(e^z k^\alpha l^{1-\alpha} + (1-\delta)k - k'\right) - \eta \frac{l^2}{2} \right)^{\rho-1} \times \\ \times \left(\frac{\alpha e^z k^{\alpha-1} l^{1-\alpha} + (1-\delta)}{e^z k^\alpha l^{1-\alpha} + (1-\delta)k - k'} \right) \end{aligned} \tag{11}$$

Plug (11) into (9) to get:

$$\left(\log c - \eta\frac{l^2}{2}\right)^{\rho-1}\frac{1}{c} = \beta\left(\sum_{z'}\sum_{\alpha'}\pi_{zz'}\pi_{\alpha\alpha'}V(z',\alpha',k')^{\psi}\right)^{\frac{\rho}{\psi}-1}\times$$
$$\times\sum_{z'}\sum_{\alpha'}\left[\pi_{zz'}\pi_{\alpha\alpha'}V(z',\alpha',k')^{\psi-\rho}\left(\log c' - \eta\frac{l'^2}{2}\right)^{\rho-1}\frac{1}{c'}\left(\alpha' e^{z'}k'^{\alpha'-1}l'^{1-\alpha'} + (1-\delta)\right)\right] \quad (12)$$

By simplifying equation (10) we get:

$$(1-\alpha)e^z k^\alpha l^{-\alpha} = \left(e^z k^\alpha l^{1-\alpha} + (1-\delta)k - k'\right)\eta l \quad (13)$$

The two equations above give us the necessary conditions of an interior solution for the optimal path. Equation (12) represents the intertemporal trade-off, while equation (13) represents the intratemporal trade-off the household faces.

2 Steady State

Assume that the steady-state level for productivity is $z_{ss} = 0$ and for capital share is $\alpha_{ss} = 0.3$. Then at the steady state equations (12) and (13) become respectively:

$$\beta(\alpha_{ss}k_{ss}^{\alpha_{ss}-1}l_{ss}^{1-\alpha_{ss}} + (1-\delta)) = 1 \quad (14)$$
$$(1-\alpha_{ss})k_{ss}^{\alpha_{ss}}l_{ss}^{-\alpha_{ss}} = \left(k_{ss}^{\alpha_{ss}}l_{ss}^{1-\alpha_{ss}} - \delta k_{ss}\right)\eta l_{ss} \quad (15)$$

Since the variable labor has no predefined unit a priori. I decided to normalize it at the steady state to $l_{ss} = 50$. Once we have the values for the parameters and for l_{ss}, we can find the steady-state level of capital and the parameter η that imply $l_{ss} = 50$ by solving the system of two equations above. The parameter values are in the table below:

Table 1: Parameter values

Parameters	Value	Description
β	0.99	Discount factor
ψ	-9	Coefficient of relative risk aversion
ρ	0.5	Elasticity of intertemporal substitution
δ	0.1	Depreciation rate
η	3.8487e-4	Labor disutility (implied by choice of l_{ss})

Table 2: Steady-state values

Parameters	Value	Description
α_{ss}	0.3	Capital share
z_{ss}	0	Productivity shock
l_{ss}	50	Labor
k_{ss}	209.35	Capital stock
y_{ss}	76.83	Output
c_{ss}	55.90	Consumption

3 Value Function Iteration with a Fixed Grid

To estimate the value function, I first start with an estimation using a fixed grid of evenly spaced 250 points for capital centered around k_{ss} and covering $\pm 100y\%$ of k_{ss}. I perform the estimation by using a solver to find capital next period, k', and labor, l, and evaluating the policy function at the value k', which is likely to be off grid by using linear interpolation.

My algorithm follows the steps below:

1. Choose a level for l_{ss} and solve for the steady state.
2. Construct a grid for capital from $(1-y)k_{ss}$ to $(1+y)k_{ss}$ (I explain my choice for y below).
3. Take the period utility at the steady state as the initial guess for the value function (V_0) and k_{ss} as the initial guess for capital next period, k'.
4. At a given iteration n, calculate, using the transition matrices and the value function of found at iteration $n-1$ as a guess, the expected value next period for each level of productivity and capital share in the current period and each *on-grid* level of capital next period: $\sum_{z'}\sum_{\alpha'}\pi_{zz'}\pi_{\alpha\alpha'}V_{n-1}(z',\alpha',k')^{\psi}$, i.e. a value that is a function of z, α and k'
5. For each possible combination of capital, productivity and capital share today, use a solver, such as MATLAB's fminbnd find the k' that maximizes the value function V_n defined as:

$$V_n(z,\alpha,k) = \max_{k'}\left[\left(\log\left(e^z k^{\alpha} l(z,\alpha,k,k')^{1-\alpha} + (1-\delta)k - k'\right) - \eta\frac{l(z,\alpha,k,k')^2}{2}\right)^{\rho} + \beta\left(\sum_{z'}\sum_{\alpha'}\pi_{zz'}\pi_{\alpha\alpha'}V_{n-1}(z',\alpha',k')^{\psi}\right)^{\frac{\rho}{\psi}}\right]^{\frac{1}{\rho}} \quad (16)$$

   where $l(z,\alpha,k,k')$ is defined as the solution to (13) (in MATLAB this solution can be found through fzero).

   Since k' is almost certainly off grid, it is necessary to apply a linear interpolation to find $\sum_{z'}\sum_{\alpha'}\pi_{zz'}\pi_{\alpha\alpha'}V_{n-1}(z',\alpha',k')^{\psi}$.

6. Run the algorithm until the sup norm of the difference between the value function values at a given period and the values from the previous period is less than a certain tolerance. I picked 10^{-6} for this exercise:

$$\sup_{z,\alpha,k}|V_n(z,\alpha,k) - V_{n-1}(z,\alpha,k)| < 10^{-6} \quad (17)$$

As the algorithm is run, I update and store the policy functions and use them as guesses for the solvers in the next iteration.

For my first simulation, I followed the instructions on the problem set and chose y to be 0.3. However, this made the grid for capital too narrow and resulted in distorted labor policy functions for high shocks values, as we can see below:

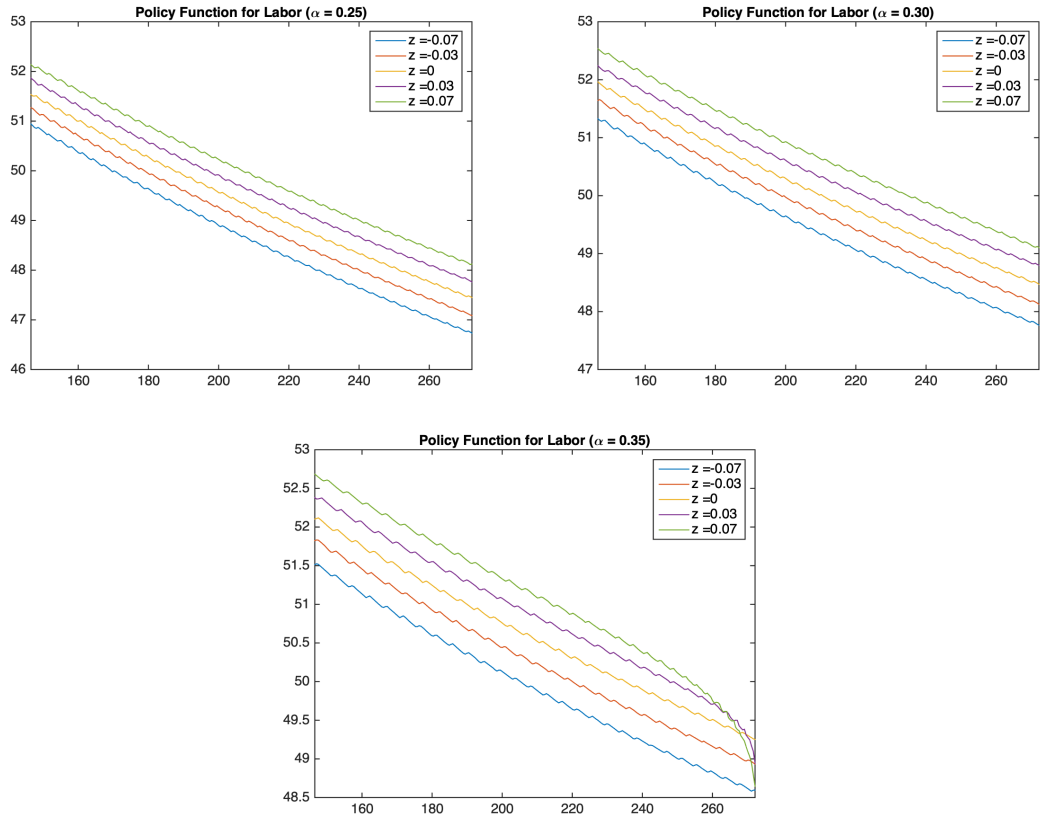

Figure 1: Estimated policy functions for labor with $\pm 30\%$ of k_{ss}

Therefore, I picked a grid of $\pm 50\%$ of k_{ss} and the results can be found in Section A.1 of the Appendix. (Running time: $17,989$ seconds or almost 5 hours for over 500 iterations). The convergence of the value function is illustrated in Figure 2 below:

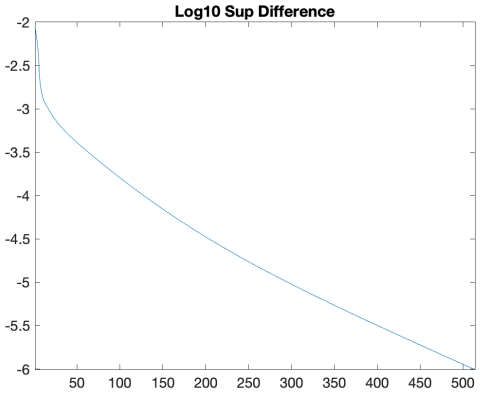

Figure 2: Value function convergence

As a measure of accuracy, especially to compare the many methods that I will analyze here, I compute the Euler errors for each method. The Euler errors are simply the difference between the left and right-hand sides of the Euler equation for each combination of states when we substitute for the estimated value and policy functions. For this algorithm, the mean of the decimal log of all Euler errors was -4.8286 and the maximum was -4.2254.

4 Value Function Iteration with an Endogenous Grid

The endogenous grid method applied to value function iteration seeks to estimate the value and policy functions of a given problem more efficiently and more accurately. The main idea behind the algorithm is to change the state variable of the model: instead of considering capital at the beginning of the period as a state variable, the method considers total market resources (defined as Y_t), which is defined as the output of the period plus the undepreciated capital:

$$Y_t \equiv c_t + k_{t+1} = y_t + (1-\delta)k_t \tag{18}$$

We can then rewrite the Bellman equation as:

$$V(z,\alpha,Y) = \max_{l,k'} \left[\left(\log(Y-k') - \eta\frac{l^2}{2} \right)^{\rho} + \beta \left(\sum_{z'} \sum_{\alpha'} \pi_{zz'}\pi_{\alpha\alpha'} V(z',\alpha',Y')^{\psi} \right)^{\frac{\rho}{\psi}} \right]^{\frac{1}{\rho}} \tag{19}$$

subject to

$$Y = e^{z'} k'^{\alpha'} l'^{1-\alpha'} + (1-\delta)k' \tag{20}$$

Define the discounted expected value on the right-hand side as $\tilde{V}$:

$$V(z,\alpha,Y) = \max_{l,k'} \left[\left(\log(Y-k') - \eta\frac{l^2}{2} \right)^{\rho} + \tilde{V}(z,\alpha,k') \right]^{\frac{1}{\rho}} \tag{21}$$

My algorithm is as follows:
First, solve the model for the steady-state level of labor:

1. Start with a guess for $\tilde{V}_0(z_t,\alpha_t,k_{t+1})$.
2. At iteration n, for every value of $\tilde{V}_n(z_t,\alpha_t,k_{t+1})$, compute the derivative at the grid points for k_{t+1} to obtain $\tilde{V}_{k_n}(z_t,\alpha_t,k_{t+1})$ and compute the optimal level of consumption using the

FOC of the problem. Given this solution compute the value of the endogenously determined market resources:

$$\hat{Y}_t(z_t, \alpha_t, \hat{k}_t) = c_t^*(z_t, \alpha_t, \hat{k}_t) + k_{t+1} \tag{22}$$

3. Update the value function as below:

$$V_n(z_t, \alpha_t, \hat{Y}_t) = \left[\left(\log(c_t^*(z_t, \alpha_t, \hat{k}_t)) - \eta \frac{l_{ss}^2}{2} \right)^{\rho} + \tilde{V}_n(z_t, \alpha_t, k_{t+1}) \right]^{\frac{1}{\rho}} \tag{23}$$

The new guess has to be computed for the grid points at the market resources grid. Interpolate $V_{n+1}(z_t, \alpha_t, \hat{Y}_t)$ on that grid with the value of $\hat{Y}_t$ computed in step 2 to obtain $V_{n+1}(z_{t+1}, \alpha_{t+1}, Y_{t+1})$.

4. Calculate $\tilde{V}_{n+1}(z_t, \alpha_t, k_{t+1}) = \beta \mathbb{E}_{z,\alpha} V_{n+1}(z_{t+1}, \alpha_{t+1}, Y_{t+1})$.

5. If $\sup_{i,j,\ell} |\tilde{V}_{n+1}(z_i, \alpha_j, k_\ell) - \tilde{V}_n(z_i, \alpha_j, k_\ell)| \geq$ 1e-6, then proceed to the next iteration (go back to step 2). If not, then the algorithm has converged.

6. After convergence, use a solver to retrieve $\hat{k}_t(z, \alpha)$ for all shocks from $\hat{Y}_t(z_i, \alpha_j, \hat{k}_t) = e^{z_i} \hat{k}_t^{\alpha_j} l_{ss}^{1-\alpha_j} + (1-\delta)\hat{k}_t$.

5 Value Function Iteration with a Fixed Grid and an Accelerator

This approach builds on the one developed in Section 3. In this case, instead of finding k' with the solver at each iteration, I do it only once at every ten iterations. In the nine iterations in between, I just update the value function. The steps of the algorithm are as follows:

1. Perform steps 1 through 4 from Section 3.

2. At a given iteration $n = 1, 11, 21, ...$, run step 5 from Section 3 and store the policy functions: $k' = g_k(z, \alpha, k)$, $l = g_l(z, \alpha, k)$ and $c = g_c(z, \alpha, k)$.

3. For all iterations $n \neq 1, 11, 21, ...$, only update the value function using the policy functions found at the last iteration $n = 1, 11, 21, ...$ Do not update the policy functions:

$$V_n(z, \alpha, k) = \left[\left(\log g_c(z, \alpha, k) - \eta \frac{g_l(z, \alpha, k)^2}{2} \right)^{\rho} + \beta \left(\sum_{z'} \sum_{\alpha'} \pi_{zz'} \pi_{\alpha\alpha'} V_{n-1}(z', \alpha', k')^{\psi} \right)^{\frac{\rho}{\psi}} \right]^{\frac{1}{\rho}} \tag{24}$$

The second term of the right-hand side of the equation above has to be calculated using linear interpolation, since k' is almost certainly off grid.

4. As before, repeat the steps until convergence:

$$\sup_{z,\alpha,k} |V_n(z, \alpha, k) - V_{n-1}(z, \alpha, k)| < 10^{-6}$$

In total, the algorithm took 1769 seconds to run, or 29 minutes and 29 seconds, about ten times faster than the regular one. Since it provided the same estimated functions as the standard algorithm (the ones found in Section A.1 of the Appendix), I will not report them. Instead, it is interesting to look at the convergence of the value function for this algorithm. Figure 3 shows us that it even took fewer iterations for the algorithm to converge as compared to the previous case, depicted in Figure 2.

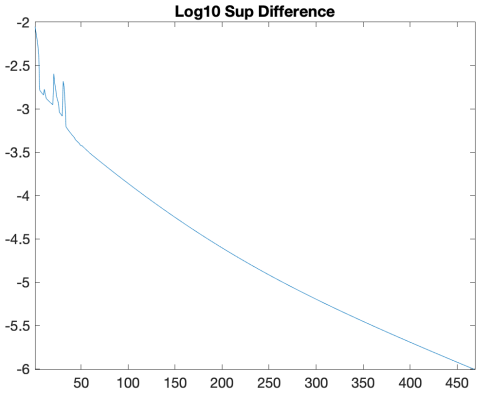

Figure 3: Value function convergence using the accelerator

The accuracy of the algorithm, as measured by the decimal log of the Euler errors, was -3.5295 (mean) and -2.4415 (maximum), which implies a much lower degree of accuracy as compared to the standard approach in Section 3.

6 Value Function Iteration using a Multigrid

Value Function Iteration using a Multigrid is a very clever way to find the value and policy functions using very fine grids. The algorithm does not differ too much from the one in Section 3, and, actually, it is just that same algorithm repeated over and over for finer and finer grids (with the same start and end points). Here, I use three grids with increasing degrees of fineness: first, one with 100 points, then with 500 and finally one with 5000. The steps are as follows:

1. Start with the coarser grid and go through all steps from Section 3 until convergence.
2. Store the vectors for the value and policy functions and use linear interpolation in order to convert these functions to functions of a finer grid with more grid points.
3. Use the interpolated functions as initial guesses for the next and finer grid. Run all steps from Section 3 again.
4. Repeat the process for finer and finer grids until the grid is fine enough for the purpose at hand.

The total running time for this code was 8,189 seconds, that is, a little bit less than two hours and a half, around half of the time spent to run the code in Section 3. This is a big improvement, since the code runs faster and the functions are much more accurate. This algorithm strictly dominates the regular one.

The plots for the value and policy functions can be found in Section A.2 of the appendix. Note that now, given the finer grid, the labor and consumption functions are less bumpy than the ones found before.

With respect to accuracy, the mean of the decimal log of the Euler errors is -4.9261 and the maximum is -4.2646, much more precise than the method with the accelerator and slightly more precise than the standard method.

7 Solution using Projection (Chebyshev Polinomials)

Projection methods are computational methods indicated to finding a global solution for a given problem. Technically, a projection method seeks to find the parameters θ that index a given function $d(x|\theta)$ defined on a given domain $X \subseteq \mathbb{R}$, that approximately solves an operator $\mathcal{H}(d)$:

$$\mathcal{H}(d) = \mathbf{0} \tag{25}$$

The operator $\mathcal{H}$ is built upon the necessary conditions for a solution and measures how far away the approximation is from the actual solution. Function d can be any function on X indexed by parameters, but it is usually taken to be a linear combination of a given number I of linearly independent basis functions defined on X where θ represents the weights assigned to each one of those functions. Mathematically:

$$d^I(\mathbf{x}|\theta) = \sum_{i=0}^{I} \theta_i \Psi_i(\mathbf{x}) \tag{26}$$

Plugging (26) back into equation (8) we get to define a residual function:

$$\mathcal{R}(\mathbf{x}|\theta) = \mathcal{H}\left(\sum_{i=0}^{I} \theta_i \Psi_i(\mathbf{x})\right) \tag{27}$$

Then we need to define which inner product to use to evaluate the residual. The sum of squared residuals is a common choice for this case. Note that different choices for the d function and for the inner product yield different solutions and it is up to the researcher to pick that one that is most adequate for the model at hand.

To find an approximation to the global solution for our problem, I picked eight Chebyshev polynomials ($\psi_i : [-1, 1] \to [-1, 1], i = 0, 1, ..., 7$) to be the basis functions and as for the inner product, I simply chose the dot product. The algorithm works as follows:

1. Choose a level for l_{ss} and solve for the steady state.
2. Construct a grid for capital from $0.5k_{ss}$ to $1.5k_{ss}$.
3. Define the number of linearly independent functions that will be used as basis functions. For this exercise I picked eight Chebyshev polynomials ($I = 7$).
4. Define the nodes at which these polynomials will be evaluated. Since we have eight linearly independent polynomials, we need eight nodes. I picked the eight roots of the polynomial of eighth order (the one that will not be part of the set of basis functions).
5. Project the nodes onto the capital grid given the minimum and maximum value chosen.
6. Each coefficient can be written as $\theta^m_{iz\alpha}$, which means that it is the weight of the i-th Chebyshev polynomial for function m evaluated at states z and α. In this case, there are two functions (value and labor functions) and so there are in total $2 \times 8 \times 5 \times 3 = 240$ coefficients to be found. We can write the value and labor functions as:

$$V\left(z, \alpha, (x+1)\frac{k_{\max} - k_{\min}}{2 + k_{\min}}\right) = \sum_{i=0}^{7} \theta^V_{iz\alpha} \psi_i(x) \tag{28}$$

$$g_l\left(z, \alpha, (x+1)\frac{k_{\max} - k_{\min}}{2 + k_{\min}}\right) = \sum_{i=0}^{7} \theta^l_{iz\alpha} \psi_i(x) \tag{29}$$

7. Evaluate equations (8) and (12) using the coefficients θ and the equations (28) and (29) above (note that for given values of current capital and shocks z and α we can calculate output, consumption and capital next period). Calculate their residuals as the difference between the left-hand side and the right-hand side of each equation.

   Since each function evaluated at given values of z and α will be evaluated at eight nodes on the capital grid, the total number of coefficients will be exactly equal to the number of function evaluations. This will allow for an exact solution for those nodes. We can then use MATLAB's solver fsolve to solve this system of 240 equations.

The total code ran in only 70 seconds. The resulting value and policy functions can be seen at Section A.3 of the Appendix. A first inspection makes us conclude that the estimated policy functions approximate the ones obtained through the value function iteration algorithms very closely (actually, if we consider the policy functions for labor and consumption, the projection method provides us with a better solution, at least one that is not bumpy).

We can then estimate the accuracy of the projections by calculating their Euler errors: estimate the value function and the policy function for labor using the coefficients, but now evaluate them at a much finer grid. Then, simply compute the difference between the left-hand side and the right-hand side of the Euler equation (12). The graph below shows the decimal log of these errors. We can see that their magnitude is very small (and very close to zero at each one of the eight nodes chosen for estimation, as expected).

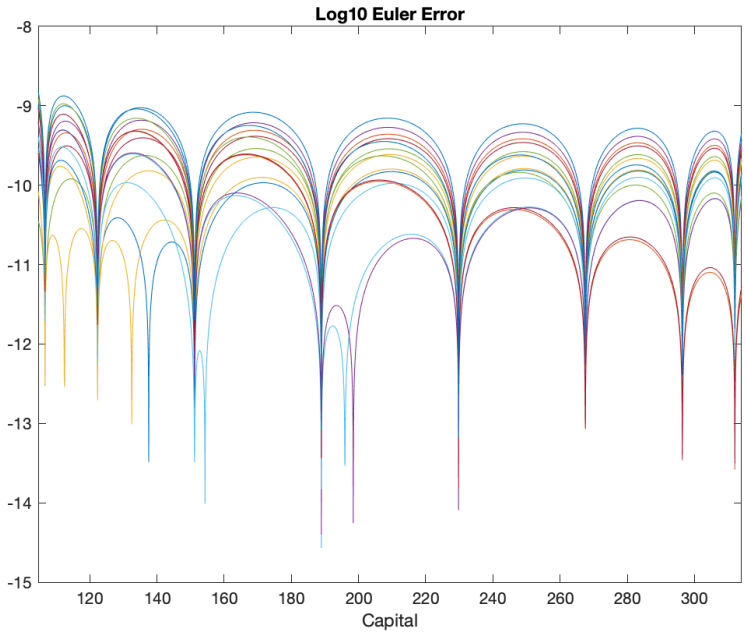

Figure 4: Euler errors - projection method

When compared to the Euler errors calculated for the previous methods, this method becomes the most accurate among all the ones analyzed here.

I also ran simulations with the resulting functions and obtained the ergodic distribution for each variable of interest. Their densities are shown in Figure 5 below.

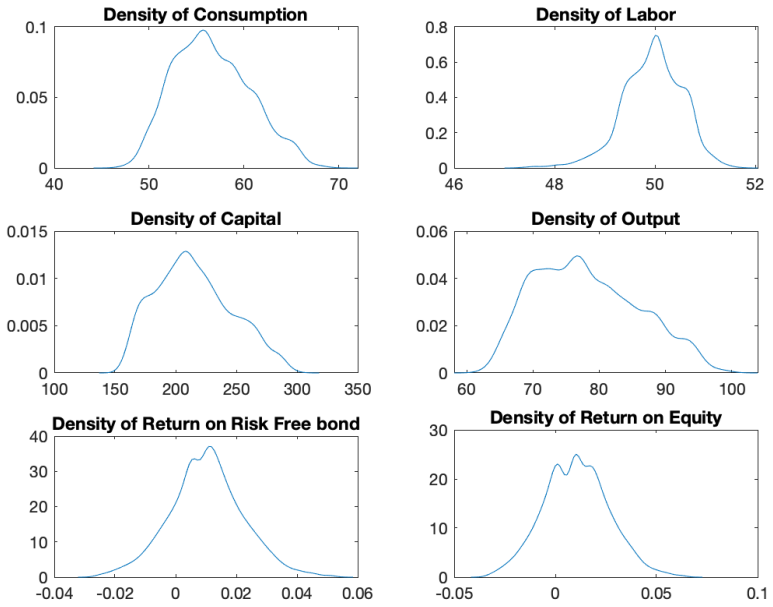

Figure 5: Ergodic densities - projection method

8 Solution using Perturbation

Perturbation methods are different from projection methods with regard to the precision of the solution. Perturbation methods are recommended for accurate local solutions, since they build Taylor series approximations around the deterministic steady state of a given model. Technically, perturbation methods solve the same functional problem as a projection: equation (8):

$$\mathcal{H}(d) = \mathbf{0}$$

The difference is that pertubation specifies a Taylor series expansion to the unknown function $d : \Omega \to \mathbb{R}^m$ in terms of the state variables of the model $\mathbf{x}$ and coefficients θ.

The following impulse response functions were estimated through a third-order Taylor approximation using dynare and show the response of capital, labor and consumption to a one-standard-deviation shock to productivity and to capital share.

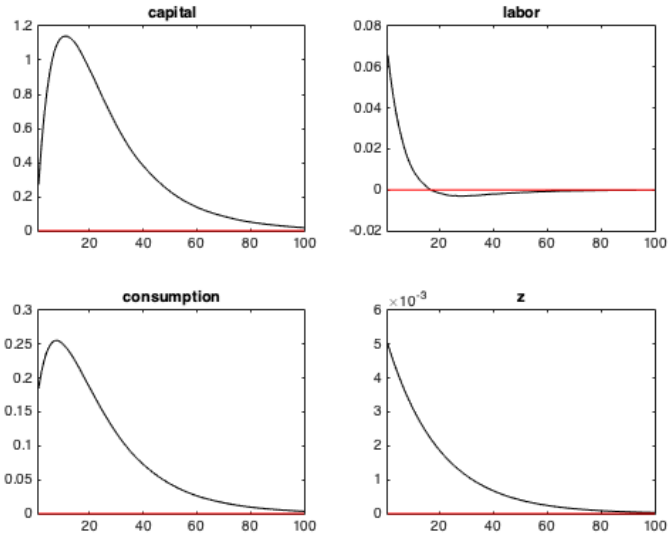

Figure 6: One-standard-deviation shock to z.

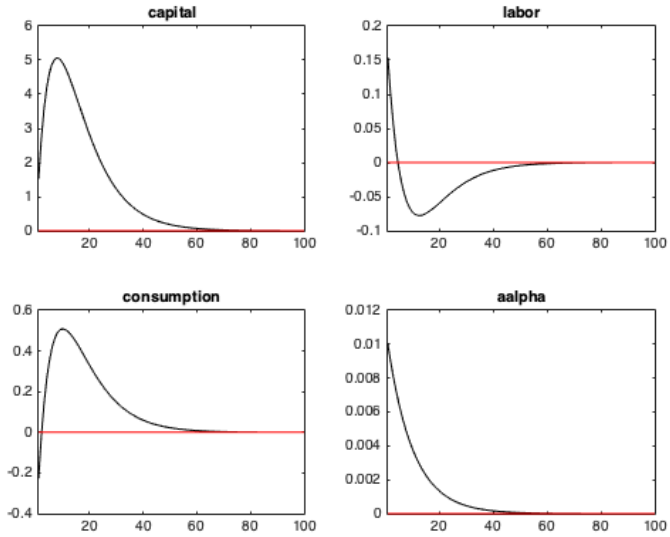

Figure 7: One-standard-deviation shock to α.

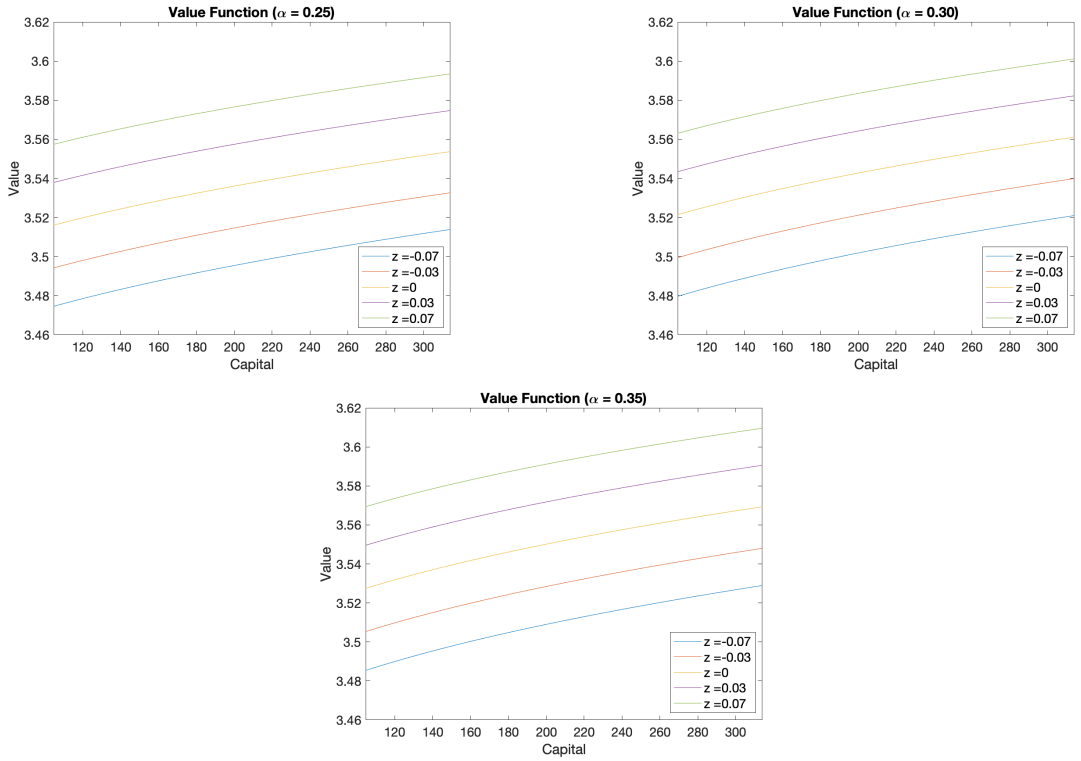

Figure 8: Estimated value functions with ±50% of k_{ss} - 250 grid points

A Appendix

A.1 Plots for the Standard Algorithm with Interpolation

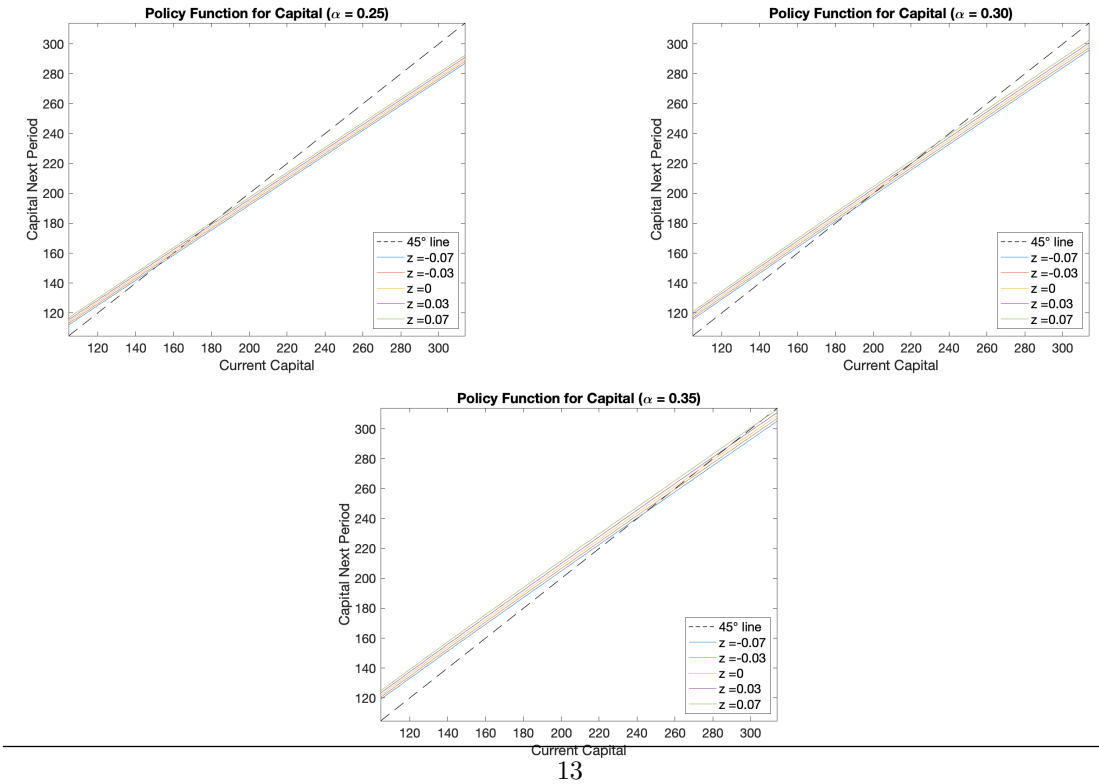

Figure 9: Estimated policy functions for capital next period with ±50% of k_{ss} - 250 grid points

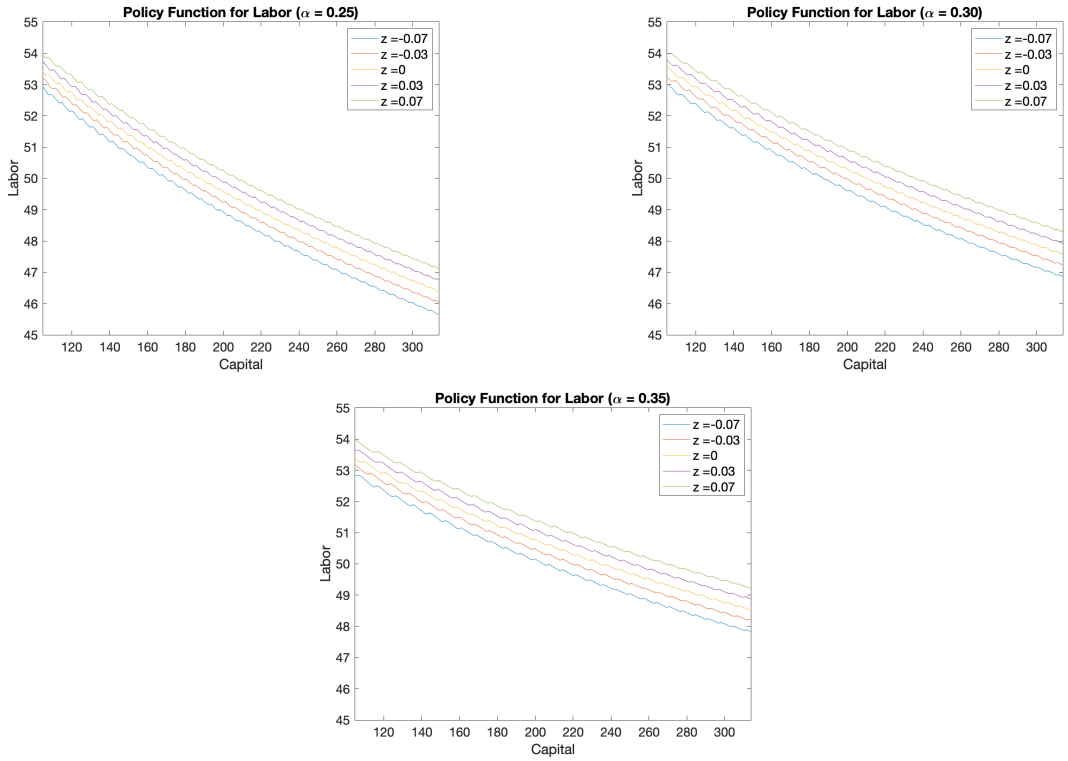

Figure 10: Estimated policy functions for labor with $\pm 50\%$ of k_{ss} - 250 grid points

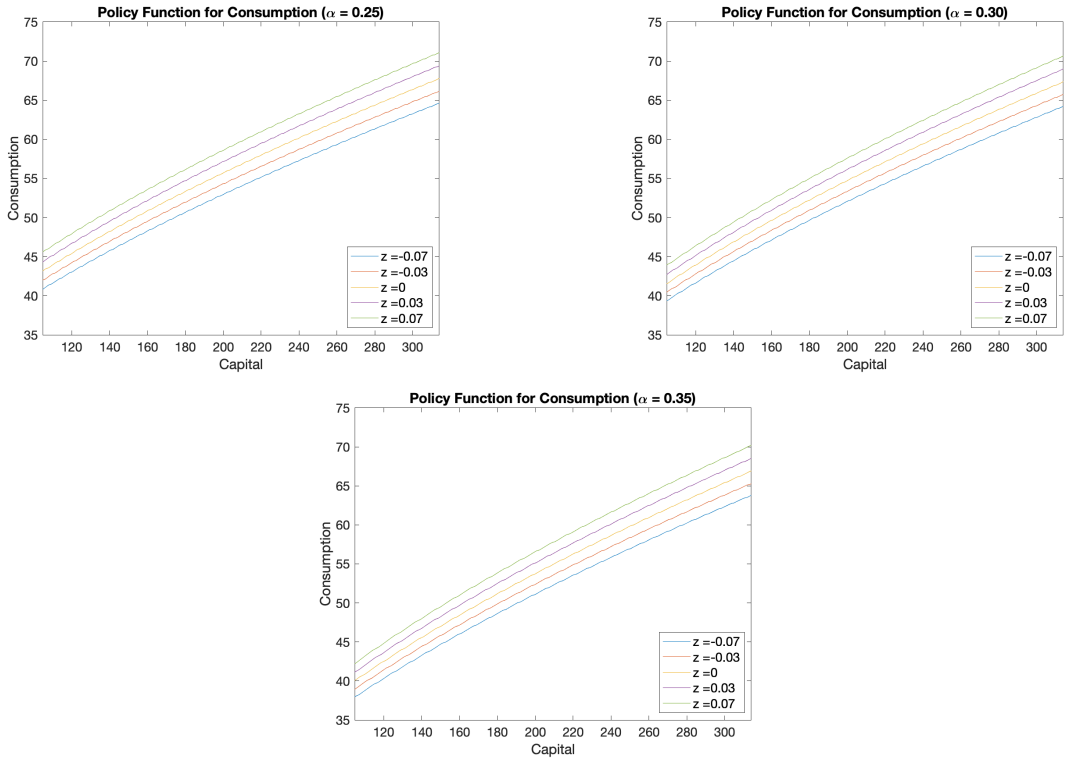

Figure 11: Estimated policy functions for consumption with $\pm 50\%$ of k_{ss} - 250 grid points

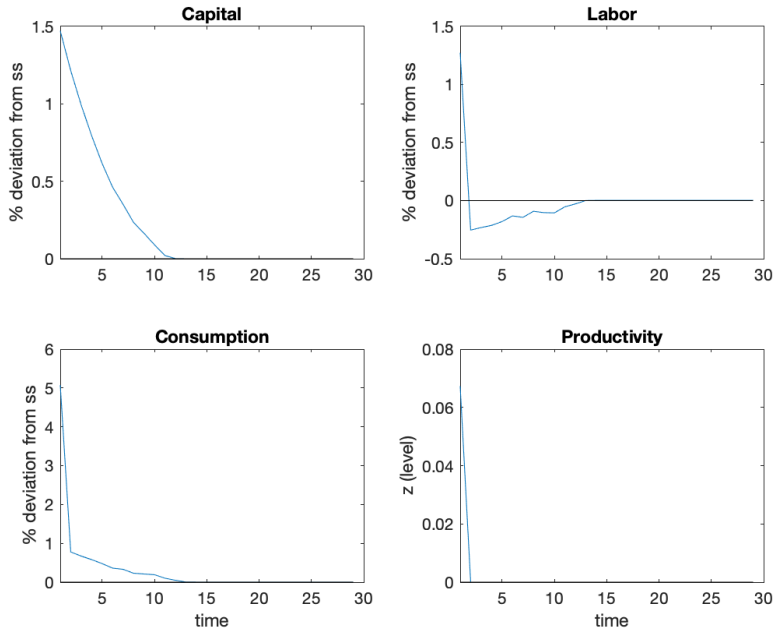

Figure 12: Impulse response functions for a positive shock on z - 250 grid points

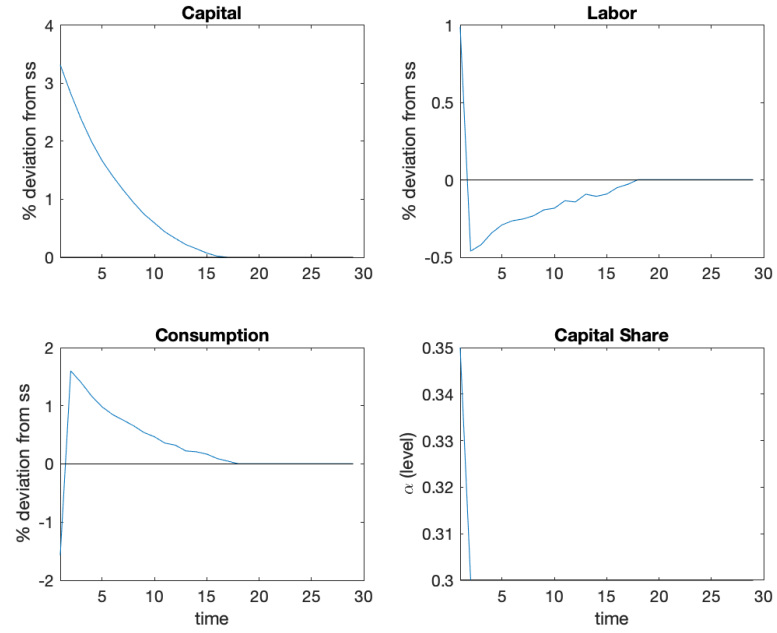

Figure 13: Impulse response functions for a positive shock on α - 250 grid points

A.2 Plots for the Multigrid Algorithm

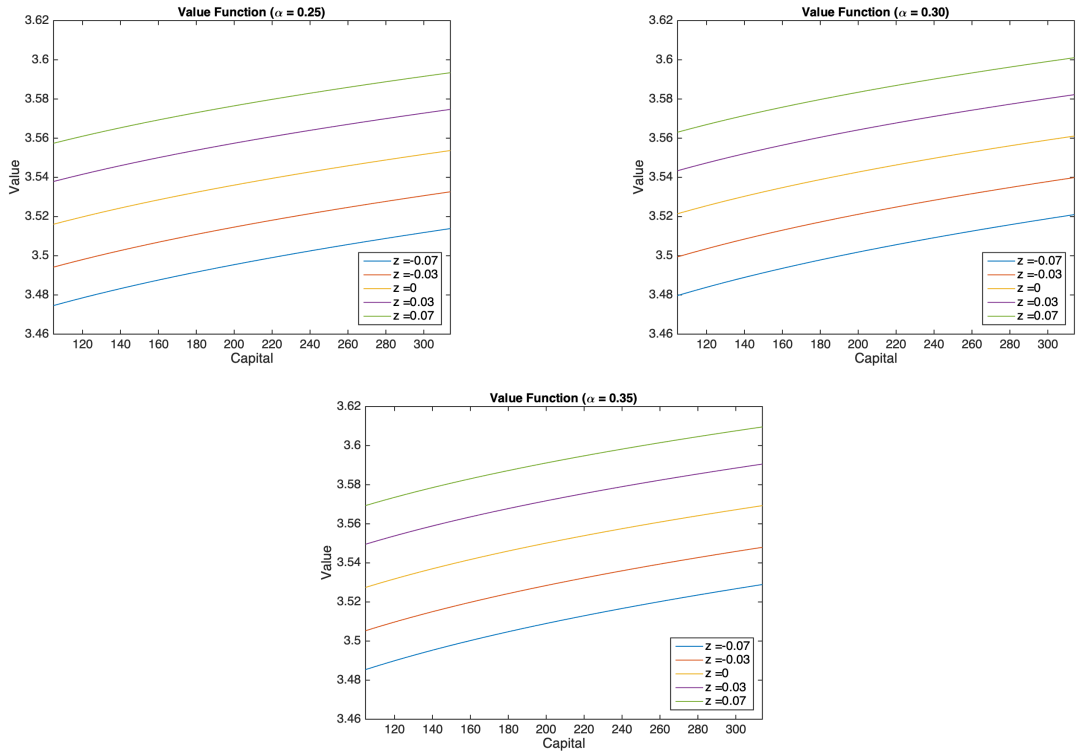

Figure 14: Estimated value functions with ±50% of k_{ss} - 5000 grid points

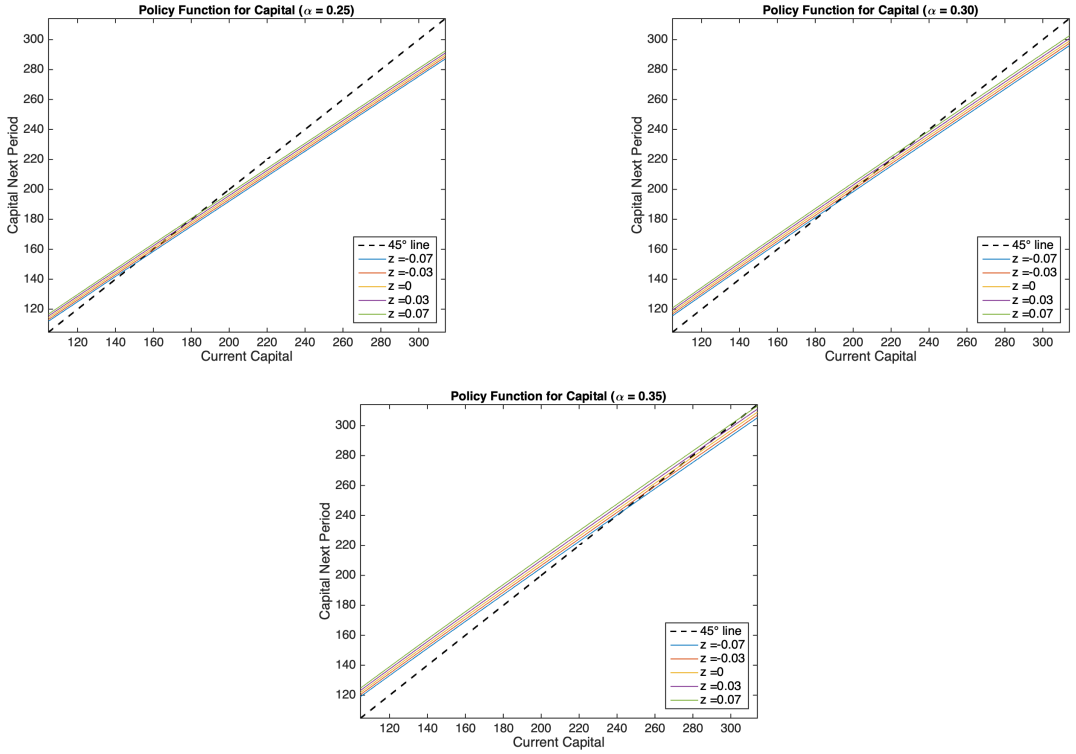

Figure 15: Estimated policy functions for capital next period with ±50% of k_{ss} - 5000 grid points

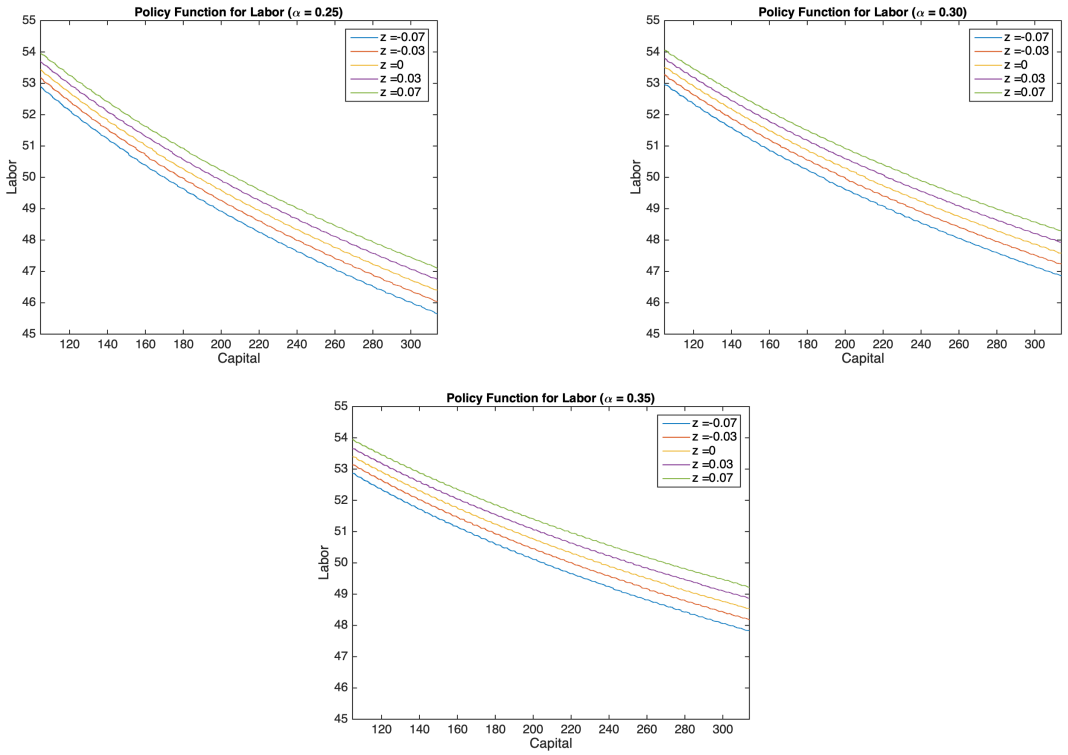

Figure 16: Estimated policy functions for labor with ±50% of k_{ss} - 5000 grid points

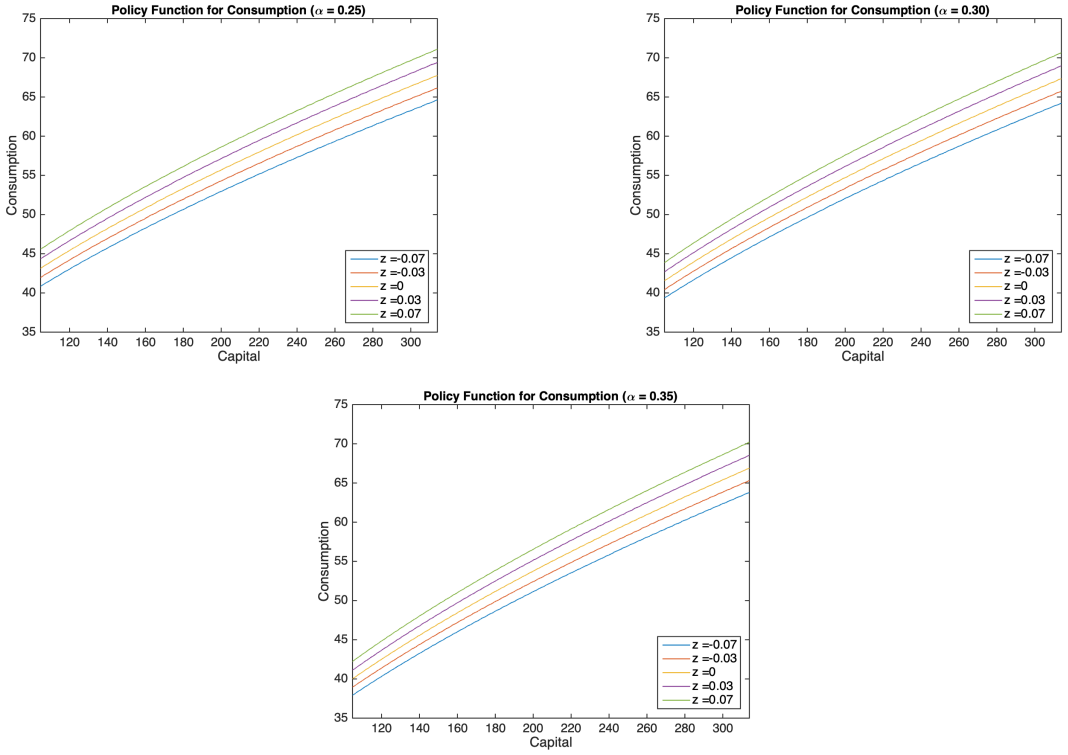

Figure 17: Estimated policy functions for consumption with ±50% of k_{ss} - 5000 grid points

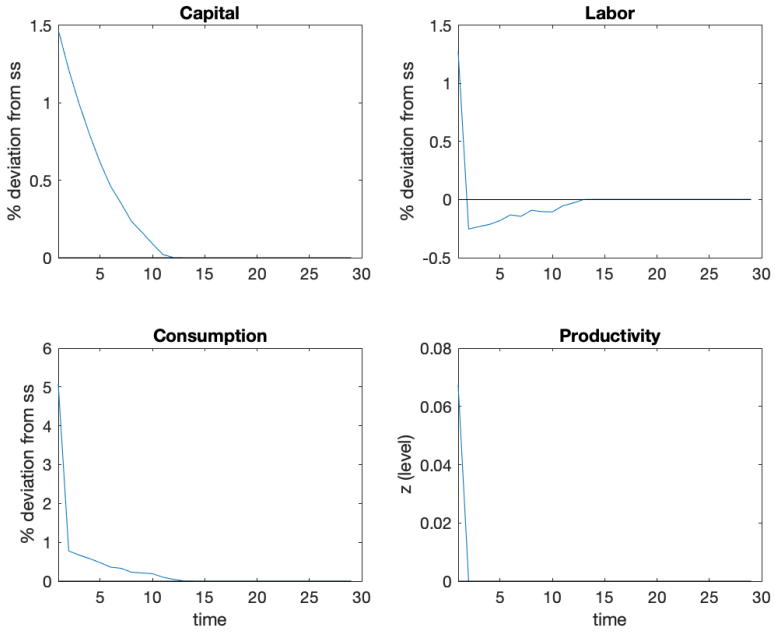

Figure 18: Impulse response functions for a positive shock on z - 5000 grid points

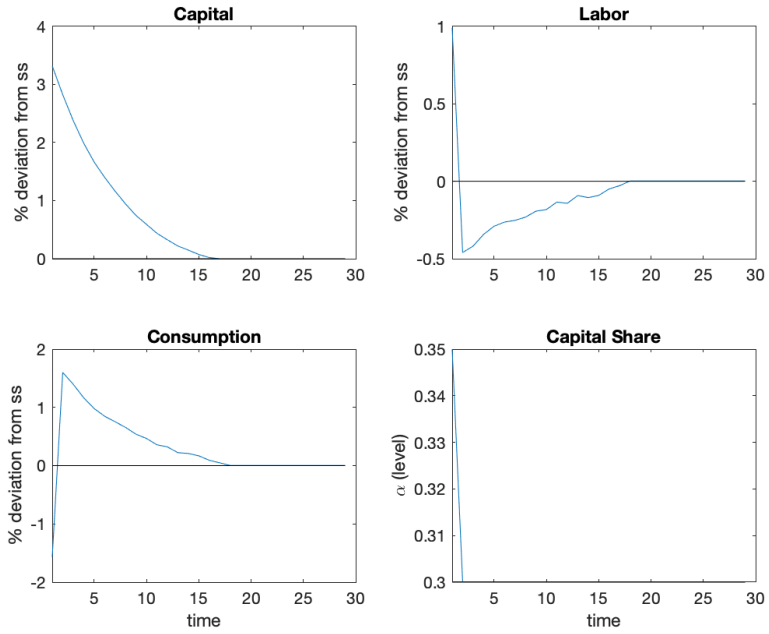

Figure 19: Impulse response functions for a positive shock on α - 5000 grid points

A.3 Plots for the Projection Method

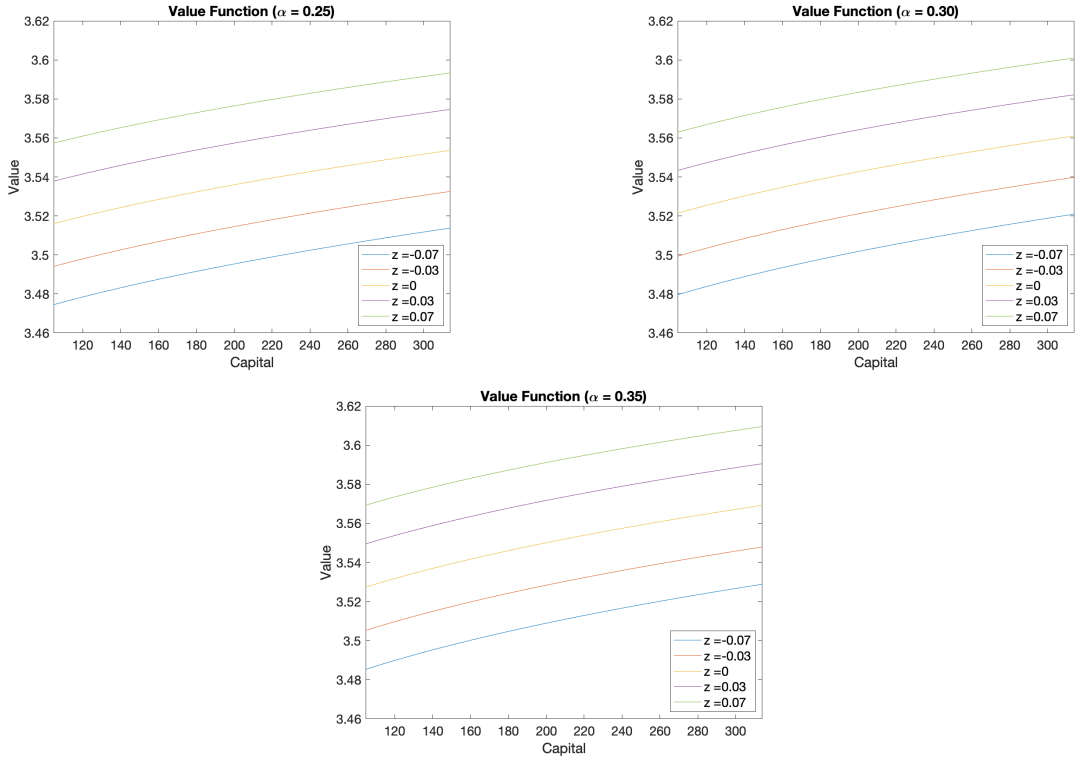

Figure 20: Estimated value functions with $\pm 50\%$ of k_{ss} - projection method

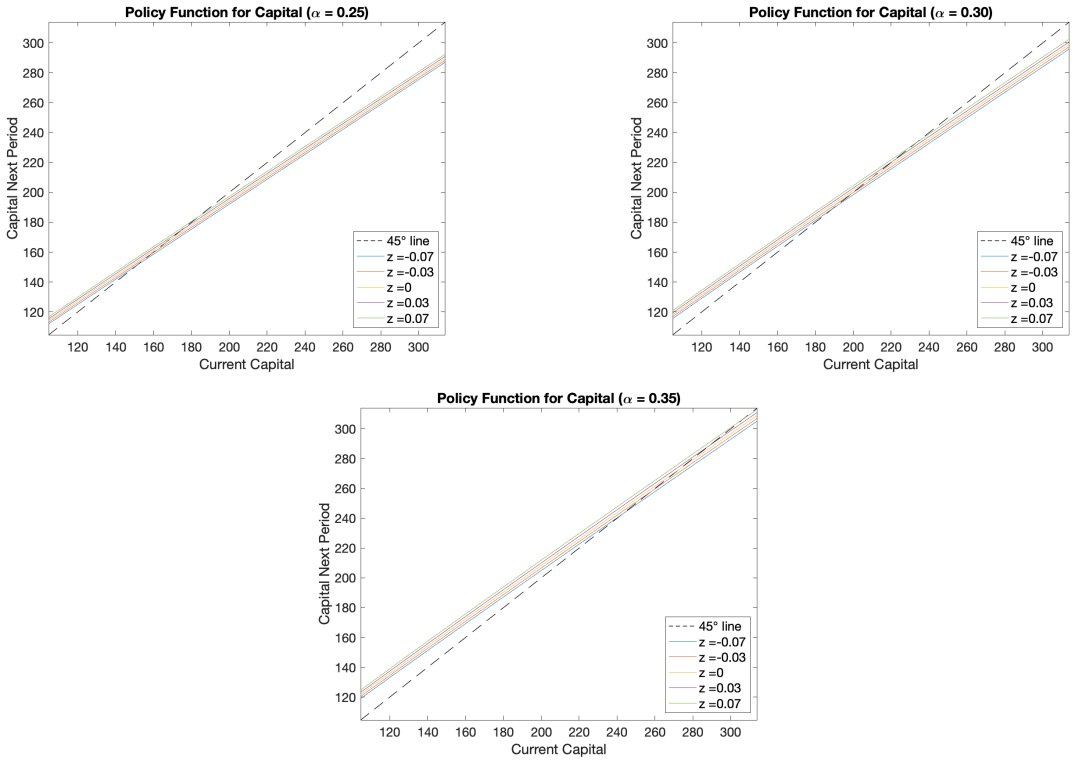

Figure 21: Estimated policy functions for capital next period with ±50% of k_{ss} - projection method

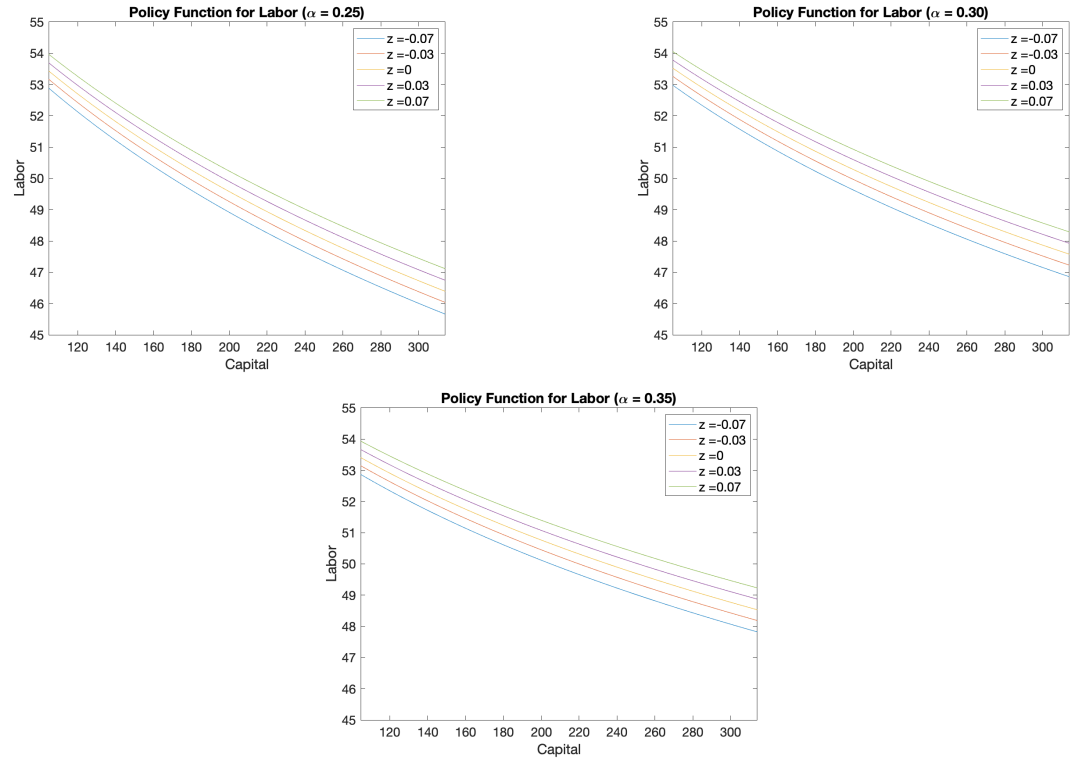

Figure 22: Estimated policy functions for labor with ±50% of k_{ss} - projection method

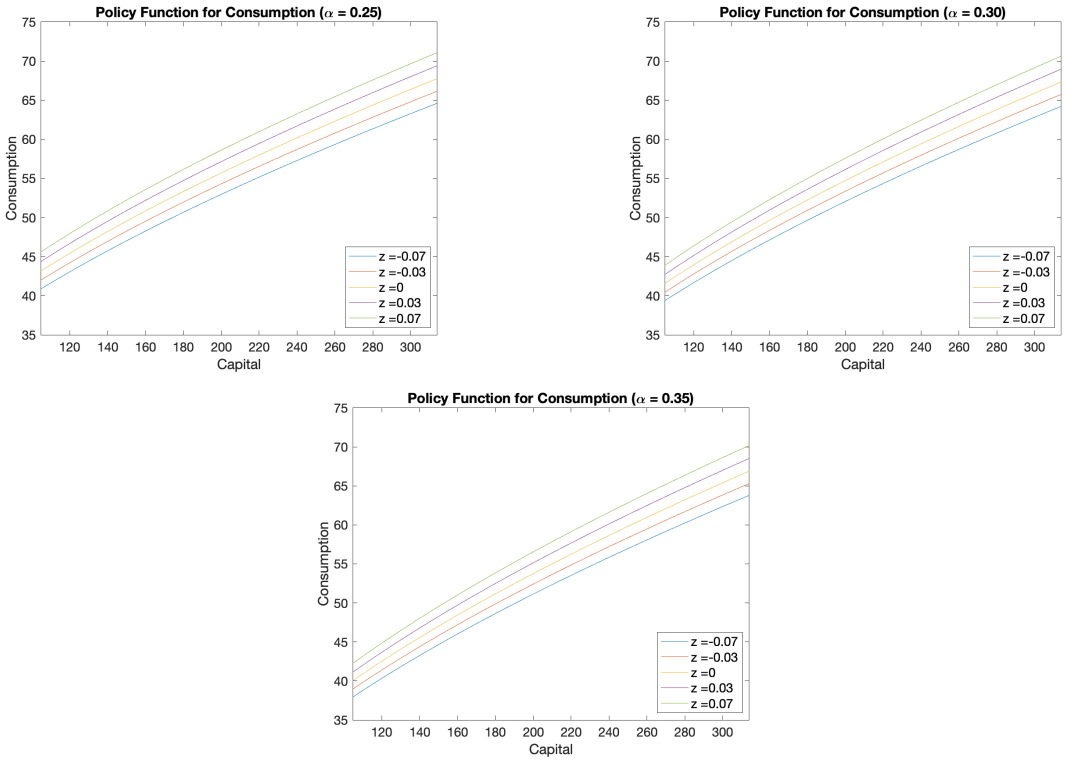

Figure 23: Estimated policy functions for consumption with ±50% of k_{ss} - projection method