Problem Set III: *Quantitative Macroeconomics*

Jorge Batanero Rodríguez

My laptop characteristics:

- Asus VivoBook S14.
- RAM: 8GB.
- CPU: Intel(R) Core(TM) i5-8265U CPU, 1.60GHz
- Windows office.
- OS: Microsoft Windows 10 v.1903 (compilation 18362.1082).

1 Value Function Iteration

Consider a stationary economy populated by a large number of identical infinitely lived households that maximize:

$$\mathbf{E}_0 \left\{ \sum_{t=0}^{\infty} \beta^t u(c_t, h_t) \right\} \tag{1}$$

over consumption and leisure $u(c_t, h_t) = lnc_t - \kappa \frac{h_t^{1+\frac{1}{\nu}}}{1+\frac{1}{\nu}}$ subject to:

$$c_t + i_t = y_t \tag{2}$$

$$y_t = k_t^{1-\theta} h_t^{\theta} \tag{3}$$

$$i_t = k_{t+1} - (1-\delta)k_t \tag{4}$$

Set $\theta = 0.679$, $\beta = 0.988$, $\delta = 0.13$. Also start with, set $h_t = 1$, that is labor is inelastically supplied. To compute steady state normalize output to one.

Using $h_t = 1$ then $c_t = k_t^{1-\theta} + (1-\delta)k_t - k_{t+1}$, replacing c_t in the utility function we can write the problem in recursive formulation as follows:

$$\begin{aligned} V(k) = \max_{(k')\in\Gamma(k)} \quad & u(k^{1-\theta} + (1-\delta)k - k') + \beta V(k') \\ \text{where} \quad & \Gamma(k) = \{(k') : 0 \leq k' \leq k^{1-\theta} + (1-\delta)k\} \end{aligned}$$

STEP1: the continuous variable k is discretized by defining a grid for the values of k, that is, $K = (k_1, k_2, ..., k_i, ..., k_p)$, with $k_1 = k_{min}$ and $k_p = k_{max}$.

For the k_{min} I should choose a value slightly higher than zero to avoid potential problems of having zero output, and perhaps zero consumption. And for k_{max} I will chose a value 20% higher than the steady state.

To get the capital steady state I solve for the Euler equation in sequential formulation and impose a ss:

$$\max_{k_{t+1}} \sum_{t=0}^{\infty} \beta^t \left(\ln(k_t^{1-\theta} + (1-\delta)k_t - k_{t+1}) - \frac{\kappa}{1+\frac{1}{\nu}} \right)$$

FOC:

$$\frac{\partial}{\partial k_{t+1}} = 0 \iff \frac{\beta^{t+1}((1-\theta)k_{t+1}^{-\theta} + 1 - \delta)}{(k_{t+1}^{1-\theta} + (1-\delta)k_{t+1} - k_{t+2})} = \frac{\beta^t}{(k_t^{1-\theta} + (1-\delta)k_t - k_{t+1})}$$

Imposing ss:

$$\beta((1-\theta)k^{-\theta} + 1 - \delta) = 1 \iff k_{ss} = \left(\frac{1-\theta}{\frac{1}{\beta} - 1 + \delta} \right)^{\frac{1}{\theta}}$$

And now, following the recommendations of the lecture notes we can discretize k in a grid with k_{max} slightly above the steady state.

STEP2: guess a solution $V^{s=0} \in R^p$, say the null vector, that is, $V^{s=0}(k_i) = 0 \ \forall \quad i = \{1, ..., p\}$. S keep track of the number of iterations.

STEP3: Define the return matrix M, where the generic element of matrix M contains the return function:

$$M_{i,j} = m(k_i, k_j) = u(f(k_i) + (1-\delta)k_i - k_j) - \kappa\frac{1}{1+\frac{1}{\nu}}, \quad k_i, j \in K$$

Step4: Impose nonnegativity on consumption, replace by a very negative number any value of the matrix M for which this nonnegativity does not hold.

Step5.1: Given a vector V^s. Compute the matrix $\chi_{ij} = M_{ij} + \beta V_j^s$ Step5.2: Update the value function V^{s+1} as the maximun of each row of matrix χ.

Step6: Check that $||V^{s+1} - V^s < tolerance||$ if that is the case stop, if not repeat.

In the pyhton file the step are explained in more detail.

a) Solve with brute force iterations of the value function. Plot your value function.

Here I followed the steps previously defined.

General comments on the value function that can be applied to any other value function in the problem set, they are all the same for exercise 1, and has the same shape for 2 and 3. I observe that the value function well behaved meaning that it is increasing and concave, also the policy function for capital that is the one I plot in all the exercises, is increasing in capital and monotone.

- Tolerance: $\varepsilon = 0.01$.
- Grid size: 150.
- Time: 11.04 seconds

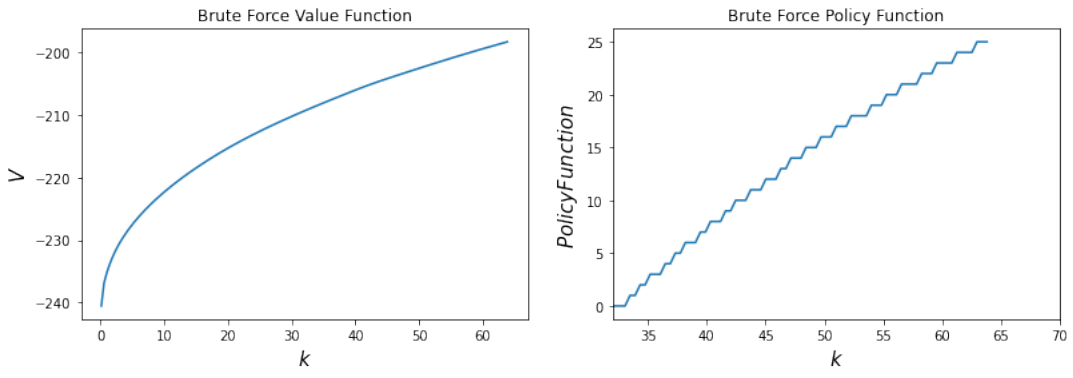

Figure 1: Brute Force

b) Iterations of the value function taking into account monotonicity of the optimal decision rule.

Here I use the property that the optimal decision rule increases in K. Therefore if $k_j > k_i \rightarrow g(k_j) \geq g(k_i)$. Hence if we want to find $g(k_j)$ when $k_j > k_i$, we can rule out searching for all grid values of capital that are smaller than $g(k_i)$.

- Tolerance: $\varepsilon = 0.01$.
- Grid size: 150.
- Time: 9.91 seconds

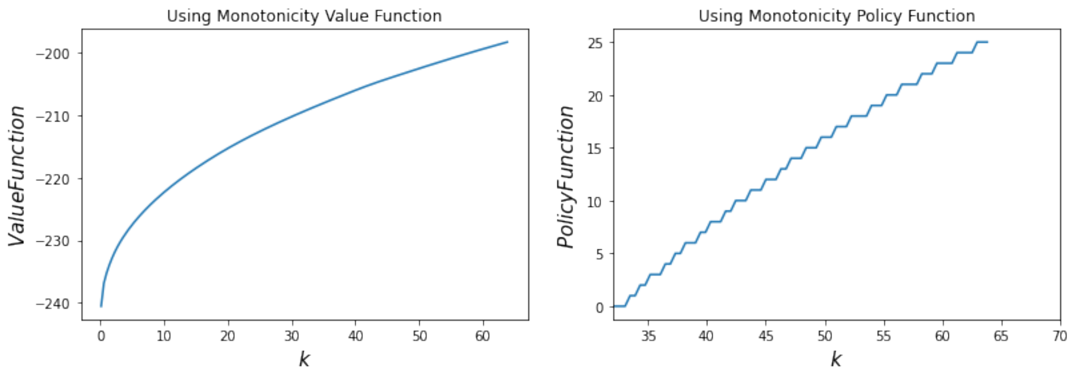

Figure 2: Monotonicity

c) Iterations of the value function taking into account concavity of the value function.

I don't know why my code is not working, I'm introducing the following condition in the code:

$$if \quad M_{i,j} + \beta V_j > M_{i,j+1} + \beta V_{j+1} \rightarrow M_{i,j} + \beta V_j > M_{i,j+2} + \beta V_{j+2}$$

So I'm using the concavity of the value function, but my code is always returning a flat solution with 0 iterations, I've tried in different ways but I always get the same. The same thing happen in part e.

d) Iterations of the value function taking into account local search on the decision rule.

Here I use the fact that the optimal rule is continuous,if we know that $k_j = g(k_i)$, and we have reasonable fine grids, then we hope that $g(k_i + 1)$ is in a small neigborhood of k_j . This restricts the elements of χ that need to be computed. For this part of the code Sergi Quintana helped me.

- Tolerance: $\varepsilon = 0.01$.
- Grid size: 150.
- Time: 3.34 seconds

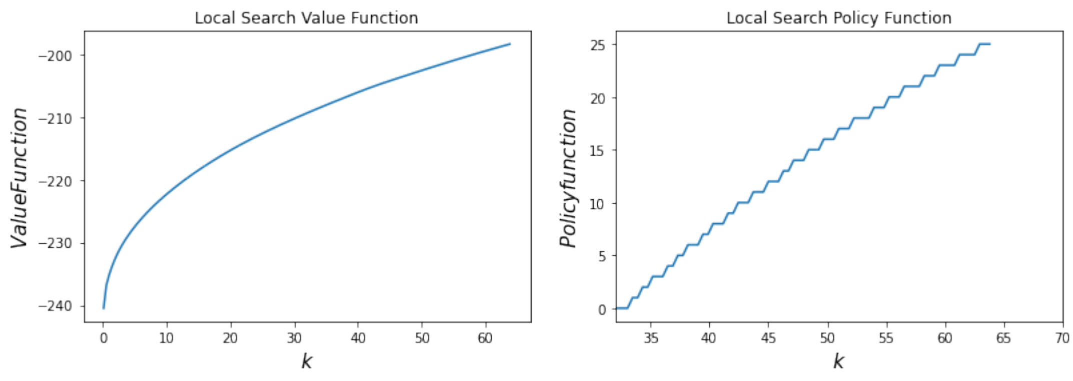

Figure 3: Local Search

f) Use Howard's policy iterations waiting until converged to solve the problem. Start the policy iteration at three different iterations of the value function, and report the differences.

g) Use policy iterations with 5, 10, 20 and 50 steps in between policy reassessments.

I only plotted the Howard value function and policy function for the 50 steps, because it is exactly the same for the others steps, the other plots can be found in the folder of the github.

- Tolerance: $\varepsilon = 0.01$.
- Grid size: 150.
- Time: Between 11-12 seconds depending on the steps

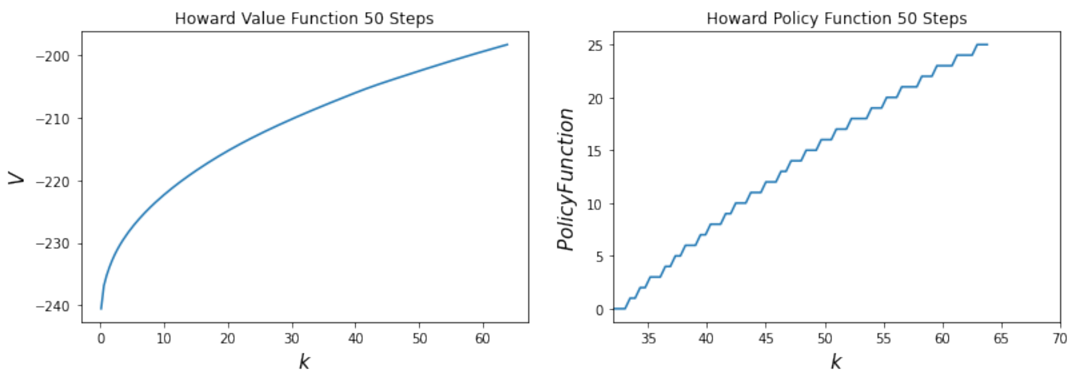

Figure 4: Howard 50 steps

2. Redo item 1 adding a labor choice that is continuous.

- Tolerance: $\varepsilon = 0.01$.
- Grid size: 150.
- Time: 126.91 seconds

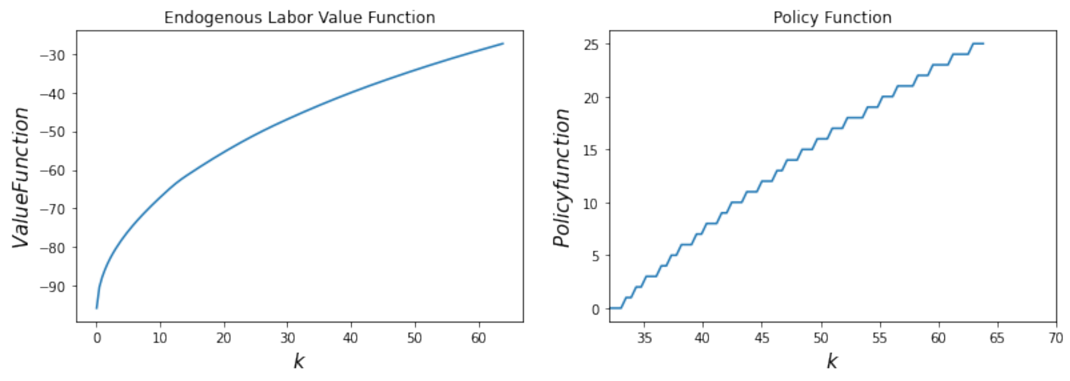

Figure 5: Endogenous Labor

Here I follow the same reasoning as in exercise 1B, but now my code gets slower when using monotonicity, I could not figure out what is happening.

- Tolerance: $\varepsilon = 0.01$.
- Grid size: 150.
- Time: 149.17 seconds.

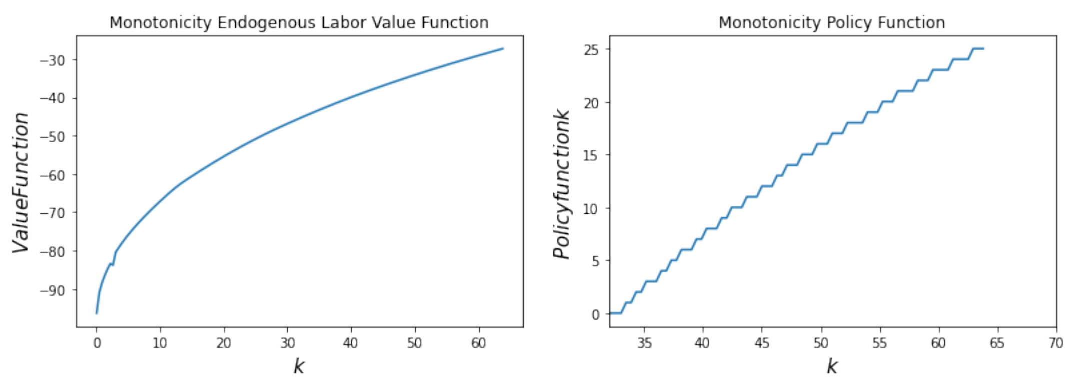

Figure 6: Endogenous Labor Using Monotonicity

3. Redo item 1 using a Chebyshev regression algorithm

To solve this exercise I followed, Makoto Nakajima's notes (Note on Neoclassical Growth Model: Value Function Iteration + Chebyshev Regression, from 2006). My code takes much longer than in the previous methods, so I must have done something wrong. Also the values that the value function takes are much different than the values of the other value functions that I computed in exercise 1, however the form is very similar.

- Tolerance: $\varepsilon = 0.01$.
- Grid size: 150.
- Time: 19.82 seconds
- Polynomial of order 2.

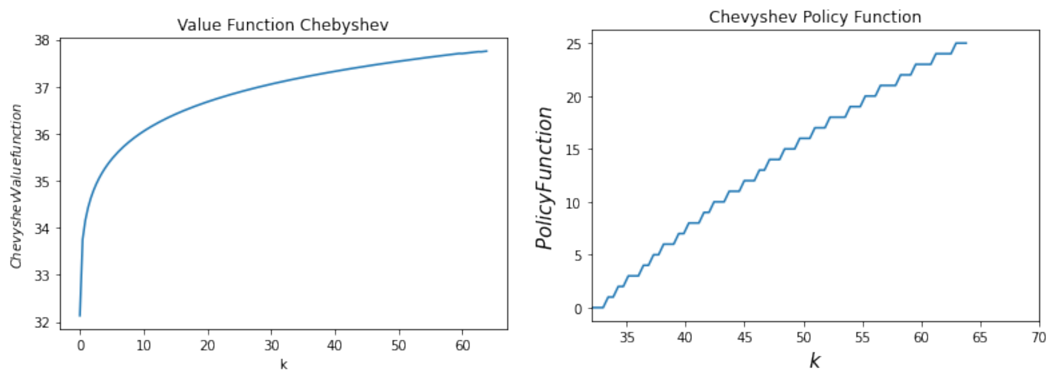

Figure 7: Chevyshev Approximation