Exercise set 2

Introduction to Python Programming for Economics & Finance

Richard Foltyn
University of Glasgow

May 23, 2023

Contents

1 Plotting lifetime utility

In the first lab, we considered the the following two-period consumption/savings problem

$$\begin{aligned}
\max_{c_1,c_2} \quad & \frac{c_1^{1-\gamma}}{1-\gamma} + \beta \frac{c_2^{1-\gamma}}{1-\gamma} \\
\text{s.t.} \quad & c_1 + \frac{c_2}{1+r} = w \\
& c_1 \geq 0, \; c_2 \geq 0
\end{aligned}$$

where the solution was given by

$$\begin{aligned}
c_1 &= \alpha \cdot w \\
c_2 &= (1+r)(1-\alpha)w \\
\text{with} \quad \alpha &= \left[1 + \beta^{\frac{1}{\gamma}} (1+r)^{\frac{1}{\gamma}-1}\right]^{-1}
\end{aligned}$$

Using the parameters $\beta = 0.96$ and $r = 0.04$, plot the lifetime utility implied by the optimal allocation for 100 uniformly spaced wealth levels w on the interval $[0.1, 1.0]$. Plot two lines (in a single plot) for two values of the relative risk aversion parameter γ:

1. $\gamma = 1$ with color `'blue'`
2. $\gamma = 2$ with color `'red'` using a dashed line.

Label both axes and include a legend to distinguish these two scenarios.

You can use the following functions to compute the lifetime utility:

[1]:
```
import numpy as np

# Function to compute per-period utility
def util(c, gamma):
    if gamma == 1:
        u = np.log(c)
    else:
```

```
        u = c**(1.0 - gamma) / (1.0 - gamma)
    return u

# Function to compute lifetime utility
def lifetime_util(c1, c2, beta, gamma):
    return util(c1, gamma) + beta * util(c2, gamma)
```

2 Markov processes

Consider the following Markov process defined on two states with transition matrix

$$\Pi = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$$

Let $\mu_t = \begin{bmatrix} x \\ 1-x \end{bmatrix}$ denote the distribution of a large number (a unit mass) of households over these two states at time t. The distribution in period $t+1$ is then given by

$$\mu_{t+1}^{\top} = \mu_t^{\top} \Pi$$

Assume that at time zero the distribution is given by $\mu_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Simulate the Markov process for $T = 50$ periods and plot the fraction of households in state 1.

Hint: You need to perform matrix multiplication in a loop and save the value of μ_t for each period.

3 Custom function to compute averages

Implement the function `myaverage()` defined as

```
def myaverage(x):
    # Compute and return average of values in x
```

which takes an array (or list, tuple) `x` containing numerical values and returns its average. Implement this function using a `for` loop.

Time your function using `%timeit` and compare the run time to the NumPy's `np.average()`. Perform these tests for two arrays:

1. An array of ones with 100 elements.
2. An array of ones with 10,000 elements.

Does the relative run time depend on the array size?