ECON 714 Homework 2

Sara Casella

December 10, 2018

1 Introduction

The structure of the document is as follows. In sections 2 and 3 I define and characterize the social planner problem and the deterministic steady state. Then I dedicate a section to each solution method, and for all of them I present derivations if necessary, comments on the algorithm and code implementation, and results. Finally I compare all the different schemes in a separate **comparison** section at the end.

2 Social Planner

First welfare theorem holds in this economy, so we can solve the associated social planner problem. In recursive form:

$$\begin{aligned} V(k,z,\alpha) = \max_{c,l,k'} & \left[\left(\log c - \eta \frac{l^2}{2} \right)^{\theta} + \beta \left(\mathbb{E}(V(k',z',\alpha')^{1-\gamma} \right)^{\frac{\theta}{1-\gamma}} \right]^{\frac{1}{\theta}} \\ \text{s.t.} \quad & c + k' = (1-\delta)k + zk^{\alpha} l^{1-\alpha} \end{aligned} \tag{1}$$

the law of motions for the shocks being defined by the transition matrices.

Characterization

Let $\tilde{V} \equiv \left(\log c - \eta \frac{l^2}{2} \right)^{\theta} + \beta \left(\mathbb{E}(V(k',z',\alpha')^{1-\gamma} \right)^{\frac{\theta}{1-\gamma}}$. First order conditions are:

$$\begin{aligned} [c]: \quad & \tilde{V}^{\frac{1-\theta}{\theta}} \left(\log c - \eta \frac{l^2}{2} \right)^{\theta-1} \frac{1}{c} &=& -\lambda \\ [l]: \quad & -\tilde{V}^{\frac{1-\theta}{\theta}} \left(\log c - \eta \frac{l^2}{2} \right)^{\theta-1} \eta l &=& \lambda(1-\alpha) z k^{\alpha} l^{-\alpha} \\ [k']: \quad & \beta \tilde{V}^{\frac{1-\theta}{\theta}} \mathbb{E} \left(V'^{1-\gamma} \right)^{\frac{\theta+\gamma-1}{1-\gamma}} \mathbb{E} V'^{-\gamma} V_k(k',z',\alpha') &=& -\lambda \end{aligned}$$

Using Benveniste-Scheinkman:

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

Combining them we get the optimality conditions. Define the stochastic discount factor:

$$m' \equiv \beta \frac{c}{c'} \left(\frac{\log(c') - \eta l'^2/2}{\log(c) - \eta l^2/2} \right)^{\theta-1} \left(\frac{\mathbb{E} \left(V'^{1-\gamma} \right)^{\frac{1}{1-\gamma}}}{V'} \right)^{\theta+\gamma-1} \tag{2}$$

then Euler equation and intratemporal consumption/labor equation are:

$$\begin{aligned} 1 &= \mathbb{E}\left(z'\alpha' k'^{\alpha'-1} l'^{1-\alpha'} + 1 - \delta\right) m' \\ c &= \frac{(1-\alpha) z k^{\alpha} l^{-\alpha}}{\eta l} \end{aligned} \tag{3}$$

3 Steady State

I compute the deterministic steady state of the model when $z_{ss} = 1$ and $\alpha_{ss} = 0.3$ by fixing a value for labor and then finding a normalization for η that makes it optimal. This way there is no need to solve a system of non-linear equations. In particular, I pick $l_{ss} = 100$ and then calculate:

$$\begin{aligned} k_{ss} &= \left(\frac{\beta \alpha_{ss} z_{ss} l_{ss}^{1-\alpha_{ss}}}{1 + \beta(\delta - 1)}\right)^{\frac{1}{1-\alpha_{ss}}} \\ y_{ss} &= z_{ss} k_{ss}^{\alpha_{ss}} l_{ss}^{1-\alpha_{ss}} \\ c_{ss} &= y_{ss} - \delta k_{ss} \end{aligned}$$

and finally

$$\eta = \frac{(1-\alpha_{ss}) y_{ss}}{c_{ss} l_{ss}^2}$$

Below I present a table with the steady state values given my normalization (approximated to the second digit).

l_{ss}	k_{ss}	y_{ss}	c_{ss}
100	418.70	153.66	111.79

Table 1: Steady State values for given parameters

The corresponding η is approximately 0.0001.

4 Value Function Iteration With a Fixed Grid

Code comments and speeding up

The algorithm for the value function iteration with interpolation is as follows:

1. I construct a uniform grid for capital with 250 points. I initially centered the grid around k_{ss} with a coverage of $\pm 30\%$ as suggested. However, I noticed that with these values not all of the policy functions for capital crossed the 45 degree line, and in particular the ones for high z and α. This artificially imposes a constraint on the household and can have repercussions for the rest of the policy function. Therefore I extend the upper bound to of $+50\%$ (which is approximately the minimum value needed for all policy functions to cross the 45 degree line).

2. At a given iteration n, I 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

3. For each possible combination of capital, productivity and capital share today, I use MATLAB's solver fminbnd to find the k' that maximizes the value function V_n. Since k' is almost certainly off grid, it is necessary linearly interpolate to find the corresponding expected value of V.

4. 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 10^{-6}:

$$\sup_{z,\alpha,k} |V_n(z,\alpha,k) - V_{n-1}(z,\alpha,k)| < 10^{-6} \tag{4}$$

In order to speed up the computation, I apply some of the suggestions given in the lecture on value function iteration. In particular:

- I normalize the problem by multiplying the period utility function by $(\mathbf{1} - \boldsymbol{\beta})$.
- I initialize the value function iteration using the normalized value of utility in steady state as an **initial guess**.
- I combine the two vectors of states in a single vector. This means that I use a single state variable $s = (z, \alpha)$; with $s_1 = (z_1, \alpha_1)$, $s_2 = (z_1, \alpha_2)$ and so on until $s_{15} = (z_5, \alpha_3)$. Accordingly, I construct a 15×15 transition matrix using independece of the two processes. The elements of this matrix are such that $\pi(s'|s) = \pi(z'|z)\pi(\alpha'|\alpha)$.

Using these tricks it takes around 4 hours and 5 minutes to obtain the value and policy functions.

I also explore a different route to speed up the the computation. In particular, I decided to perform the value function iteration *without* interpolation and use that as an initial guess for the value function iteration with interpolation. In order to quickly calculate the value function without interpolation I use the following procedure:

- I calculate the period utility in advance, outside of the value function iteration. This way the calculation for labor has to be perfomed only once and not every time we enter a new loop. This seemed to me a good idea given that there are only 250 points for capital (it takes around 5 minutes to calculate the $250 \times 250 \times 15$ matrix), but I keep in mind it might not be feasible with more points.
- I exploit the monotonicity of the value function by storing the index of the optimal choice of capital given the states with which I enter the loop and using it as the provisional next step of the policy function for capital.

Using this new trick it takes around 5.8 minutes to obtain the value function (5.1 minutes for calculating the period utility + 1.5 seconds for the initial guess + 35.2 second for the value function iteration with linear interpolation). This illustrates that starting with a very good initial guess can significantly cut down on time.

However, for the purposes of this problem set and for the comparison of all the methods I will keep using the value of utility in steady state as an initial guess.

Results

Figures 1 and 2 show the value functions and policy functions for every point in the state space (k, s). Each color represents a different s for all the points in the grid for k. Figure 3 shows the same value and policy functions broken down by shocks for a mean level of z and a mean level of α, in order to better understand their properties. We can see that the results pass a logic check on the behaviour of the value and policy functions.

The **value function** is clearly increasing in the level of capital and concave, as it should be by theory. Moreover, for a fixed level of the capital share α, it is increasing and concave in productivity z; and for a fixed level of productivity z, it is increasing (and more steepingly so) and concave in α.

The **policy function** for capital is also increasing in capital, in productivity and in the capital share. As can be seen in Figure 1, all of the policy functions cross the 45° line, which reassures me on the fact that the capital grid I chose is wide enough. Labor is decreasing in the level of capital, and its level doesn't change much across shocks and capital levels. Consumption is increasing in capital and in the shocks.

Impulse response functions

I compute the effects of a one time temporary shock in z and a one time temporary shock in α starting from the **ergodic steady state.** Note that the ergodic steady state is slightly different from the deterministic steady state because of precautionary savings motives, so I pay attention to start from the right one. The impulse response functions are portrayed in Figure 4 and Figure 5. All impulse response functions for endogenous variables are reported as percentage deviation from the steady state.

After a positive **one-time temporary shock in productivity** (from the mean level 1 to the max level 1.07), both capital and labor increase by about 1.5% and consumption increases by 5%. As the increase in productivity disappears, the households starts disinvesting and keeps reaping the benefits of a higher capital shock until it goes back to steady state. Note that in period 3 labor undershoots and falls slightly below its steady state level, I assume because of the wealth effect.

After a positive **one-time temporary shock in** α (from the mean level 0.3 to the max level 0.35) the household also wants to increase capital - and labor because of complementarity with capital. To do so, however, it must hurt consumption in period 1, which falls initially by around 1.7%. Once the investment in capital has been made, consumption increases to 1.5%.

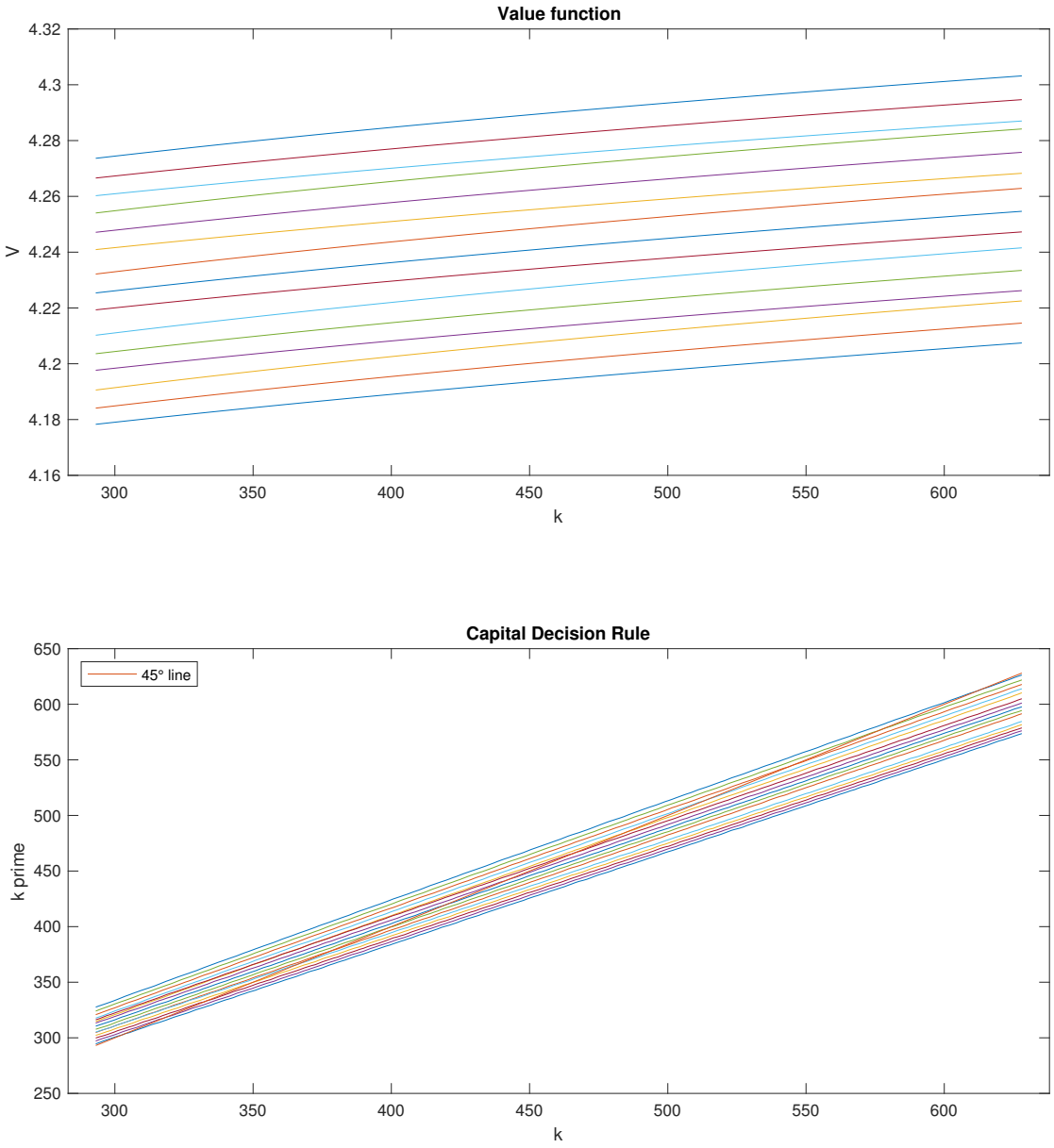


Figure 1: Value and capital policy functions with a fixed grid and interpolation

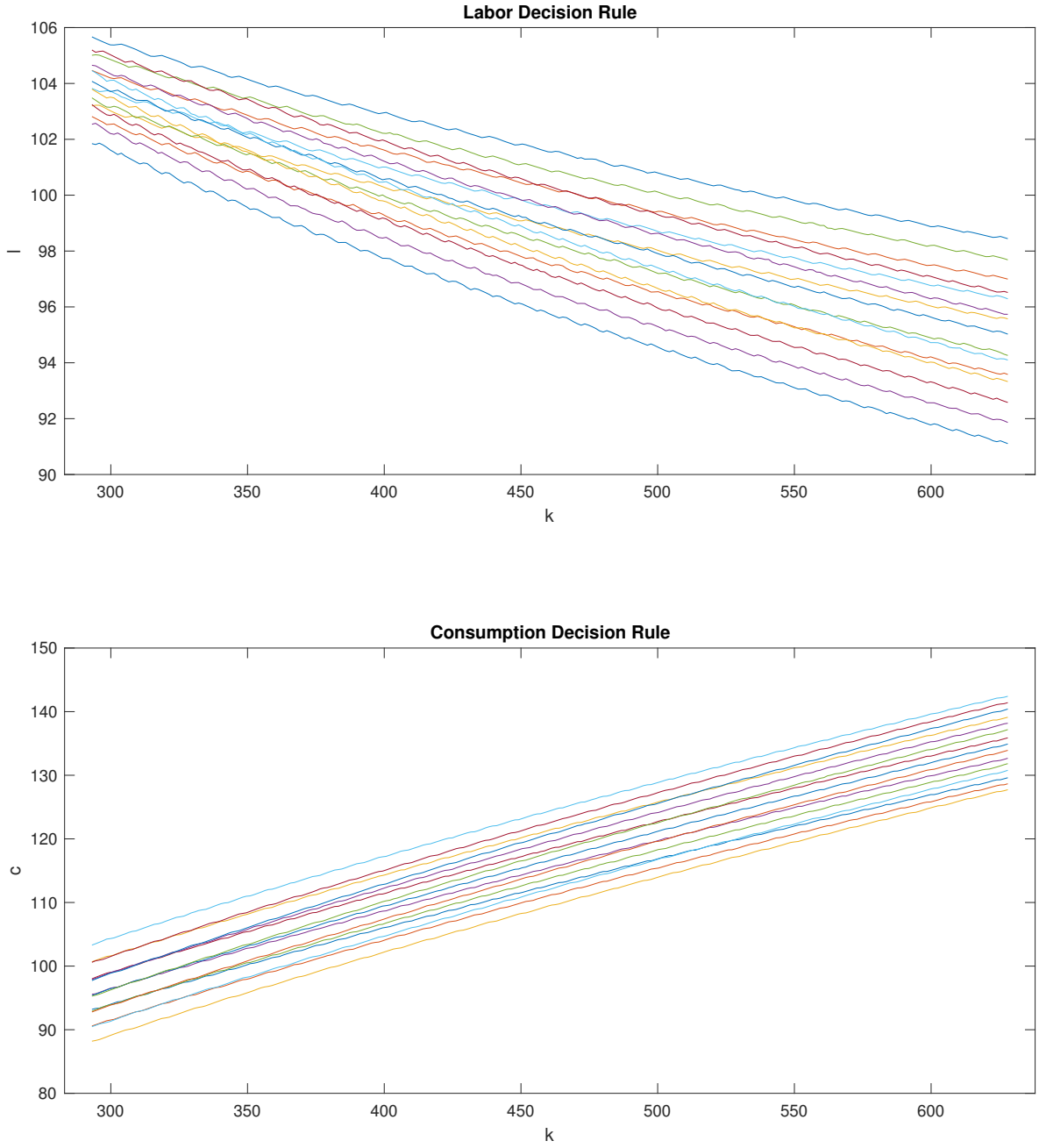


Figure 2: Labor and Consumption policy functions

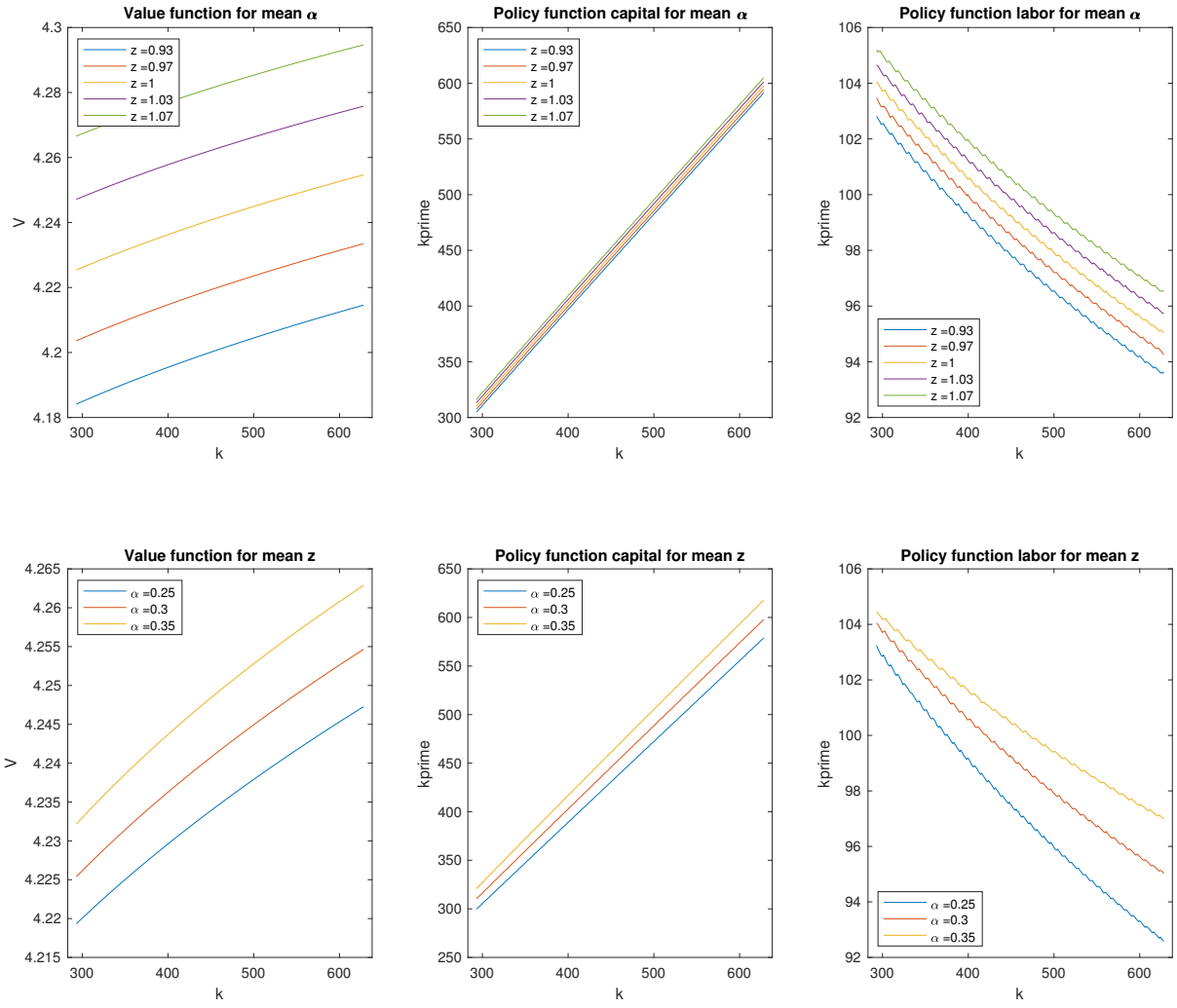


Figure 3: Value and policy functions broken down by shocks

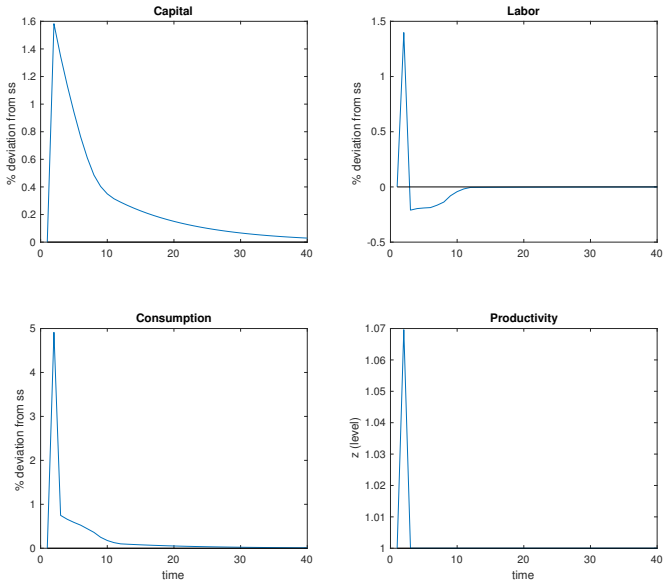


Figure 4: IRF to a shock in z

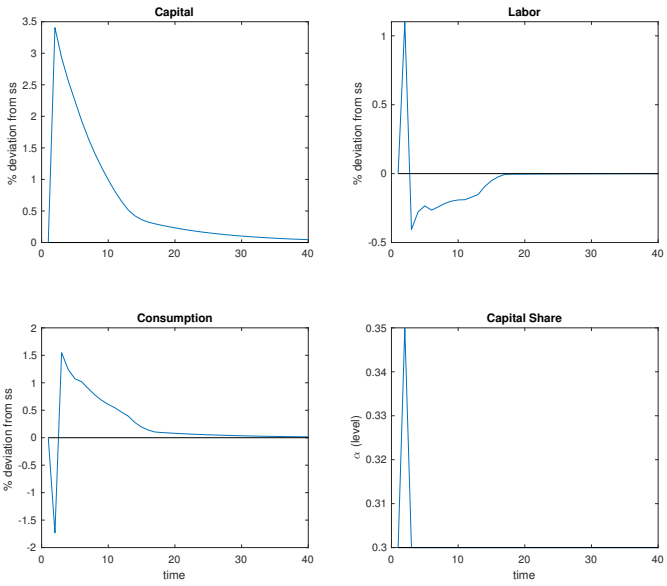


Figure 5: IRF to a shock in α

5 Value Function Iteration With an Endogenous Grid

Derivation

Define "cash-on-hand" as $Y = c + k' = zk^{\alpha}l^{1-\alpha} + (1-\delta)k$. We can write the Bellman equation as

$$\mathbb{V}(Y, z, \alpha) = \max_{l,k'} \left[\left(\log(Y - k') - \eta \frac{l^2}{2} \right)^{\theta} + \beta \left(\mathbb{E}(\mathbb{V}(Y', z', \alpha')^{1-\gamma} \right)^{\frac{\theta}{1-\gamma}} \right]^{\frac{1}{\theta}} \tag{5}$$

If we knew the time-invariant policy function for labor $l(k, z, \alpha)$, we could find Y' and take first order conditions with respect to k':

$$[k'] : \quad -\theta \left(\log(Y - k') - \eta \frac{l^2}{2} \right)^{\theta-1} \frac{1}{Y - k'} + \tilde{V}_k(k', z, \alpha) = 0$$

where I defined $\tilde{V}(k', z, \alpha) = \beta \left(\mathbb{E}(\mathbb{V}(Y', z', \alpha')^{1-\gamma} \right)^{\frac{\theta}{1-\gamma}}$. Then c^* is implicitely defined by

$$\theta \left(\log c^* - \eta \frac{l^2}{2} \right)^{\theta-1} \frac{1}{c^*} = \tilde{V}_k(k', z, \alpha)$$

Code Comments

I follow closely the algorithm for the generalized EGM explained in section 3.2 of Barillas and Fernandez-Villaverde (2006). This is a brief summary of the algorithm:

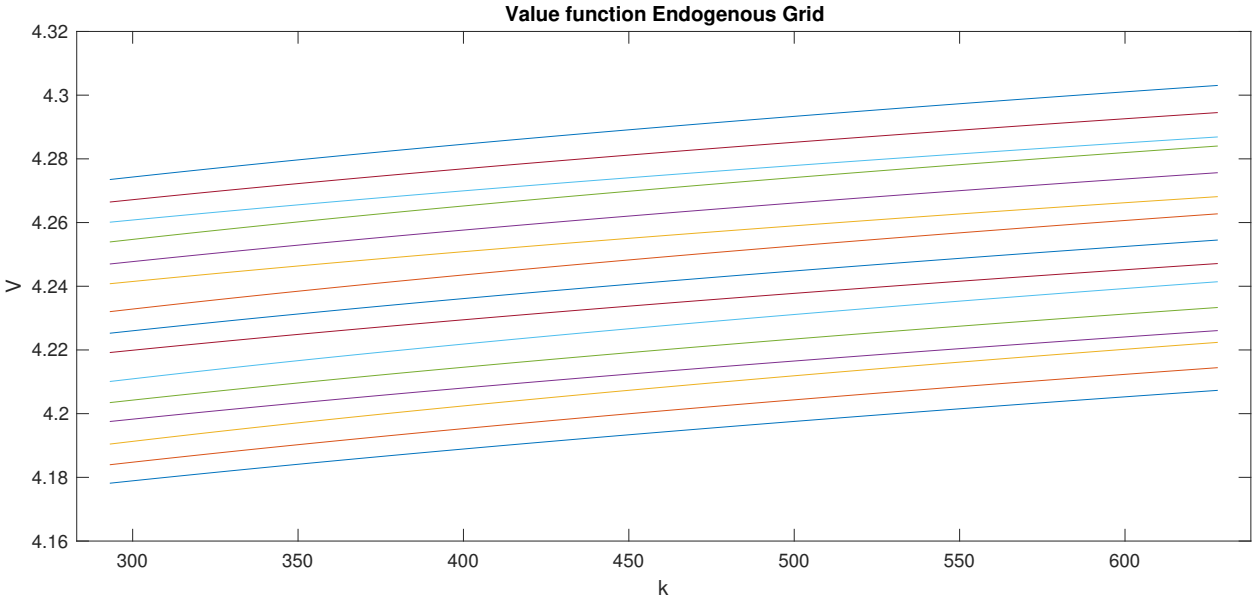
1. I implement the basic endogenous grid method with $l = l_{ss}$ until convergence is achieved.
2. I use the converged value function interpolated on the grid for capital as an input for the standard VFI algorithm and perform two iterations to recover the policy function for labor, capital and consumption. I use the code for the value function iteration with interpolation that I used in the previous questions for this step.
3. I apply EGM using the policy function for labor from step 2 and a tolerance level 10^{-7} . Upon convergence I go back to step 2. I keep iterating between setps 2 and 3 until the sup norm in the functions in step 2 is less than 10^{-6}.

As it is explained in Barillas and Fernandez-Villaverde (2006) the EGM step is reminiscent of the 9 iterations in the accelerator where the policy function is not updated, but the accuracy and speed is superior because the problem is solved exactly.

The endogenous grid method has been the one with the longest coding time. However as I will document in the last section of the document the computing time is significantly cut down.

Results

The results are reported in figures 6 and 7.



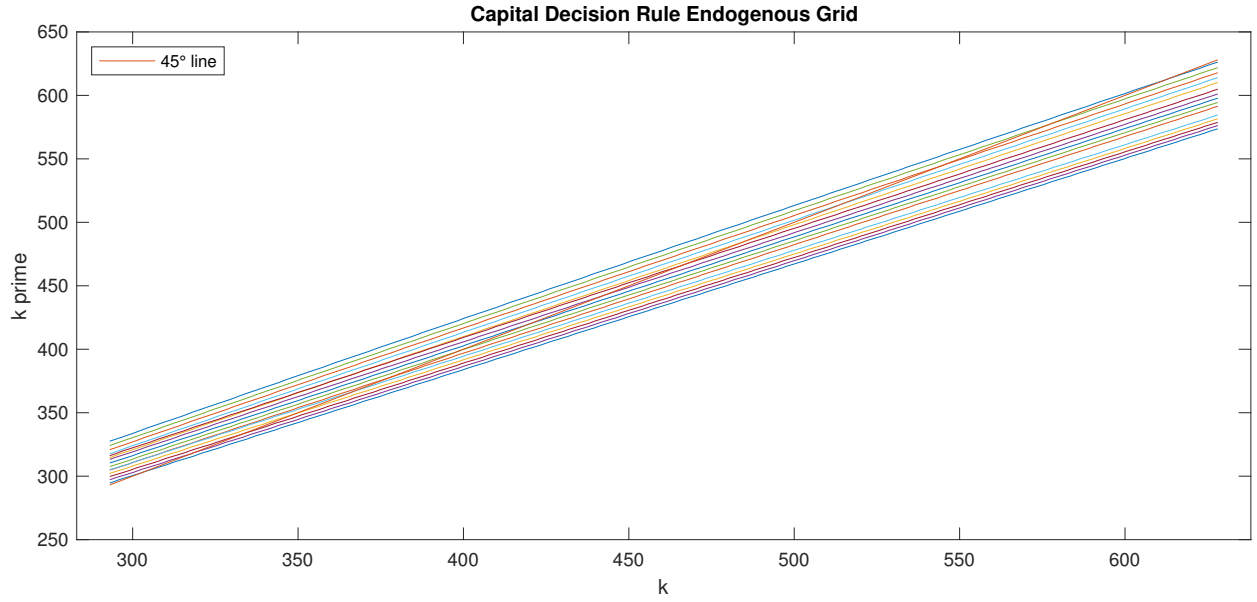


Figure 6: Value and policy function for endogenous grid scheme

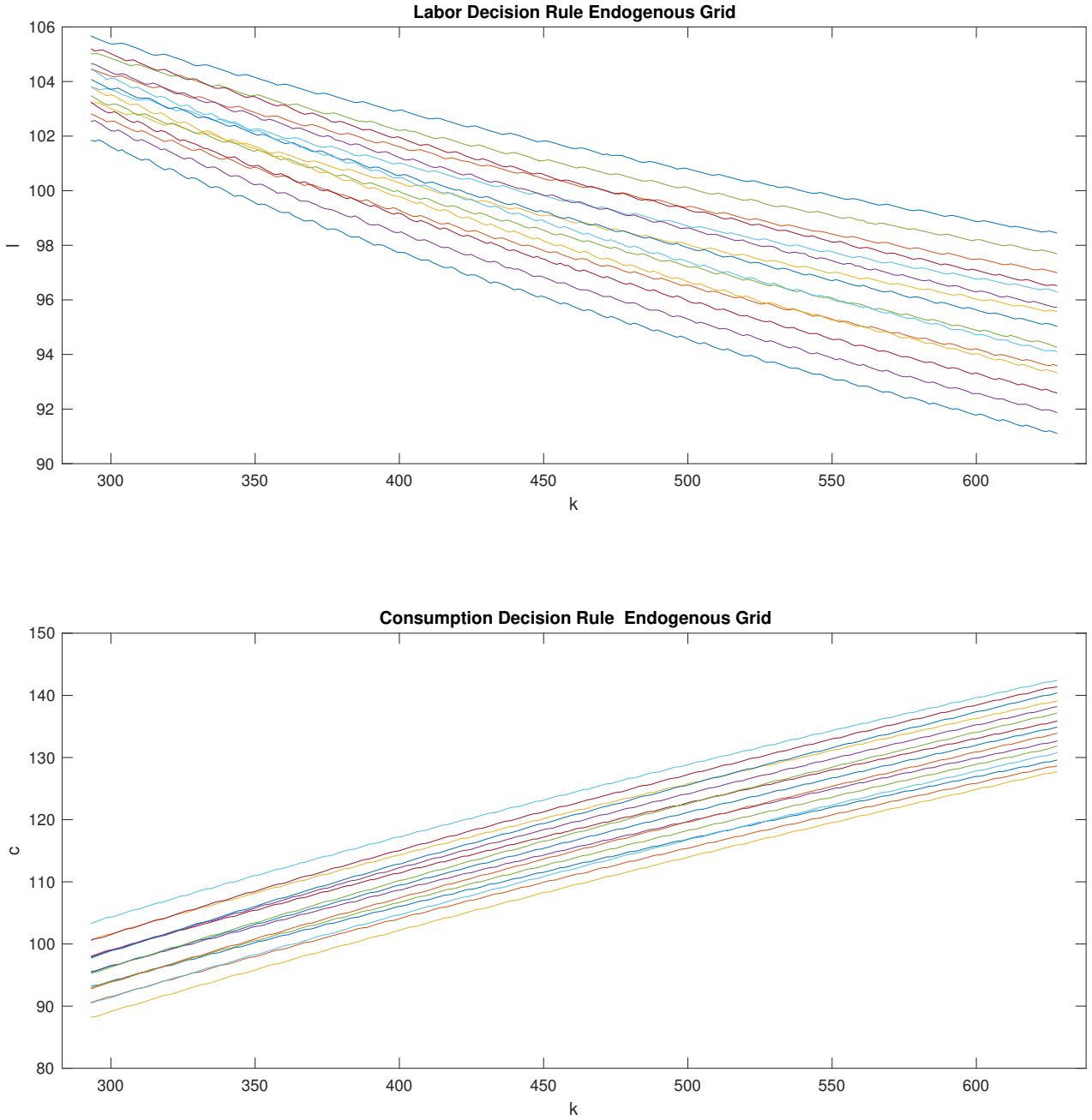


Figure 7: Labor and Consumption policy function for endogenous grid scheme

6 Accelerator

Code comments

The accelerator follows in a straightfoward fashion from the basic code. I insert an "if" statement beore the grid search for capital prime. If the number of iterations is 1 or a multiple of 10 it goes through the process of updating the policy functions. Otherwise, only the value function is updated.

Results

I obtain the same value and policy function as in question 1 and therefore skip including the plots. One interesting observation is that the sup norm doesn't decrease monotonically, but jumps everytime the maximization step is performed.

7 Multigrid

Code comments

I implement a multigrid scheme 100 capital grid points in the first grid, 500 capital grid points in the second, and 5000 capital grid points in the third as instructed.

Each time, I use the final value function approximation of the previous grid as the first guess for the loop on the next grid. To do so I interpolate the approximation derived on the coarser grid along the finer grid; this way the number of points will match.

The code for the value function iteration in each step is recycled from question 1.

Results

In Figure 8 I report the value and policy function obtained. The value function is very similar to the one obtained in question 1, and the policy functions appear smoother. I will formally compare the two methods in the comparison section using the Euler Errors method.

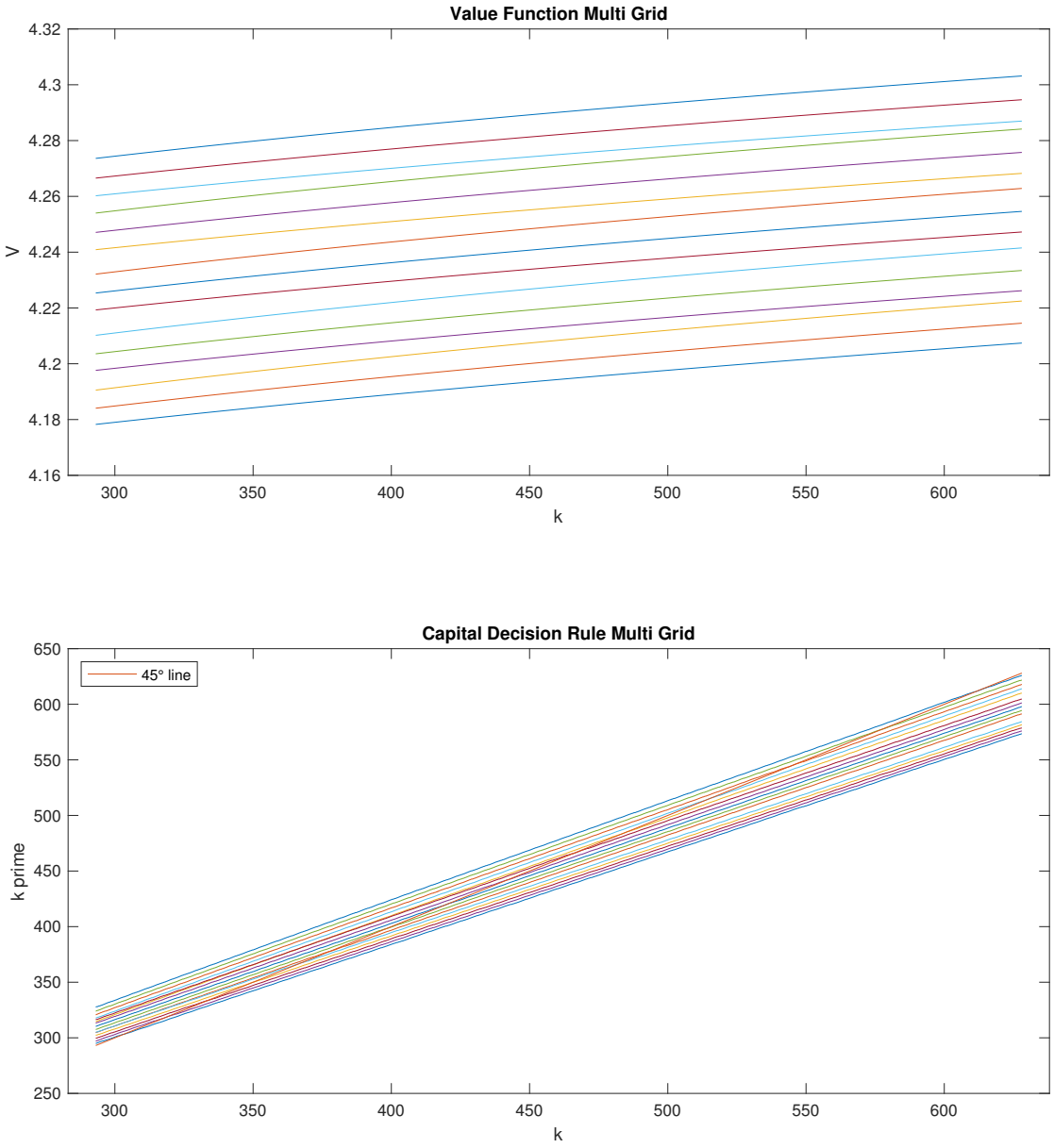


Figure 8: Value and policy function for multigrid scheme

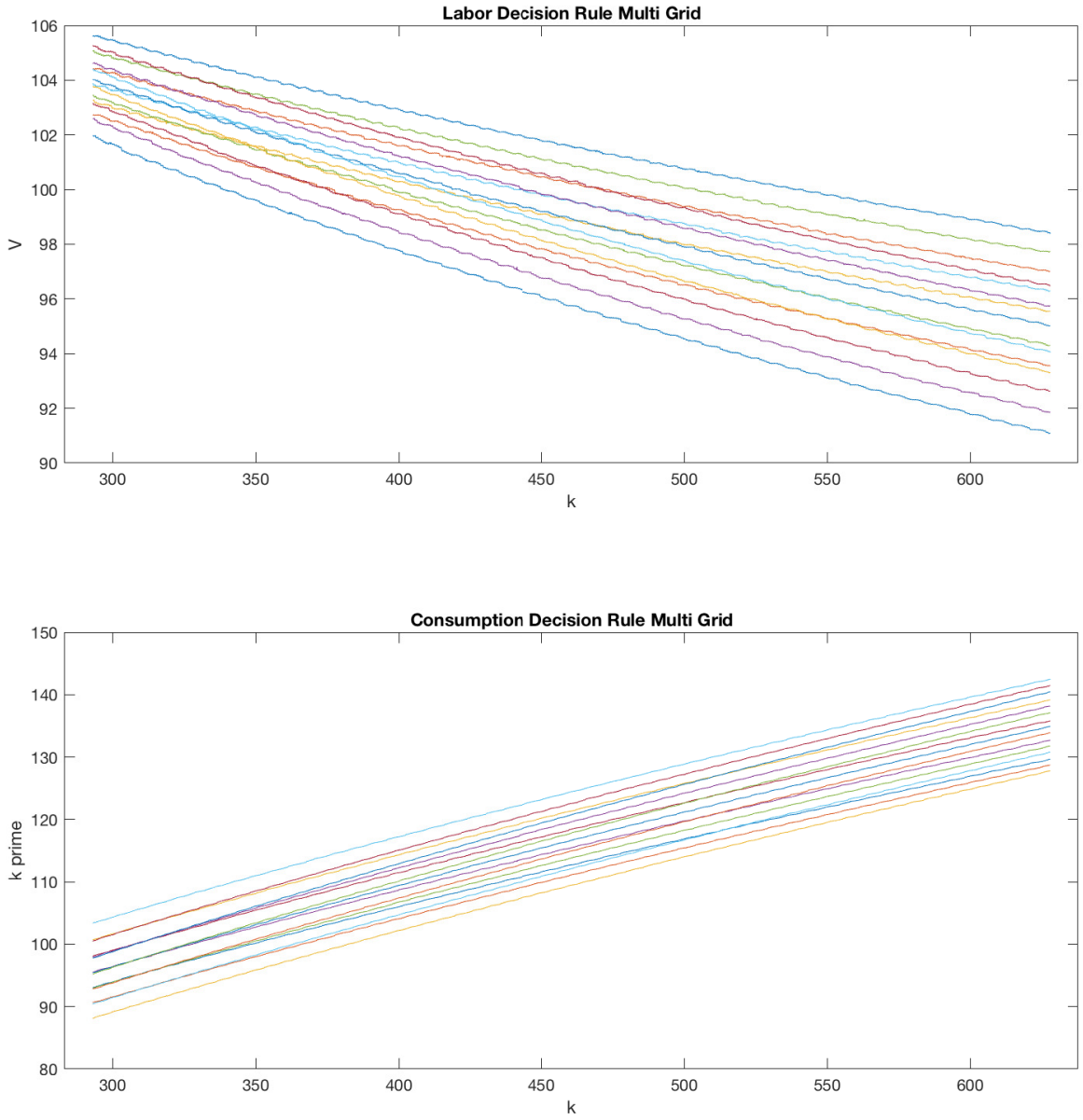


Figure 9: Labor and Consumption policy function for multigrid scheme

8 Chebyshev

Derivation

I will build a residual function $R(k, s, \theta)$ using the Euler equation:

$$1 = \beta \mathbb{E}\left(z'\alpha' k'(k,s)^{\alpha'-1} l(k',s')^{1-\alpha'} + 1 - \delta\right) \frac{c(k,s)}{c'(k',s')} \left(\frac{\log\left(c'(k',s')\right) - \eta l(k',s')^2/2}{\log\left(c(k,s)\right) - \eta l(k,s)^2/2}\right)^{\theta-1} \left(\frac{\mathbb{E}V'^{1-\gamma}}{V'}\right)^{\theta+\gamma-1} \tag{6}$$

and the Bellman equation:

$$V(k,s) = \left[\left(\log c(k,s) - \eta \frac{l(k,s)^2}{2}\right)^{\theta} + \beta\left(\mathbb{E}(V(k',s')^{1-\gamma}\right)^{\frac{\theta}{1-\gamma}}\right]^{\frac{1}{\theta}} \tag{7}$$

that I derived in section 1, where $s = (z, \alpha)$ is the value of my synthetic shock.

I approximate the decision rules for labor as $l(k,s) = \sum_{i=1}^{P} \theta_i^L \psi_i(k)$ and the value function as $V = \sum_{i=1}^{P} \theta_i^V \psi_i(k)$; where $\theta = [\{\theta_i^j\}_{i=1}^{P}\}_{j \in L,V}]$ are unknown coefficients and $\{\psi_i(k,s)\}_{i=1}^{P}$ are Chebychev polynomials up to order P. Note that the fact that the θ parameters depend on s and the ψ polynomials only on k stems froms the discrete nature of the shocks space.

Given the approximation of the policy function for labor $\hat{l}(k,s)$ I find the other policy functions in the following way:

1. $\hat{c}(k,s)$ is found using the static equation

$$c(k,s) = \frac{(1-\alpha) z k^{\alpha} l(k,s)^{-\alpha}}{\eta l(k,s)}$$

2. $\hat{k}'(k,s)$ is found using the budget constraint

$$k'(k,s) = (1-\delta)k + z k^{\alpha} l(k,s)^{1-\alpha} - c(k,s)$$

Then I find the θ^j using collocation. This means that I choose a number P of points equal to the number of polynomials in the capital space $[0.7k_{ss}, 1.5k_{ss}]$ where equations 6 and 7 must hold exactly for each state s, and then solve the system of $2 \times P \times 15$ equations $R(k_i, s_i, \theta)$ in $2 \times P \times 15$ unknows θ using Newton's method (Matlab's fsolve).

Code Comments

I write the code for the projection using Chebychev polynomial by adapting the code that was shared with the class on Canvas. I do the computation first solving with polynomials till the 3rd order, and inputing the solution as an guess for the 8th order round.

Results

The resulting value and policy functions are plotted in Figure 10.

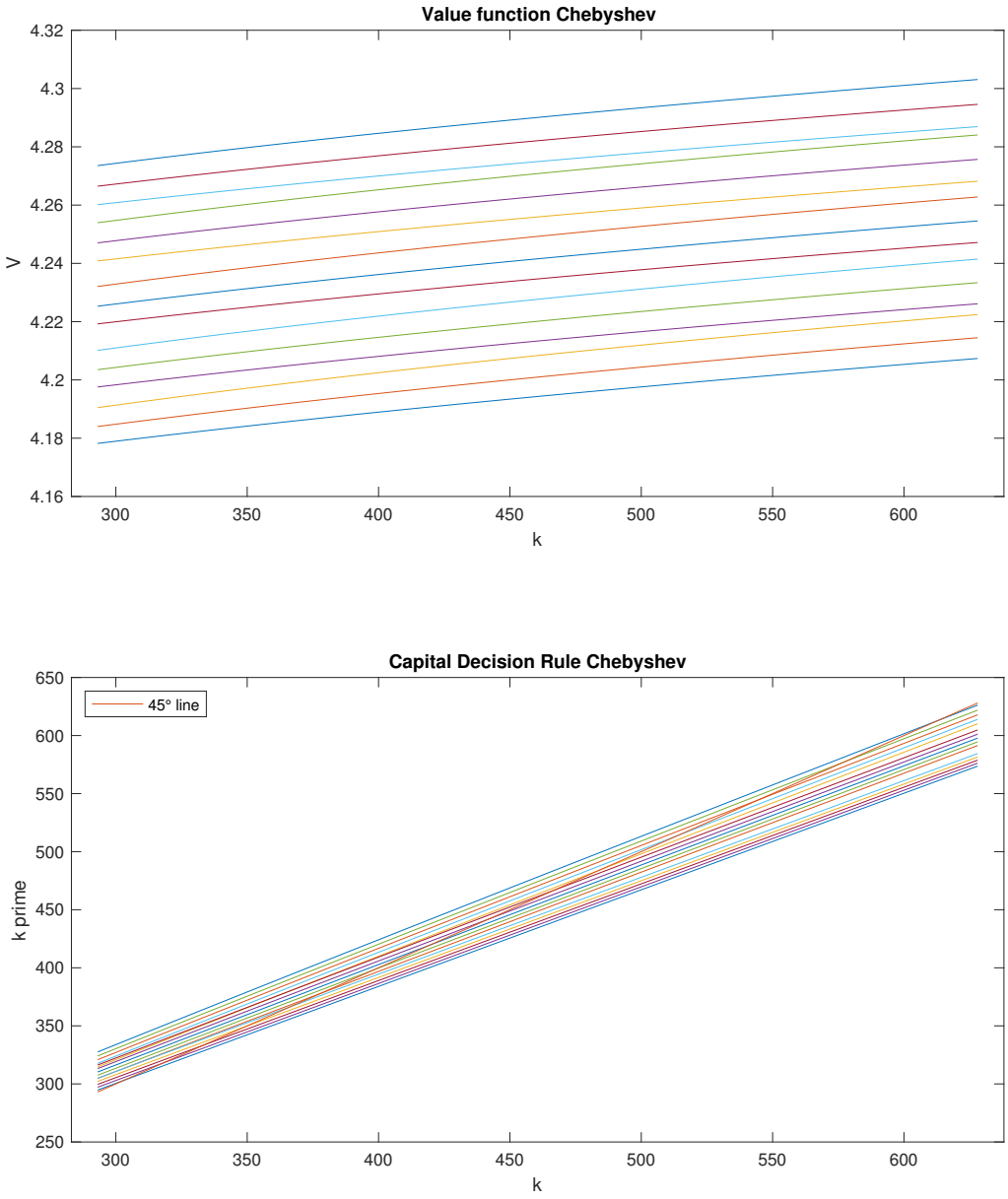


Figure 10: Value and policy functions for Chebyshev projection

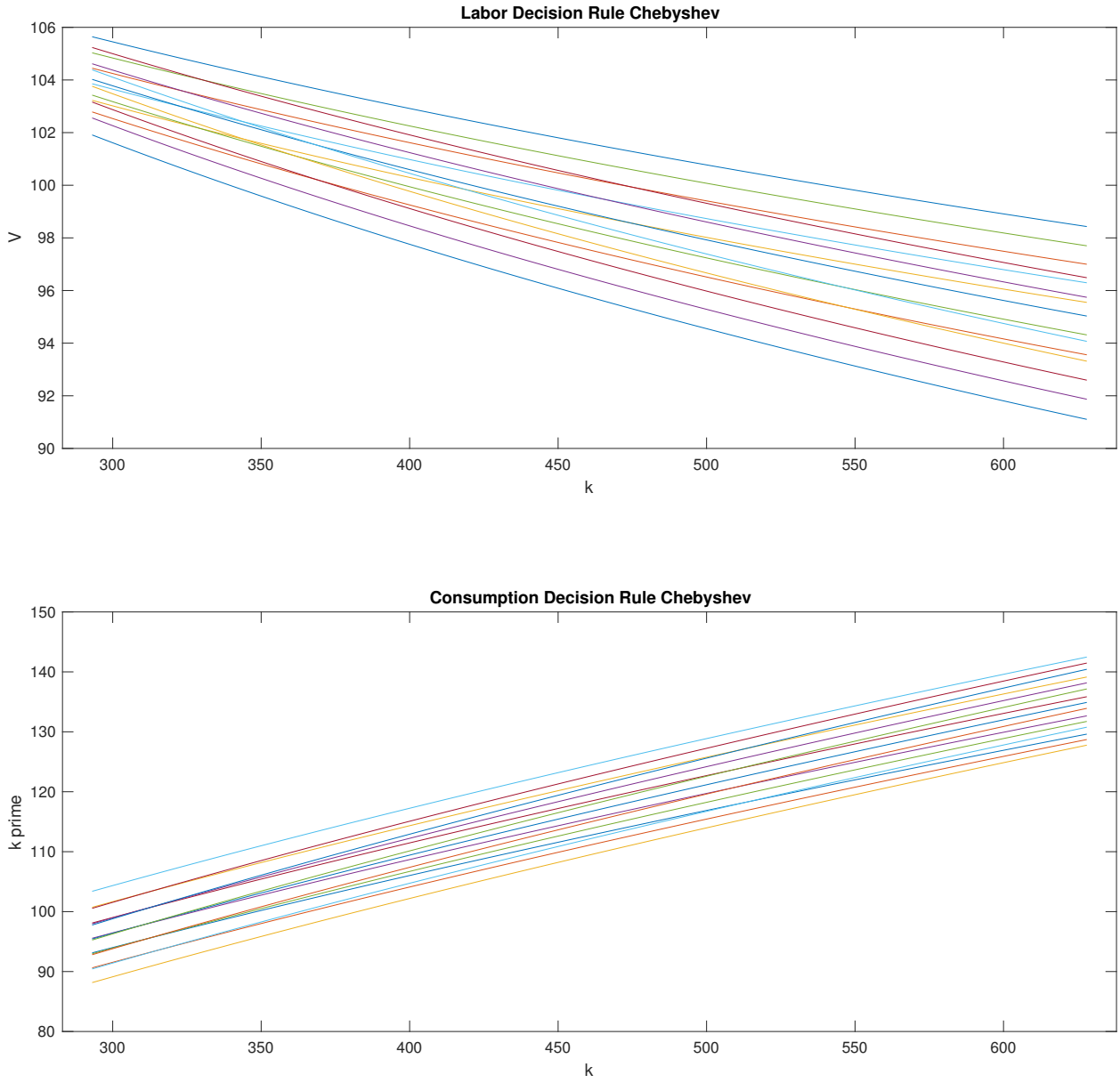


Figure 11: Labor and Consumption policy functions for Chebyshev projection

9 Perturbation

Derivation

The system of equations on which I am going to apply Taylor's expansion around the steady state is given by the optimality conditions found in part 1. I will report them here again:

Number of equations: (7). Number of variables: (7) $[c, l, k, V, m, z, \alpha]$:

$$
\begin{aligned}
[1] \quad V &= \left[\left(\log c - \eta \frac{l^2}{2} \right)^{\theta} + \beta \left(\mathbb{E}(V(k', z', \alpha')^{1-\gamma} \right)^{\frac{\theta}{1-\gamma}} \right]^{\frac{1}{\theta}} \\
[2] \quad c &= \frac{(1-\alpha) z k^{\alpha} l^{-\alpha}}{\eta l} \\
[3] \quad m' &= \beta \frac{c}{c'} \left(\frac{\log(c') - \eta l'^2/2}{\log(c) - \eta l^2/2} \right)^{\theta-1} \left(\frac{\mathbb{E}\left(V'^{1-\gamma} \right)^{\frac{1}{1-\gamma}}}{V'} \right)^{\theta+\gamma-1} \\
[4] \quad 1 &= \mathbb{E}\left(z' \alpha' k'^{\alpha'-1} l'^{1-\alpha'} + 1 - \delta \right) m' \\
[5] \quad c + k' &= (1-\delta) k + z k^{\alpha} l^{1-\alpha}
\end{aligned}
$$

plus the stochastic processes for z and α [6] and [7].

Code comments

I use the software Dynare to compute the third order perturbation. As explained in class, due to the way Dynare calculates expectations I have to include an extra variable: $ev = V(+1)^{1-\gamma}$ which I am going to substitute in for $\mathbb{E}\left(V'^{1-\gamma} \right)$ in the Euler equation above. This way I'm sure Dynare first takes the exponent and then calculates the expectation.

Results

The impulse response functions to a shock in productivity z are reported in Figure 12 and the ones to a shock in the capital share α in Figure 13. The behaviour is similar to the one described in question 1 section 4, with the difference that now we are using a continuous process for z and α so Dynare traces out the deterministic path for z and α. This contributes to both higher amplification and persistence of effects of the shocks

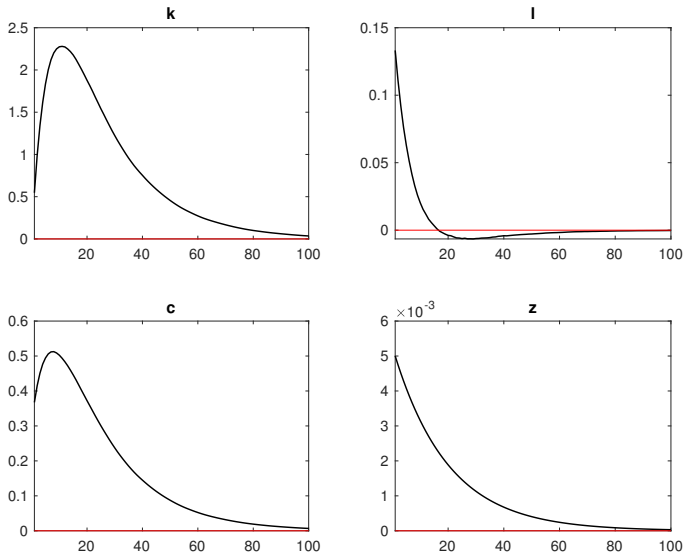


Figure 12: Impulse response to a shock in productivity

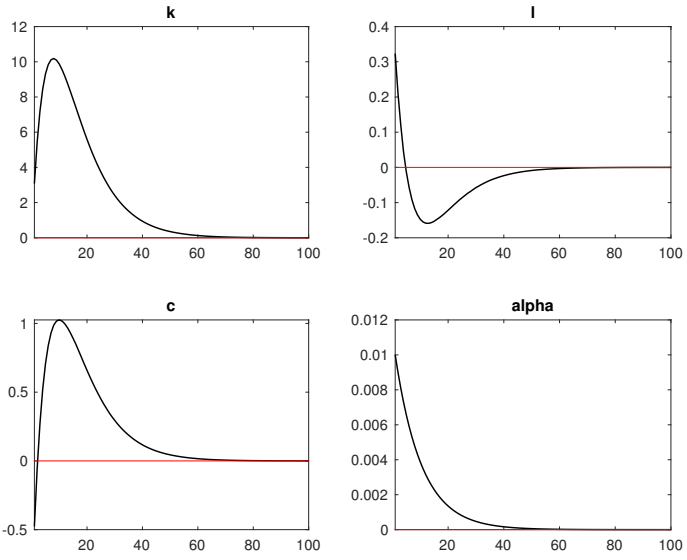


Figure 13: Impulse response to a shock in α

	Computing time	Max Error	Avg Error	Coding time (subjective measure)
Basic	4 hours 5 minutes	-2.5734	-3.1975	Medium
Accelerator	36 minutes	-2.5734	-3.1975	Short
Multigrid	1 hour 50 minutes	-3.5167	-3.5167	Short
Endogenous	23.2 minutes	-2.5670	-3.2036	Long
Chebyshev	49 second	-8.5827	-8.5827	Medium
Perturbation	9 second			Short

Table 2: Comparison across all methods

10 Comparison

I adopt the **Euler Equation Errors** as a measure of accuracy. I define the Euler Equation Errors as

$$EEE(k,s) = 1-\beta\mathbb{E}\left(z'\alpha' k'(k,s)^{\alpha'-1} l(k',s')^{1-\alpha'} + 1 - \delta\right) \frac{c(k,s)}{c'(k',s')} \left(\frac{\log\left(c'(k',s')\right) - \eta l(k',s')^2/2}{\log\left(c(k,s)\right) - \eta l(k,s)^2/2}\right)^{\theta-1} \left(\frac{\mathbb{E}V'^{1-\gamma}}{V'}\right)^{\theta+\gamma-1} \tag{8}$$

and I compute this difference using the approximated policy functions obtained in the various schemes. Having done this, I take the base 10 logarithm of the absolute value and calculate the average and worse errors both across the shocks and the capital grid.

I present a table with **computation time, maximal error and average error; and a subjective measure of how long it took to code.** I also present the plots of the errors across the grid. Errors are more or less constant across the grid for value function iterations. In Chebyshev, the lowest errors correspond to the collocation points.

As we can see, in all value function iteration schemes on average around \$1 is mistaken every \$1000, which is generally bad enough. The Multi Grid has slightly higher precision since the third grid has 5000 points, and takes half the time to run. The accelerator has the same level of accuracy but significantly cuts down running time. The endogenous grid took the longest time to code, but is also the fastest one among all of the value function iteration methods.

With Chebyshev, on average only around \$1 is mistaken every \$100000000. The performance of Chebyshev is overall excellent, it takes very little time to run and it is very precise.

I wasn't able to calculate the Euler Errors for the perturbation. The handbook suggests that for a basic neo-classical growth model the max error would be between -4.5 and -6.5, and the average error across all shocks is lower closer to steady state. In general, coding it has been very easy and the running time is the fastest.

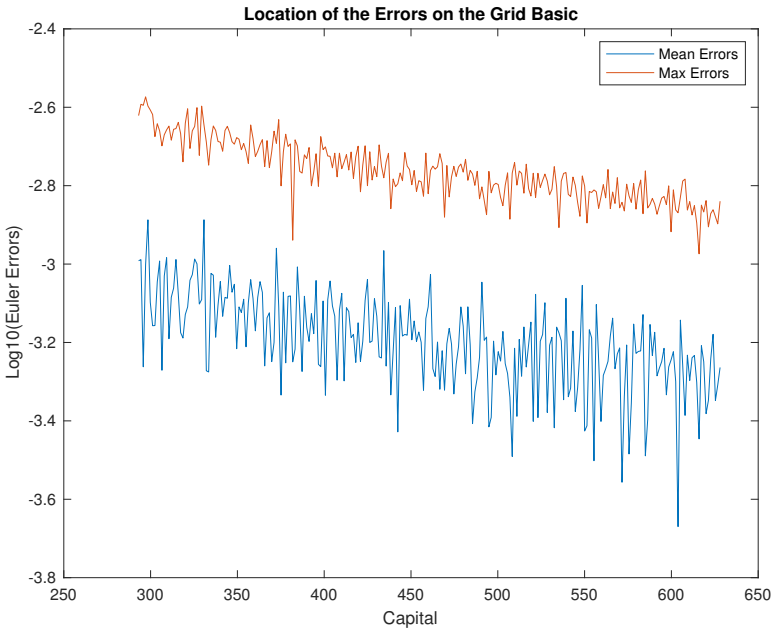


Figure 14: Euler Errors Basic

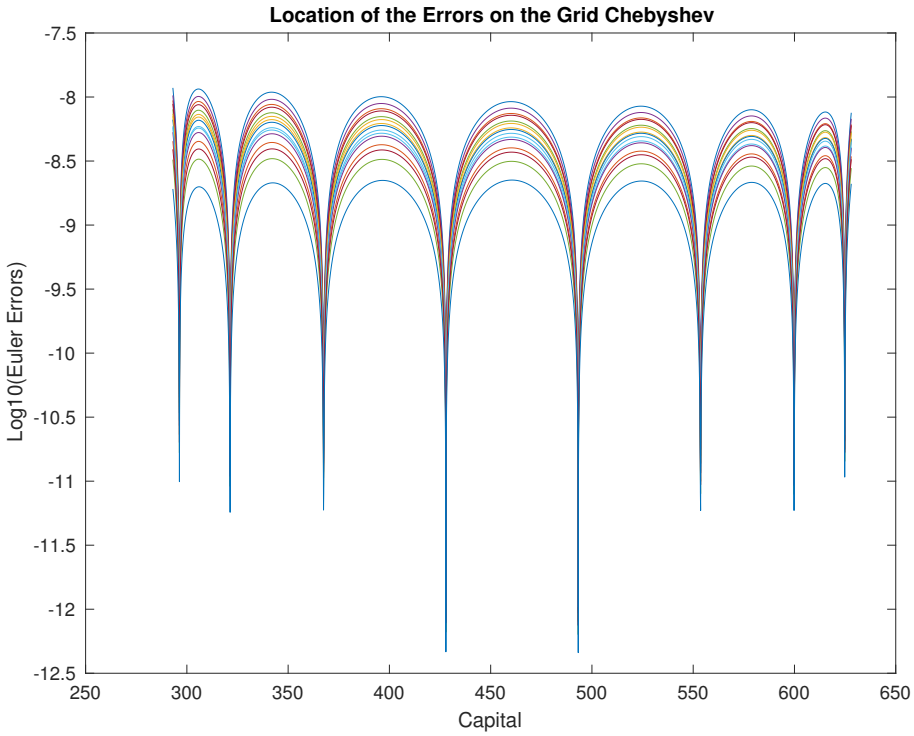


Figure 15: Euler Errors Chebychev

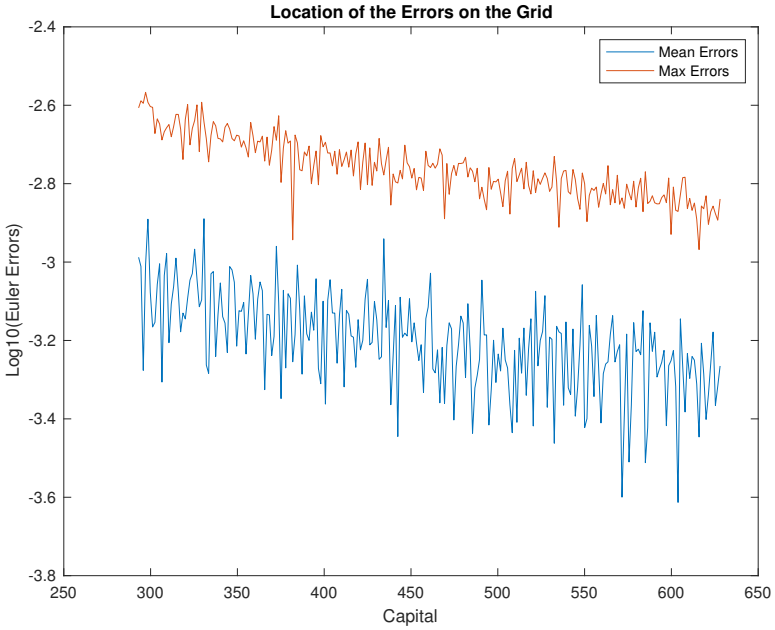


Figure 16: Euler Errors Endogenous Grid

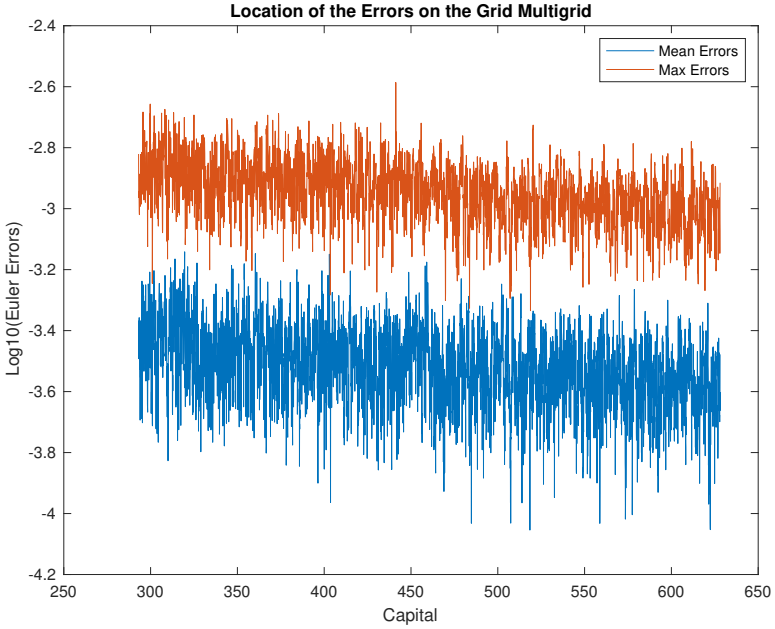


Figure 17: Euler Erros Multigrid