Quantitative Macro – Homework 4

Konstantin Boss

November 9, 2019

Value Function Iteration

In each of the exercises the starting guess for the value function is $V_0 = 0$. Moreover, It turns out that the bondary condtition for the approximated policy function(s) is always fulfilled. I write the elapsed time for the first run, it may be that consecutive runs are faster as Python(Atom) saves some information in the cache.

Part a: Brute Force

The brute force method approximates the value function in 81.78 seconds on the first run. It takes 435 iterations for the criterion of $\epsilon < 0.005$ to be met, where ϵ is the difference between the old and the updated value function.

Part b: Exploiting monotonicity of policy

Here we make use of the monotonicity of the policy function alone. It takes again 435 iterations but only 22.14 seconds to converge on the first run.

Part c: Exploiting concavity of the value function

Here we make use of the concavity of the value function alone. It takes again 435 iterations but only 62.95 seconds to converge on the first run.

Part d: Local search

Unfortunately, I have not gotten to this yet.

Part e: Exploiting monotonicity of policy and concavity of value function

Here we make use of the properties of monotonicity of the policy function for capital and the concavity which the value function inherits from the utility function. It leads to 435

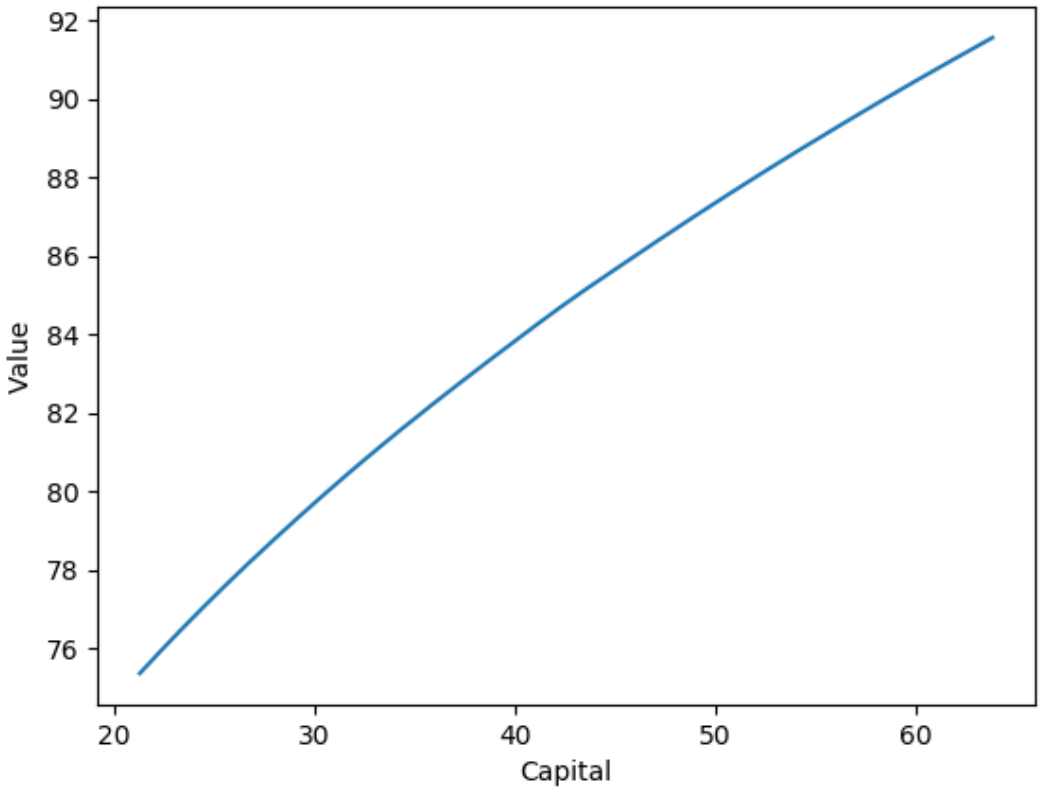

Figure 1: Value function approximation using brute force method.

iterations which elapse in 32.43 seconds on the first run.

Parts f and g: Policy Iteration

I have some code, which does not run well yet, so I refrain from discussing it.

Endogenous Labor Choice

In this section we endogenize labor supply which means that we get another dimension added to our iteration problem. I base my code on the one we saw in the TA session using the Fixed Point Calculator from the quantitative macroeconomics course of Sargent. I set the maximum iteration limit to 2000 to avoid this taking too long. The tolerance for this is $\epsilon = 0.001$. Indeed, it takes 1056 iterations and 395.32 seconds for this to reach a fixed point.

We can see that the values are negative which I assume is either due to false program-

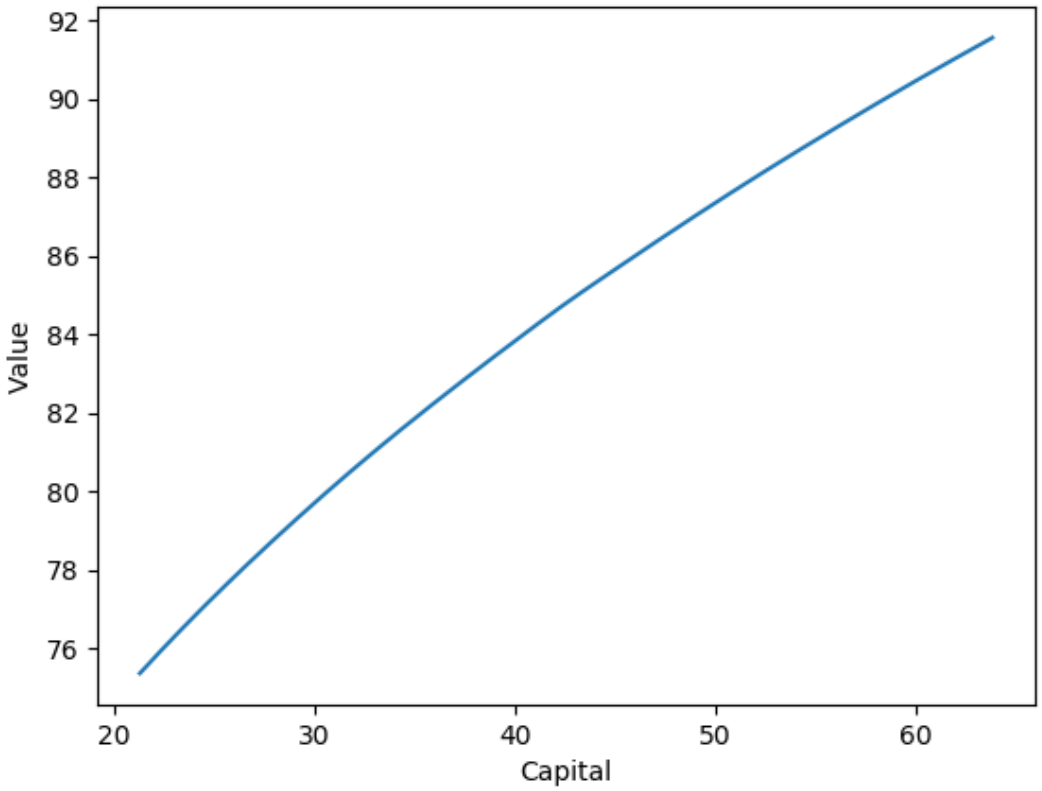

Figure 2: Value function approximation exploiting monotonicity of policy.

ming or due to high disutility of labor as the κ-parameter is rather high. The overall shape of the value function remains the same, so I reckon that the algorithm works but may have some grid choice problems.

Continuous method: Chebyshev regression

I wrote some code for this using my own functions from a previous problem set. I take only 50 nodes. It appears to run, but takes a very long time (533.95 seconds) and produces a bad policy function. This still needs fixing.

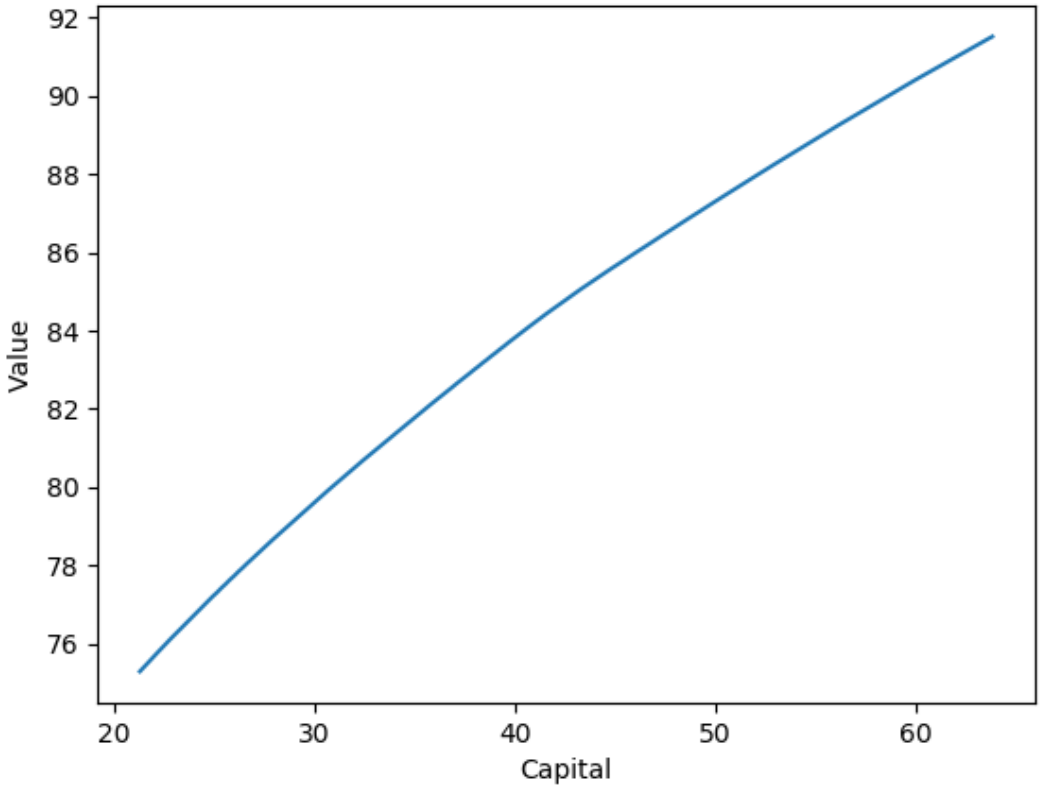

Figure 3: Value function approximation exploiting concavity of the value function

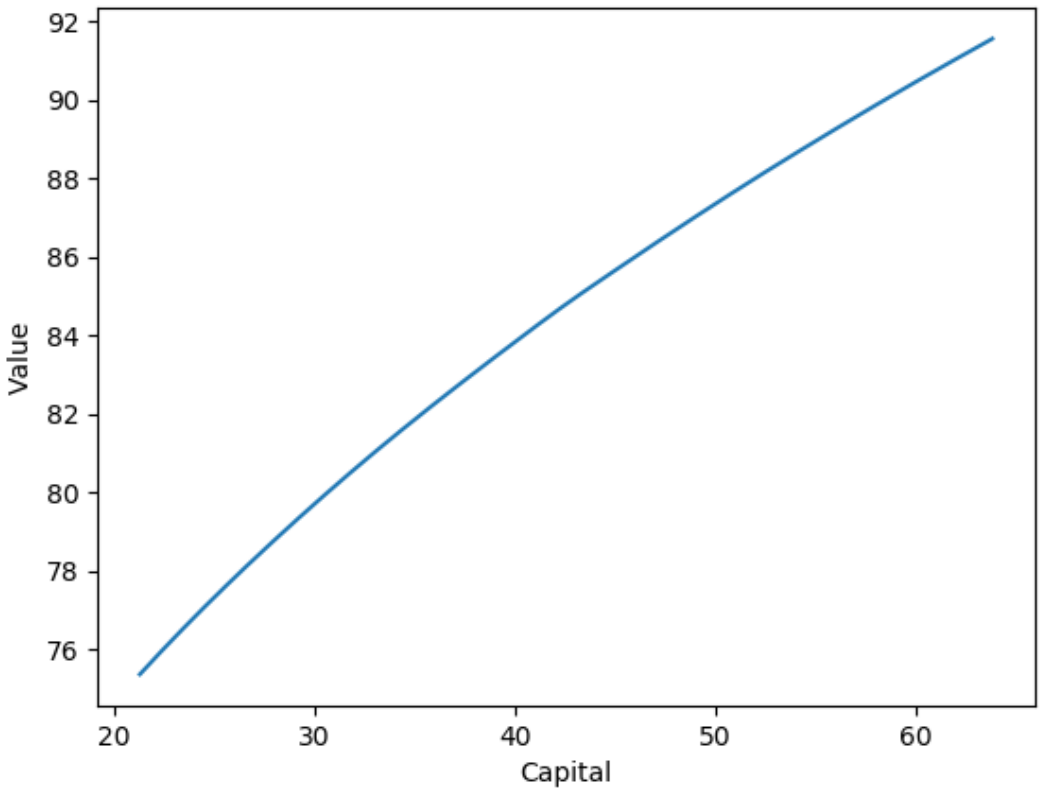

Figure 4: Value function approximation exploiting monotonicity of policy and concavity of value function.

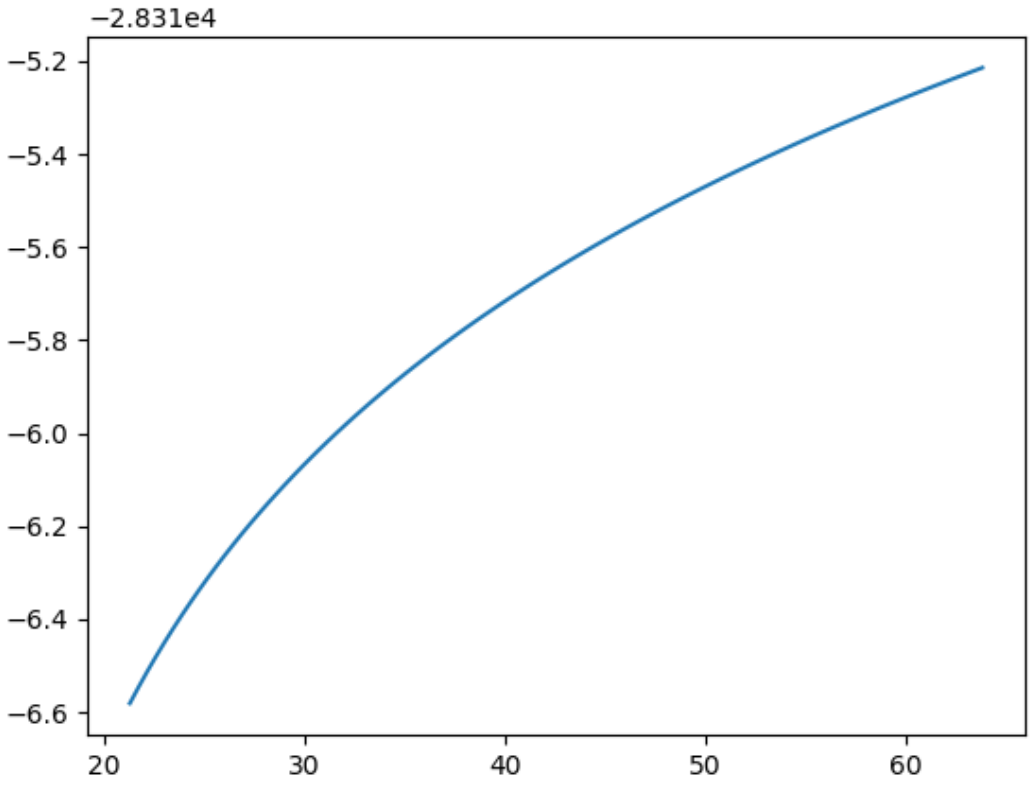

Figure 5: Value function approximation exploiting monotonicity of policy and concavity of value function.

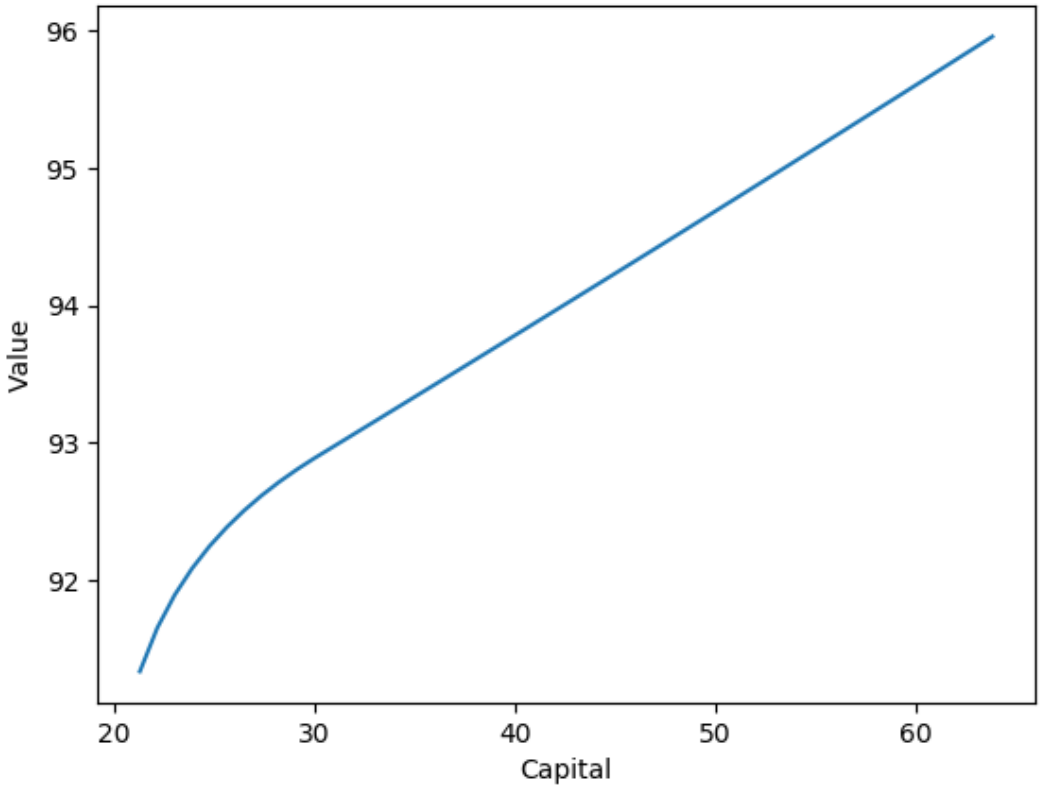

Figure 6: Value function approximation using Chebyshev regression.