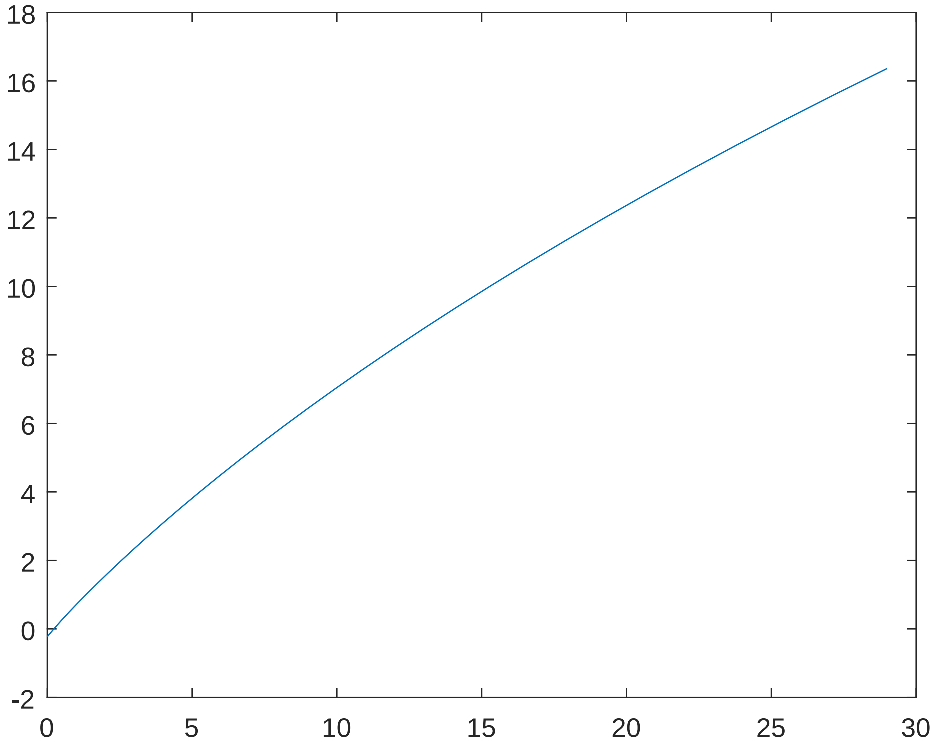
Quantitative Macro Project 2

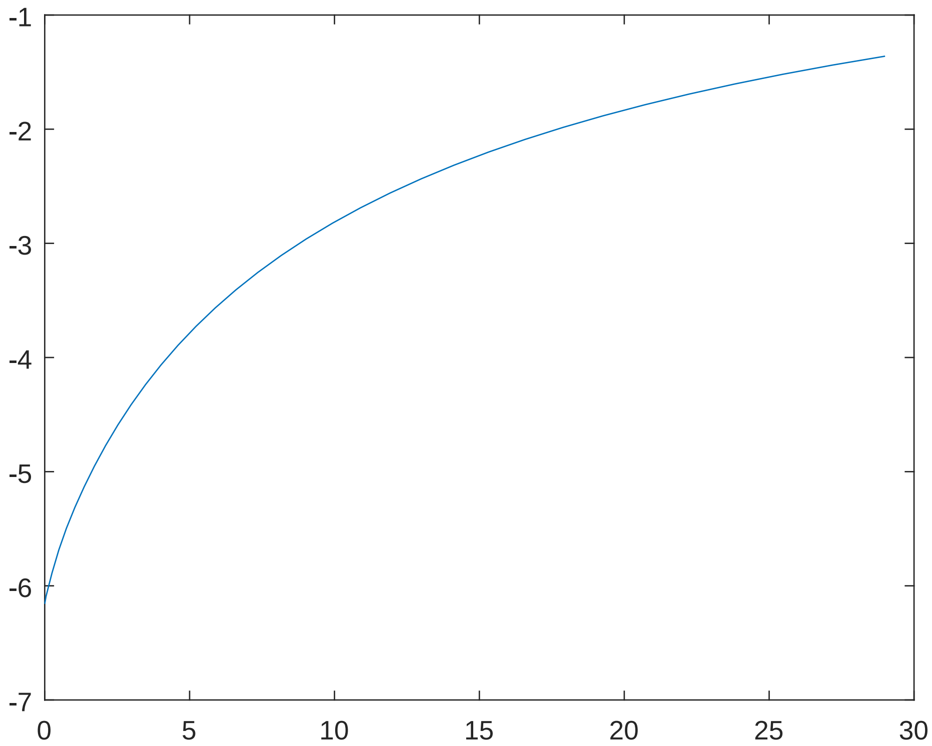
Alexander Wurdinger

November 25, 2019

problem c

1. Using the value function iteration with assets and the current shock as state variables I achieve convergence in 120 and 139 iterations for a theta of 1 and a theta of 4. The corresponding value function (plotted against assets in the 3rd outcome of the shock) are the following:



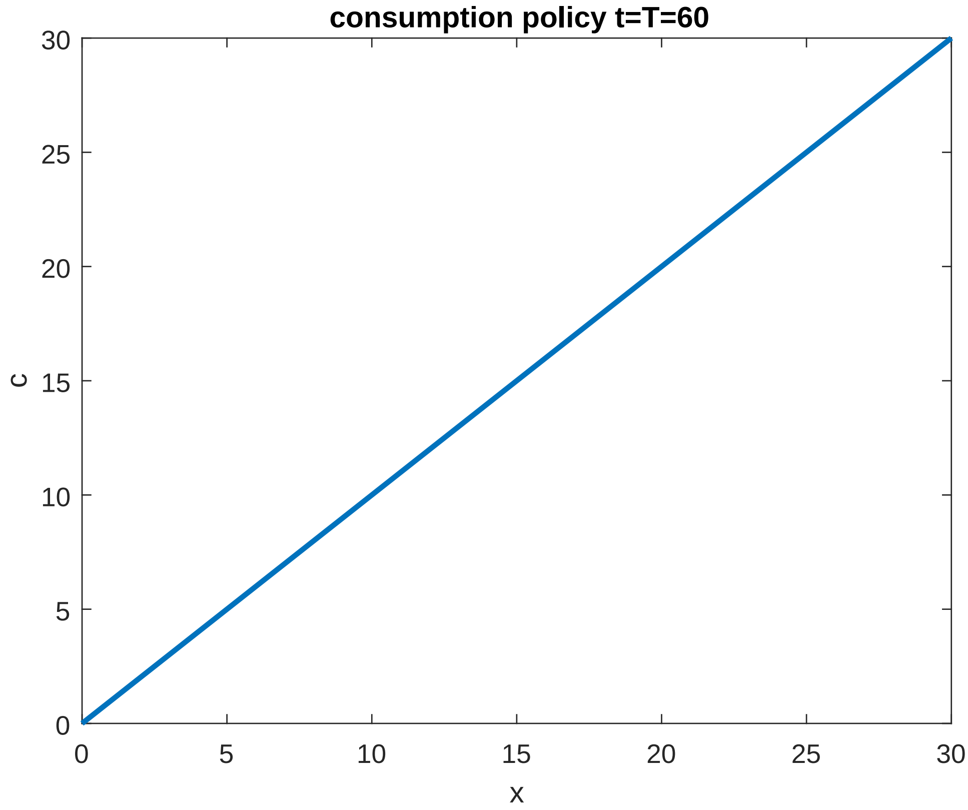


In order to account for the borrowing limit I simply created a positive grid of assets, so negative assets can never be chosen.

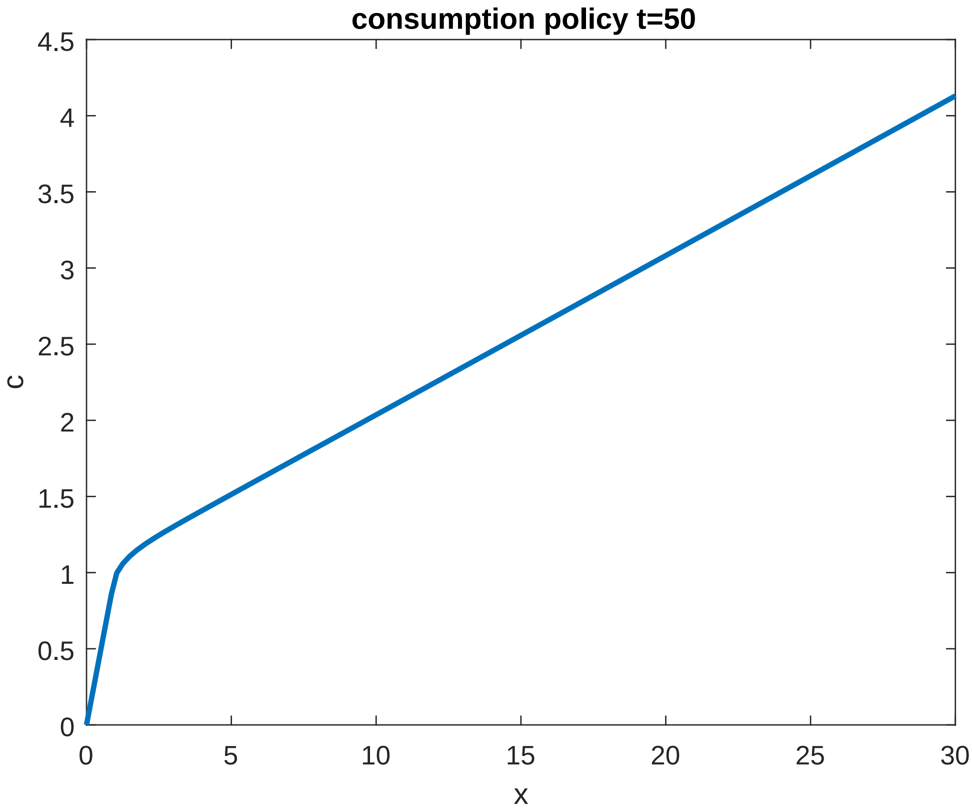
2. Introducing a Howard improvement, where I iterate 50 times on the value function before using the policy function to further iterate on the value function, reduces time to convergence. For theta 1 it reduces the time from 0.17 to 0.14 and for theta 4 from 0.19 to 0.1 seconds.

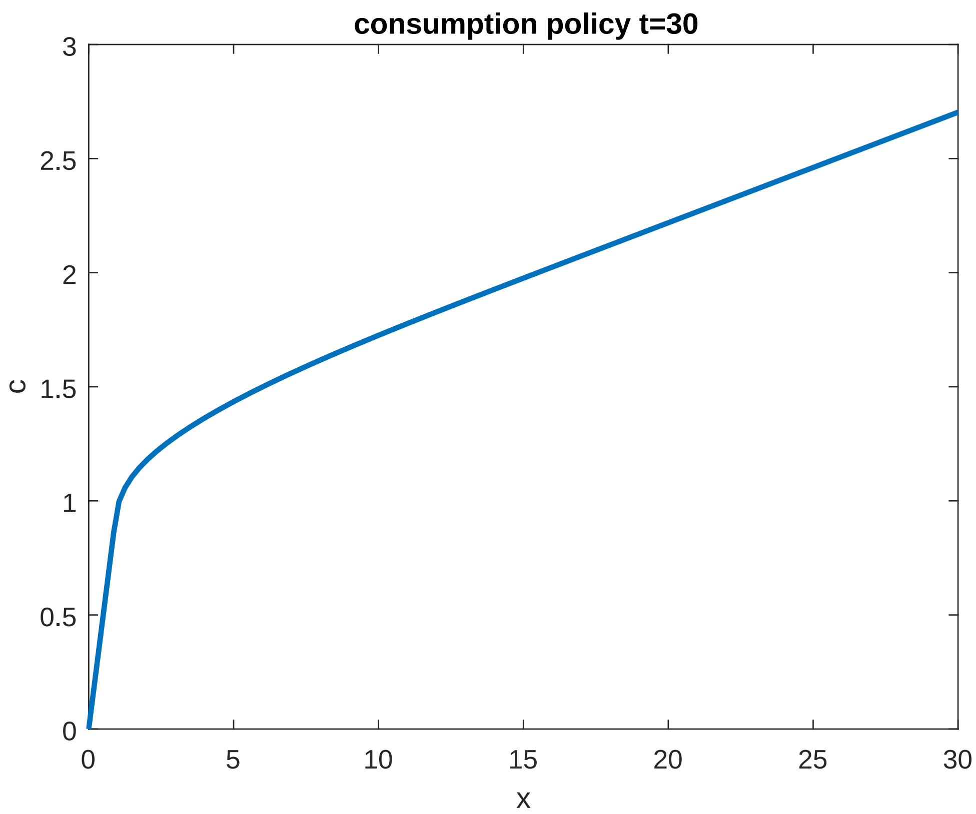
problem d

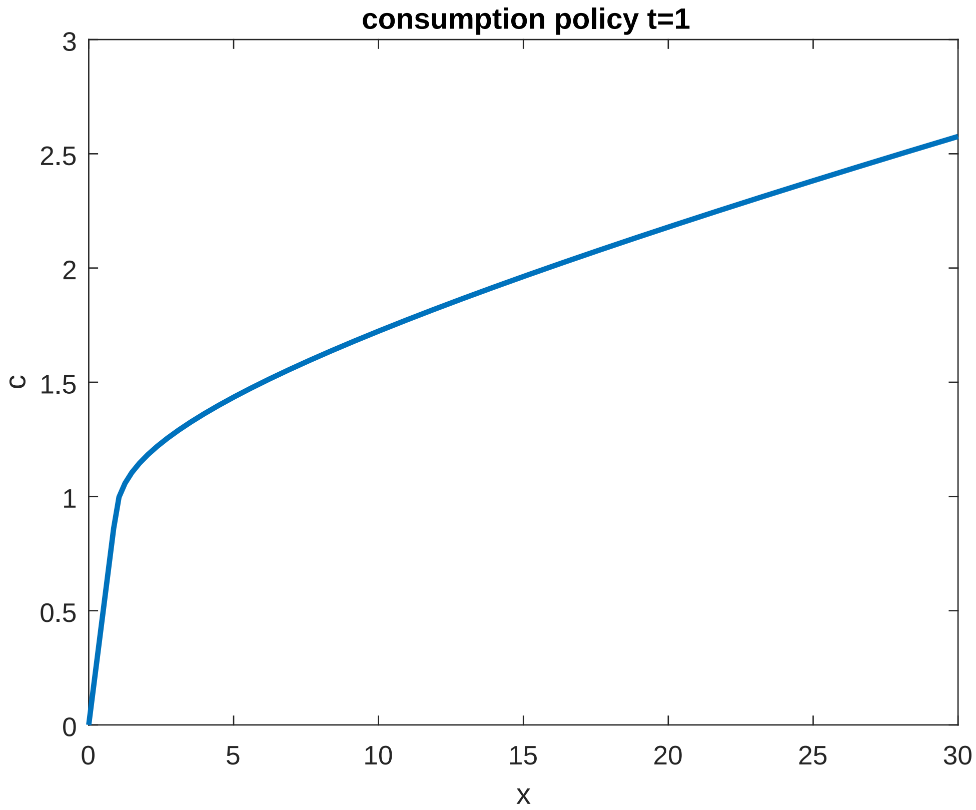
Solving the model for finite T, means as a solution T different policy functions for each period. Those policy function can be calculated recursively starting in T. In T, saving does not make sense such that the $c = x$ is optimal. In the code, consumption in the final period will be equal to the grid of cash on hand. The policy function in $t = T$ (i.e. 60 in my example) is



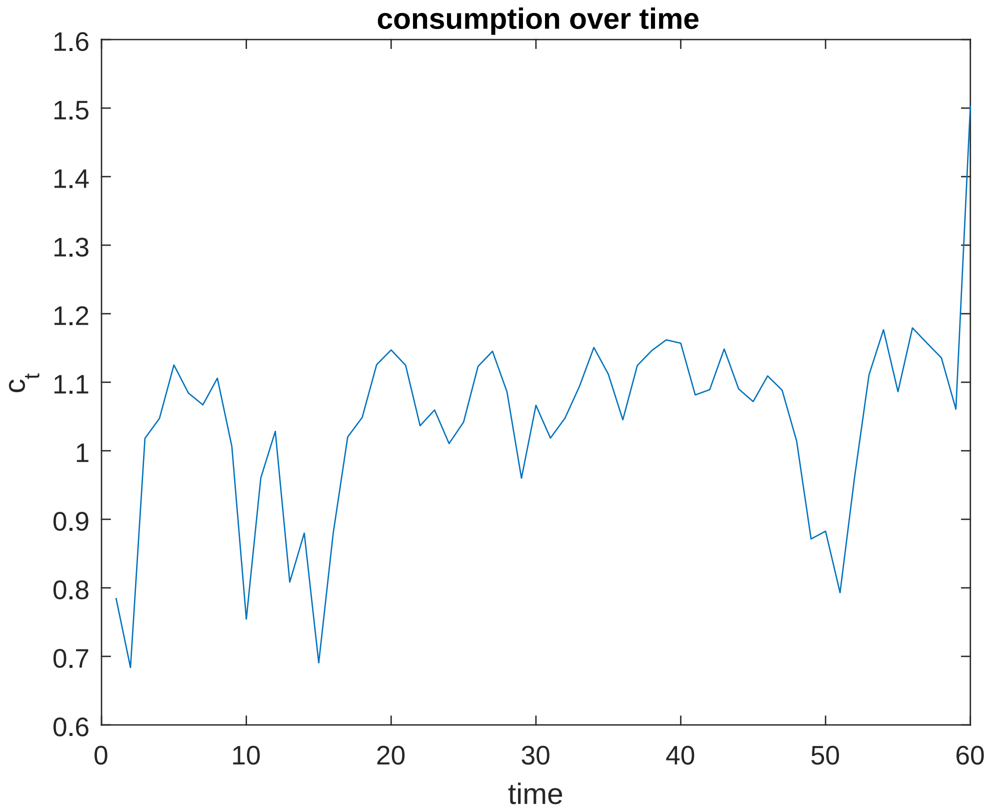
Iterating backwards, policy functions for $t = 50, 30, 1$ look like


consumption policy t=50
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
c
0
5
10
15
20
25
30
x

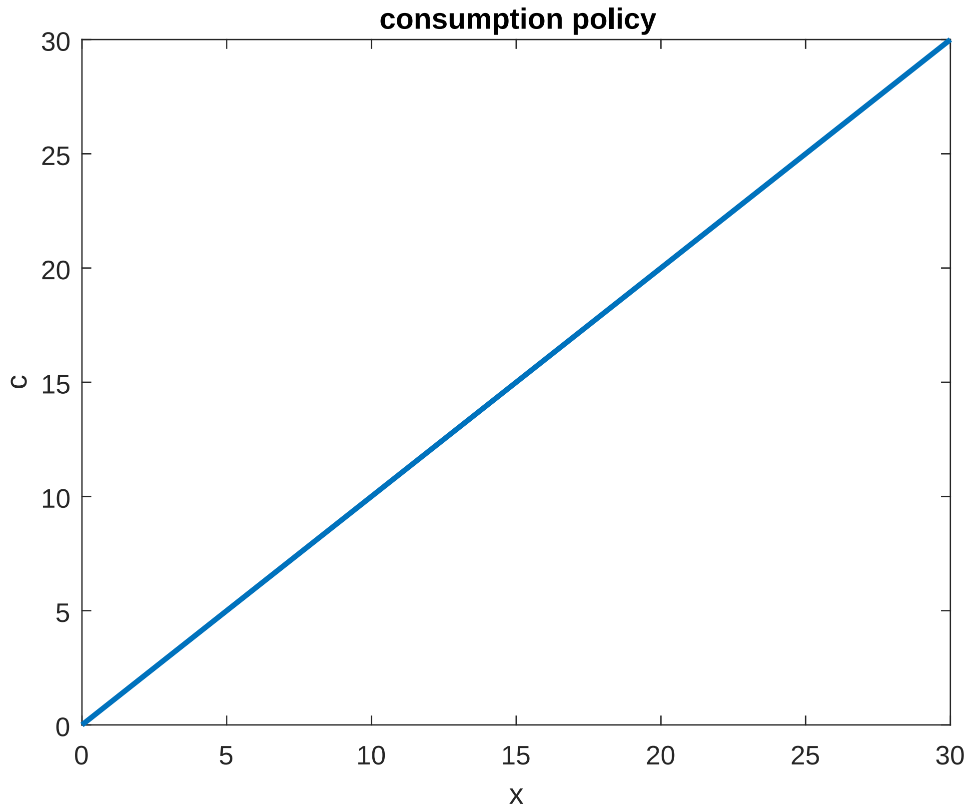

consumption policy t=30
c
x
0
0.5
1
1.5
2
2.5
3
0
5
10
15
20
25
30

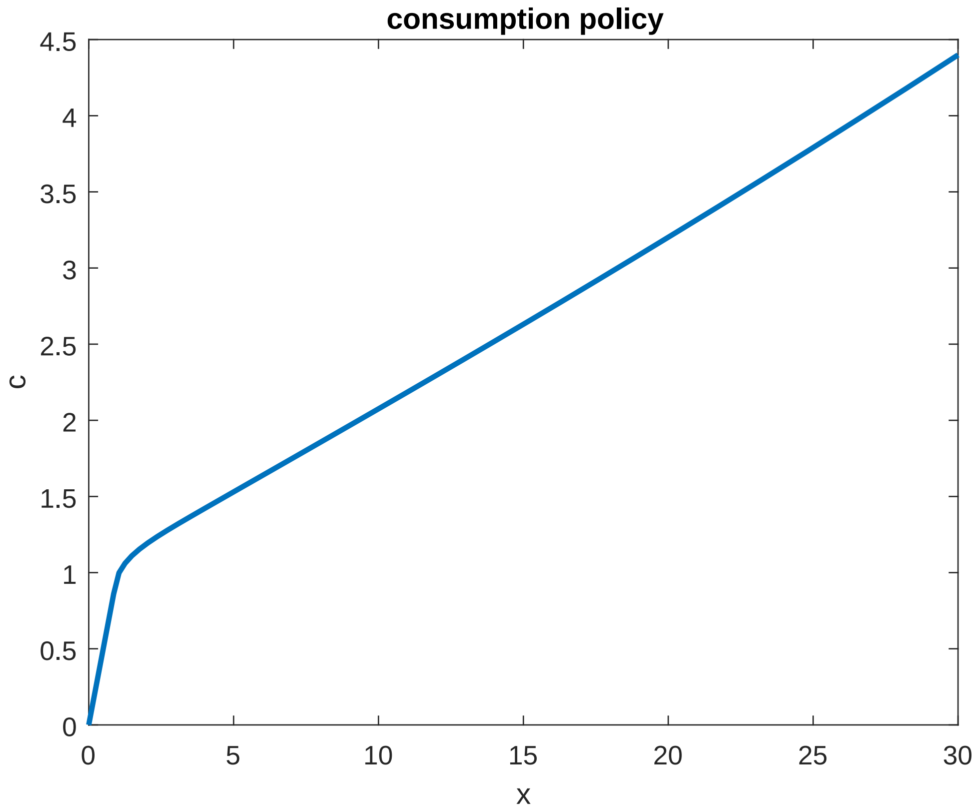


I can also simulate a path for consumption over the 60 years. A possible path can look like

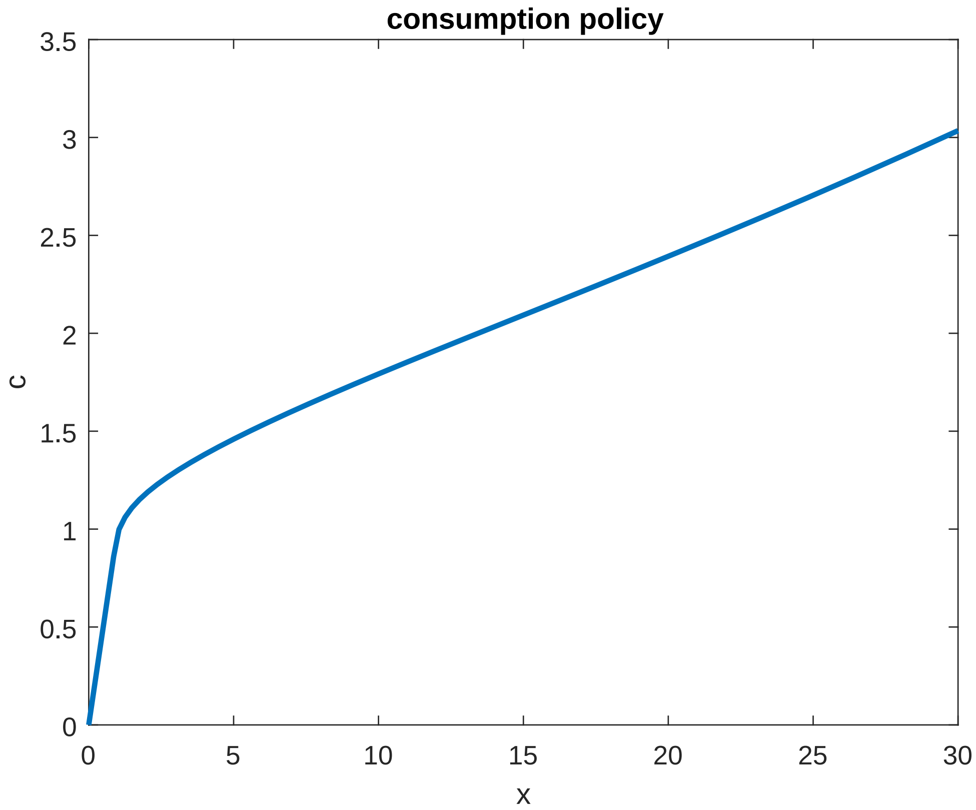


Adding the survival risk leaves the policy functions qualitatively unchanged. As the survival risk effectively decreases the discount factor, policy functions are flatter then without the risk. Plotting it for $t = 60(T), 50, 1$.


consumption policy
c
x
0
5
10
15
20
25
30



consumption policy
c
x
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
0
5
10
15
20
25
30



problem e

1. Interpolating over lambda instead of the value function decreases my speed to convergence, but slightly decreases the mean euler equation error. The algorithm gets slower because of the transformation that have to be performed, but because the set of lambda is smaller, the approximation by the interpolation is also smaller reducing the mistake.

2. Checking for the borrowing constraint, that is a positive mu, will only have to be performed for small values in the grid. More precisely for all values in the grid after the first one for which the constraint is not binding, the constraint will also not bind.

The lowest level of savings is 0, as defined by the boworring constraint.

The highest possible value of savings defined by the grid on cash on hand is $x_{max} - \epsilon_{min} = 29.362$.