UNIVERSITY OF COPENHAGEN

Fourth exercise class

Section 1 & 5

Introduction to numerical programming and analysis

Jonathan Wenzel Pedersen

Spring 2024

Overview

1. Plan for today
2. Theory
3. 2.1-2.2 shown in class
4. Problem 2.3-2.6
5. Recap

Plan for today

What are we doing today?

Today we will be working with Problem set 2, which will work with the economic problem of finding the Walras equilibrium in a multi-agent economy.

- Solve a series of practice questions before combining them for problem 2.5.
- To make sure you get to do the final problems I will solve the first two.

Projected time plan:

- 15:20-15:35: Theory and I'll do problems 2.1-2.2
- 15:45-16:00: You'll do problems 2-3-2.6 yourself
- 16:00-16:15: Break
- 16:15-16:55: You'll do problems 2-3-2.6 yourself
- 16:55-17:00: Recap and poll about class structure

Theory

Theory

Last time you learned and used:

- Functions
- Data types
- Scipy.optimize
- For loops

This class we will be using:

- Masking
- Widgets
- Different .py files (Modular programming)

Using Masking

Masking allows for selective processing of elements in an array based on specific conditions.

```
import numpy as np

# Create an array
a = np.array([1, -2, 3, -4, 5])

# Create a mask for negative values
mask = a > 0

# Apply mask to get only positive values
positive_values = a[mask]

print(positive_values)
# Output: [1 3 5]
```

2.1-2.2 shown in class

Notes for problem 2.1

Problem 1

Replace the missing lines of code to get the same output as the code already gives, but when you also insert a new np.random.seet() inside the code.

Hints:

- You need the *np.random.get_state()* and *np.random.set_state()* explained in lecture 4 section 3.2

Notes for problem 2.2

Problem 2

Find the expected value and variance of:

$$E[g(x)] \approx \frac{1}{N}\sum_{i=1}^{N} g(x_i)$$

$$VAR[g(x)] \approx \frac{1}{N}\sum_{i=1}^{N}\left(g(x_i) - \sum_{i=1}^{N} g(x_i)\right)$$

Where $x_i \sim N(0, \sigma)$ and:

$$g(x,\omega) = \begin{cases} x & if \;\; x \in [-\omega, \omega] \\ -\omega & if \;\; x < -\omega \\ \omega & if \;\; x > \omega \end{cases}$$

Notes for problem 2.2 (part 2)

Think in steps:

- Create an x-array that contains elements pulled from a normal distribution
- Create an array that contains the elements of the output of the g-function.
- Use numpy functions to calculate the statistics.

Problem 2.3-2.6

Notes for Problem 2.3

Problem 2.3

Plot a histogram and a normal distribution and make the plot interactive.

Hints:

- The first missing line is that you need to import the function *norm* from the *scipy.stats*-module. (documentation here)
- For the next two missing lines, look at lecture notebook "1 - Random numbers basics"more specifically "3.3 analytical results".
- For interactive sliders use widgets, look at "1.2 interactive figure"in lecture notebook "1 - Random numbers example". Notice that the *fitting_normal()*-function in the PS, can be called in a similar fashion as the *interactive_figure()*-function in the lecture note.

Notes for Problem 2.4

Problem 2.4

Create your own function in *mymodule.py* and call it in the main notebook.

Hints:

- You can get a function from another file using *from mymodule.py import **.
- You can use *%load_ext autoreload* and *%autoreload 2* to make your import automatic (very useful if your constantly making changes to your other files.

Notes for Problem 2.5

Problem 2.5

Write a function to solve the equilibrium of:

Consider an **exchange economy** with

1. 2 goods, (x_1, x_2)
2. N consumers indexed by $j \in \{1, 2, \ldots, N\}$
3. Preferences are Cobb-Douglas with truncated normally *heterogenous* coefficients

$$u^j(x_1, x_2) = x_1^{\alpha_j} x_2^{1-\alpha_j}$$
$$\tilde{\alpha}_j \sim \mathcal{N}(\mu, \sigma)$$
$$\alpha_j = \max(\underline{\mu}, \min(\overline{\mu}, \tilde{\alpha}_j))$$

4. Endowments are *heterogenous* and given by

$$e^j = (e_1^j, e_2^j)$$
$$e_i^j \sim f, f(x, \beta_i) = 1/\beta_i \exp(-x/\beta)$$

Hints:

- Use same method as I have shown you to break down the problem.
- Use the code from lecture (just below in the notebook).

Notes for Problem 2.6

Problem 2.6

Fill in missing lines, note that the the code can be run and if "an error is found"is printed then input in place of missing lines is not correct.

Recap

Thanks for today

An opinion poll:

I want to know your opinion on the classes via this poll.