Quantitatative Macroeconomics.
Homework 3.

SZTACHERA, Maciej Jan

all content produced in cooperation with Sebastian Zalas

October 30, 2020

```
The system data:
Processor Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, 1801 Mhz, 4 Core(s), 8 Logical Process
Installed Physical Memory (RAM) 8.00 GB
OS Name Microsoft Windows 10 Pro
Version 10.0.18363 Build 18363

Python version: 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
```

1. Pose the recursive formulation of the sequential problem without productivity shocks. Discretize the state space and the value function and solve for it under the computational variants listed below. In all these variants use the same initial guess for your value function.

I have obtained the same policy functions for all code variations a)-f). The value function is concave and increasing. Future capital and consumption policy functions are increasing in the current period capital. The labour policy function is constant as labour is exogenous in this exercise. The results below are for the number of gridpoints $S = 300$.

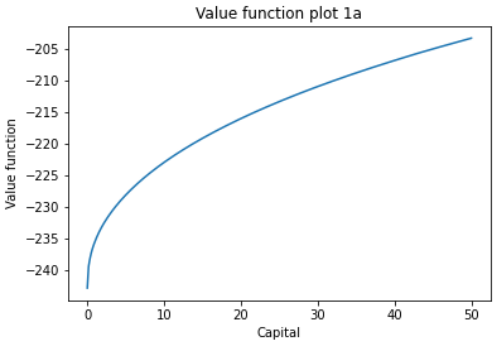
a)

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

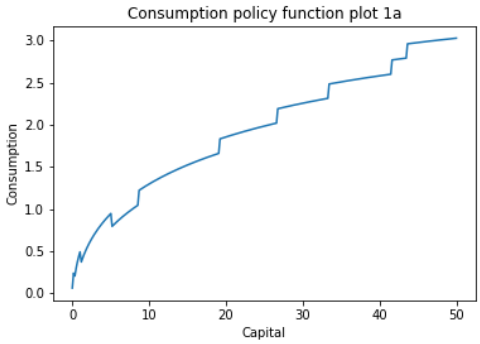
```
Ex1a VECTORIZED runtime: 3.2413036823272705
Ex1a LOOP runtime: 275.8275315761566
```

The improvements in the next points can only be applied in loops. The LOOP runtime is much higher than that for the VECTORIZED LOOPING procedure.

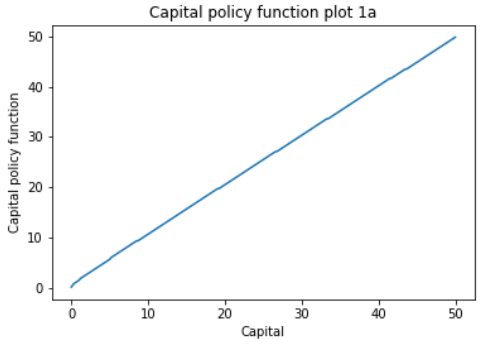
Figure 1: Exercise 1 Policy functions



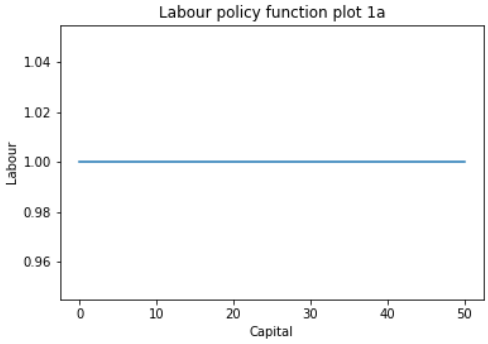
(a) Ex1 Value function



(b) Ex1 Consumption policy function



(c) Ex1 Next period capital policy function



(d) Ex1 Labour policy function

b)

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

`Ex1b runtime: 57.09018611907959`

The application of the monotonicity of the value function substantially improves efficiency to the standard brute force looped approach but its efficiency is well below that of the vectorized looping.

c)

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

`Ex1c runtime: 222.32978200912476`

The improvement based on the concavity of the value function speeds up the algorithm slightly but much less than monotonicity and is way less efficient than the vectorized looping algorith.

d)

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

```
Ex2d runtime: 103.14149904251099
Ex2d no. iterations: 1141
```

Local search substantially improves the runtime compared to other loop-based methods. The main problem in the endogenous labour supply problem is that the local search radius is very large, which limits the efficiency gains of this method. The maximum distance between the future capital gridpoint and iteration specific policy function must be set at levels very close to the size of the grid.

e)

Iterations of the value function taking into account both concavity of the value function and monotonicity of the decision rule

```
Ex1e runtime: 221.03962349891663
Ex1e no. iterations: 2
```

The combination of the monotonicity and concavity rules is less efficient than any of them applied separately but slightly more efficient than the brute force loop calculation. Similar to the monotonicity and concavity algorithms it only requires 2 iterations to converge.

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.

```
Starting at iteration 0:
Ex1f runtime: 1.3515169620513916
Ex1f no. iterations: 434
Starting at iteration 5:
Ex1f runtime: 1.2701475620269775
Ex1f no. iterations: 383
Starting at iteration 15:
Ex1f runtime: 1.2151989936828613
Ex1f no. iterations: 442
```

g)

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

```
Number of Howard iterations H = 5:
Ex1f runtime: 4.013910531997681
Ex1f no. iterations: 43
H = 10
Ex1f runtime: 4.036184310913086
Ex1f no. iterations: 43
H = 20
Ex1f runtime: 4.2148566246032715
Ex1f no. iterations: 43
H=50
Ex1f runtime: 4.195053577423096
Ex1f no. iterations: 43
```

2. Redo item 1 adding a labor choice that is continuous. For this, set $\kappa = 5{:}24$ and $\nu = 2{:}0$. the state space and the value function and solve for it under the computational variants listed below. In all these variants use the same initial guess for your value function.

I have obtained the same policy functions for all code variations a)-f). The value function is concave and increasing. The future capital policy function is increasing in the consumption policy function is concave. The labour policy function is increasing in the current level of capital. The results below are for the number of gridpoints $S = 200$.

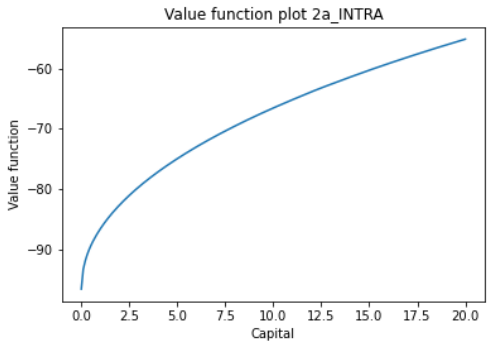
a)

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

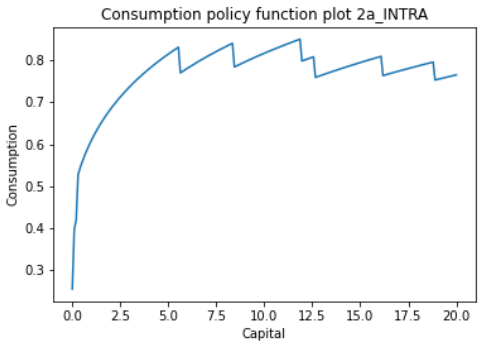
```
Ex2a runtime: 12.630058526992798 \\
Ex2a no. iterations: 1141   \\
Ex2a_LOOP runtime: 57.34546971321106  \\
Ex2a_LOOP no. iterations: 1141 \\
```

The improvements in the next points can only be applied in loops. The LOOP runtime is much higher than that for the VECTORIZED LOOPING procedure. The number of iterations is the same.

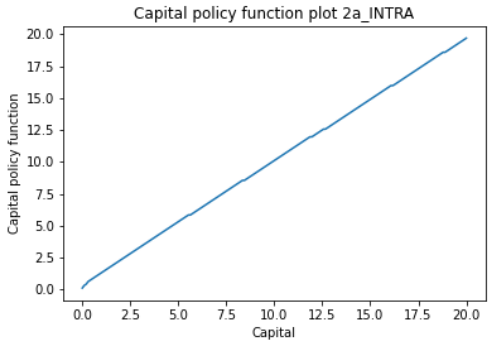
Figure 2: Exercise 2 Policy functions



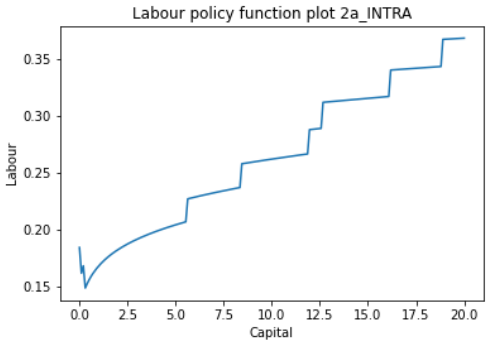
(a) Ex2 Value function



(b) Ex2 Consumption policy function



(c) Ex2 Next period capital policy function



(d) Ex2 Labour policy function

b)

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

```
Ex2b runtime: 57.85967779159546
Ex2b no. iterations: 2
```

The application of the monotonicity of the value function does not improve the runtime of the algorith with the endogenous labour force, even though the number of iterations required to converge has dropped substantially.

c)

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

```
Ex2c runtime: 101.33425545692444
Ex2c no. iterations: 1141
```

The improvement based on the concavity of the value function performs worse than the standard brute force algorithm. The number of iterations is the same as in the baseline brute force algorithm.

d)

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

```
Ex1d runtime: 56.74851036071777
Ex1d no. iterations: 2
```

Local search substantially improves the runtime compared to other loop-based methods. The result is fairly similar to the monotonicity based method but has weaker theoretical justification.

e)

Iterations of the value function taking into account both concavity of the value function and monotonicity of the decision rule

```
Ex2e runtime: 97.47393155097961
Ex2e no. iterations: 2
```

The improvement with both monotonicity and concavity performs better than concavity alone but worse than monotonicity alone.

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.

```
H_S - starting iteration of the Howard improvement
H_S = 0
Ex2f runtime: 2.4096431732177734
Ex2f no. iterations: 467
H_S = 5
Ex2f runtime: 2.4424192905426025
Ex2f no. iterations: 414
H_S = 15
Ex2f runtime: 2.5352253913879395
Ex2f no. iterations: 406
```

Unlike in the case of the exogenous labour, the immediate turning on of the Howard improvement is best in terms of computation time.

g)

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

```
H = 5
Ex2f runtime: 2.3899247646331787
Ex2f no. iterations: 467
H = 10
Ex2f runtime: 2.529910087585449
Ex2f no. iterations: 467
H = 20
Ex2f runtime: 2.5352747440338135
Ex2f no. iterations: 467
H = 50
Ex2f runtime: 2.4787631034851074
Ex2f no. iterations: 467
```

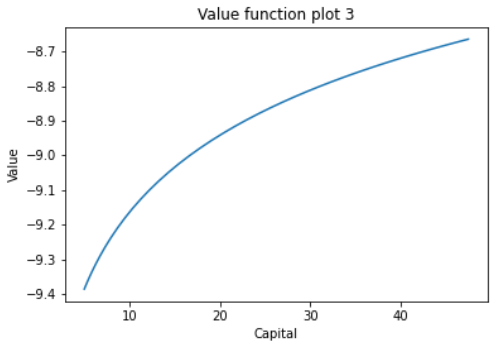
3. Redo item 1 using a Chebyshev regression algorithm to approximate the value function. Compare your results.

The calculation time and number of iterations are very low.

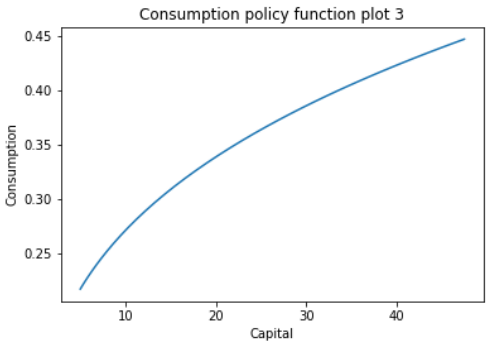
```
Ex3 runtime: 0.01595139503479004
Ex3 no. iterations: 2
```

The problem is that consumption is well below the steady state level (maximum .32 compared to .82 in the steady state).

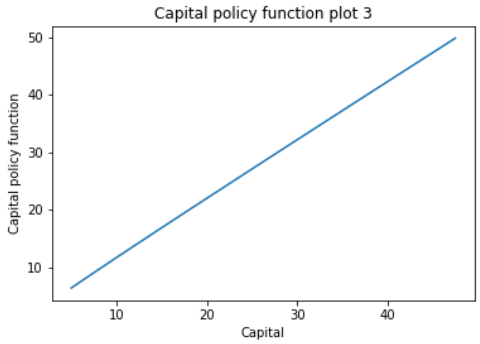
Figure 3: Exercise 3 Policy functions



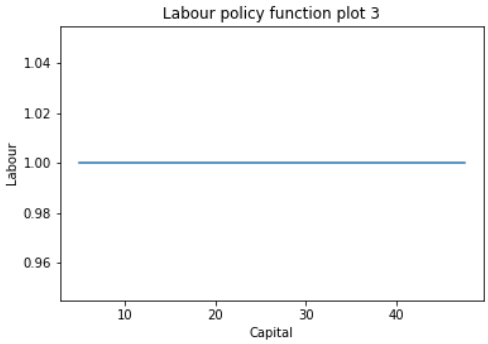
(a) Ex3 Value function



(b) Ex3 Consumption policy function



(c) Ex3 Next period capital policy function



(d) Ex3 Labour policy function