UNIVERSITY OF COPENHAGEN

Third exercise class

Class 1 & 5

Introduction to numerical programming and analysis

Jonathan Wenzel Pedersen

Spring 2024

Plan

1. My thoughts on problem sets
2. Theory
3. Problems 1-3 (Functions)
4. Problem 4 (Optimization)
5. Problems 5-8 (Utility maximization problem)
6. Recap

My thoughts on problem sets

My thoughts on problem sets

They are located here (I'll show you how to fork and clone)

- A point that bears repeating: You're not required to start from scratch at every problem, finding something similar from the lecture notes or even the internet and then rewriting code to apply to the specific problem is encouraged
- Try and understand the harder problems conceptually first: what am I'm trying to do, which steps does this require, then you can start writing up the code to solve it
- It's fine to look at the solutions, but if you do, make sure you understand each line and how it all comes together
- The internet (and me) is your friend, when getting weird errors
- Also: experimentation is key, running something and getting an error allows you to learn something

Projected time plan:

- 15:20-15:25: I'll go through
- 15:25-15:45: You'll do problems 1-3 yourself
- 15:45-15:55: You'll do problem 4 yourself
- 15:55-16:00: I'll do problem 4 on my screen.
- 16:00-16:15: Break
- 16:15-16:55: You'll do problems 5-8 yourself
- 16:55-17:00: Recap and poll about class structure

Theory

Data types

- In this problem set we will be mainly using text and numbers, for text python uses the **string (str)** data type, while for numbers python uses **integer (int)** for whole numbers and **float** for decimnal numbers.
- Note that in python 6 is not the same as 6.0 due to the difference data types!
- We can do different operations depending on what data type we are dealing with for example "+"operation will for a string concatenate the two strings i.e. "t"+ "est"will become "test", for numbers like integers it will be 1+5 will become 6.
- String and numbers usually cannot be mixed if we want to print out numerical results f-strings can be used, the basic syntax for that is: f'{**value**:*width*.*digits*f}', where *width* is the total width of the value i.e. the total number of characters and *digits* is digits after the decimal point, but other letters than f after *digits*, leads to different options. A general guide can be found here

Using optimize.minimize

```
from scipy.optimize import minimize

def objective_function(x):
    return x[0]**2 + x[1]**2

# Constraint functions must return zero or a positive
                                value
def constraint(x):
    return x[0] + x[1] - 10

# Constraints dictionary
cons = ({'type': 'eq', 'fun': constraint})

# Bounds for each variable
bounds = ((0, None), (0, None))

result = minimize(objective_function, x0=0, method='SLSQP
                                ', bounds=bounds,
                                constraints=cons)
```

Problems 1-3 (Functions)

Problem 1

Problem 1

Implement a Python version of the following function:

$$u(x_1, x_2) = \left(\alpha x_1^{-\beta} + (1-\alpha)x_2^{-\beta}\right)^{-\frac{1}{\beta}}$$

Hint:

- Remember that in python $\wedge$ is coded using $**$

Problem 2

Problem 2

Construct a Python function print_table(x1_vec,x2_vec) to print values of u(x1,x2) the result should be:

	0	1	2	3	4
0	1.05	1.16	1.44	1.48	1.57
1	1.16	1.30	1.66	1.71	1.83
2	1.44	1.66	2.30	2.40	2.64
3	1.48	1.71	2.40	2.50	2.77
4	1.57	1.83	2.64	2.77	3.10

Hints:

- To create the side numbers consider using enumerate (guide)
- f-strings is an excellent tool for printing your results. The basic syntax for printing a float is: f'{**value**:*width*.*digits*f}', where *width* is the total width of the value i.e. the total number of characters and *digits* is digits after the decimal point, but other letters than f after *digits*, leads to different options. A general guide can be

Problem 3

Problem 3

Plot the values of the utility function using matplotlib

Hints:

- Use lecture notebook 1-plot more specifically "Plot the utlity function"("in [52]"). There is no need to memorise all the plotting code
- fig.tight_layout() at the end it will make it a little prettier

Problem 4 (Optimization)

Problem 4

Problem 3

Solve for following minimization problem:

$$\min_x f(x) = \min_x sin(x) + 0.05 \cdot x^2$$

Create a solution using for loop, another using scipy optimize and visualize the function and the two solutions using matplotlib.

Hints:

- Numpy has the sine function np.sin()
- For scipy optimize see lecture notes from "1-optimize"or see documentation. I will recomend using the 'Nelder-Mead' method.

Problems 5-8 (Utility maximization problem)

Problem 5

Problem 5

Solve the 5-good utility maximization problem with the preferences $\alpha = (\alpha_1, \alpha_2, ..., \alpha_5)$ for the following function:

$$V(p_1, p_2, ..., p_5, I) = \max_{x_1, x_2, ...x_5} x_1^{\alpha_1} \dots x_5^{\alpha_5}$$

$$s.t.$$

$$E = \sum_{i=1}^{5} p_i x_i \leq I, \quad p_1, ...p_5, I > 0$$

$$x_1, ...x_5 \geq 0$$

Create a solution using for loops

Hints:

- Use same method as in problem 4 just with more goods this time (how does more goods impact for loops?)

Problem 7 and 8

Problem 5

Same problem as problem 5, just solve using scipy optimizer (problem 7: Constrained and problem 8: unconstrained)

Hints:

- For unconstrained use same method as problem 4, but insert constraint (Expenditure and I) and remember that we are maximizing not minimizing so how can you use same method but get maximization?
- For constrained you want to create a function see lecture notes "1-optimize"case 3 for inspiration.

Problem 6

Problem 6

Solve problem 5 using itertools

Hints:

- The problem is just asking for a rewrite of the solution you made for problem 5. Several for loops can be combined into one line of code using itertools.

Recap

You to-do

The deadline for the inaugural problem is 24th of March. So your priorities should be:

1. Make sure you understand what we've covered today in PS1.
2. Prepare yourself for next exercise class by reading PS2. PS1 and PS2 contains all the tools you need for your inaugural project.