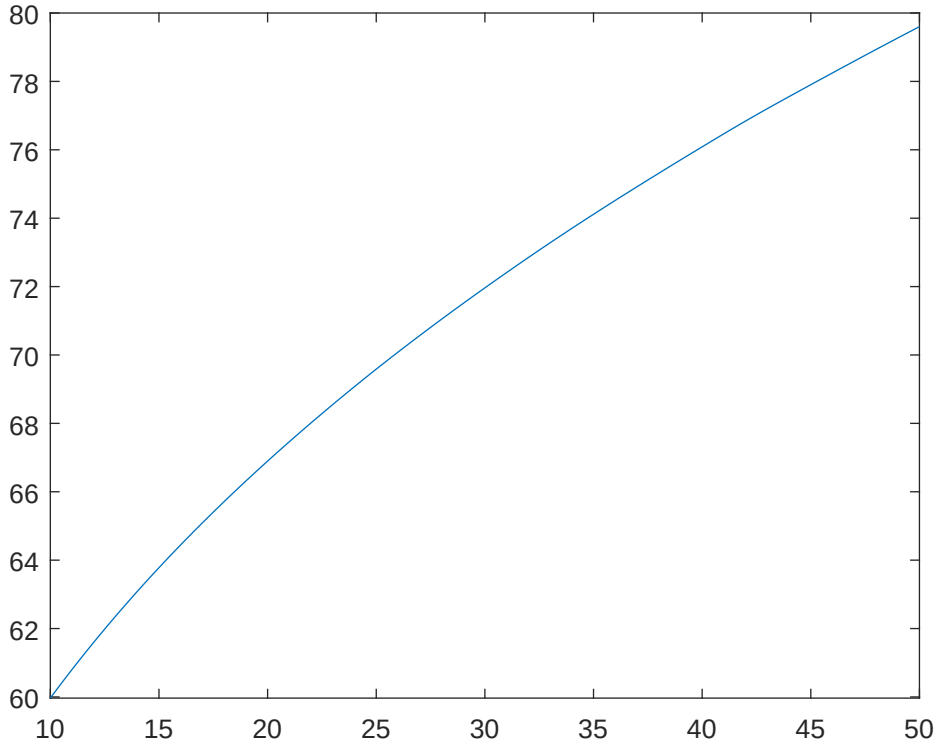
Quantitative Macro HW 3

Alexander Wurdinger

October 22, 2019

Q1

brute force



Doing all the steps described in the slides, I arrive 184 iterations at the plotted value function. The process took 0.117834 seconds. seconds.

Part b and c

Adding the described variation I arrive at the same value function that attains a fixed point. Number of iterations are also equivalent, but time needed to arrive at the fixed point is 0.280981 seconds. seconds respective 0.269514 seconds.

So while in theory those variations should speed up the process they seem to slow it down. For this specific problem this not to surprising as while in a

to attain χ only two matrices have to be added, arguably a really fast process in matlab, in b and c χ is calculated in a loop.

If there are more grid points to be evaluated the speeding up processes might work, as then there are less entries per row to be evaluated in the max function when using the concavity or no maximization at all when using the monotonicity of the policy functions.

Part d

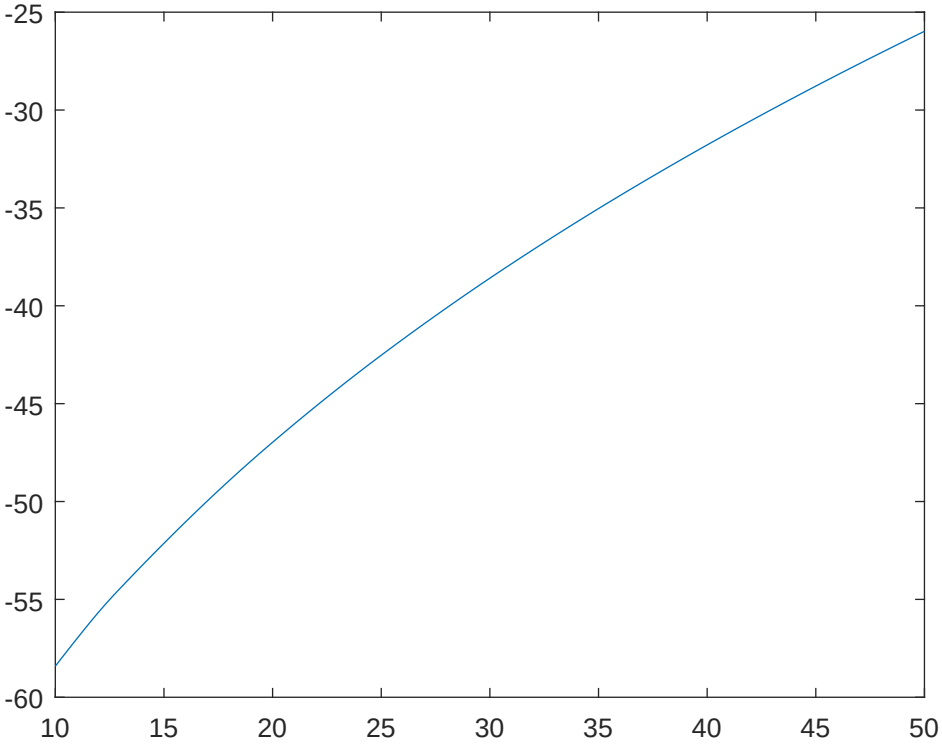
As using local search without using the monotonicity of the policy function does not seem really useful, I implemented both variants at the same time and the same equilibrium value function was achieved in the same amounts of iterations as before. Time spend was 0.273461 seconds. So faster then only using monotonicity.

It has to be noted that several runs of the program revealed that the bounds have to be set quite high (102) to avoid them being binding.

Part e,f

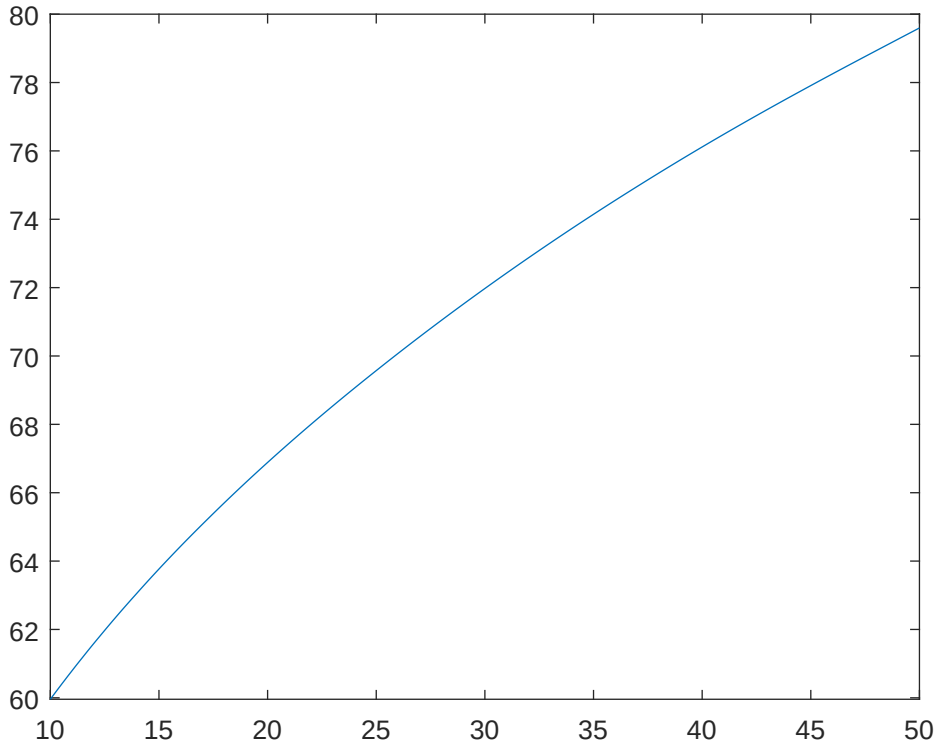
I choose initial iteration of the value function of 50, 100, 150. Then i combine them with policy iterations with 5, 10, 20 and 50 steps in between policy reassessments. It turns out that the fastest way of achieving convergence is doing 50 initial iterations and then 50 policy function iterations between reassessments. It takes 0.067751 seconds. And in three rounds of policy function iterations the value function is iterated 138 times in addition to the 50 initial iterations. This is in total very close to the 184 iterations in the brute force algorithm.

Q2



Same procedure as in Q1 with added labour supply.

Q3



As there are still problems with my chebychev approximation algorithm. I used a polynomial (degree 5) interpolation to approximate the value function in each iteration. The value function then converges. The whole process needs 52.948902 seconds and is therefore so far the slowest option in this example. The interpolation and numerical optimization are rather slow processes, which explain the slow convergence. however it does converge in less iteration,i.e. 167.