Class 10, Problem Set 6

UNIVERSITY OF COPENHAGEN

Introduction to Programming and Numerical Analysis

Data project

UNIVERSITY OF COPENHAGEN

Overall good projects but for the model project and at the exam think about:

- Explanatory markdown
- Don't repeat yourselves
- Thoughtful plotting
- **RESTART KERNEL AND RUN ALL CELLS**

Plan for today

1. Working with equations
    - SciPy linalg
    - SymPy
2. Working on PS6

UNIVERSITY OF COPENHAGEN

SciPy linalg

Another one of SciPy's modules: `from scipy import linalg`

- Makes any operation within the realm of linear algebra:
    - Matrix product
    - Solve system of equations
    - Find eigenvalues
    - ... and so on

SciPy linalg

Let's solve:

$$Ax = b$$

SciPy linalg

UNIVERSITY OF
COPENHAGEN

Let's solve:

$$Ax = b$$

In [1]:

```
import numpy as np
from scipy import linalg
np.random.seed(1900)
A = np.random.uniform(size=(5,5))
b = np.random.uniform(size=5)
print(f'Matrix A:\n{A}\n\nMatrix b:\n {b}')
```

```
Matrix A:
[[0.33224607 0.71427591 0.37834749 0.24908241 0.83598633]
 [0.02005845 0.32670359 0.05606653 0.4008206  0.13288711]
 [0.88711192 0.15490098 0.01708181 0.95781716 0.58999632]
 [0.83959058 0.7146372  0.58705537 0.40933648 0.14603168]
 [0.16407166 0.65717511 0.146494   0.67717016 0.47425348]]

Matrix b:
 [0.78485347 0.85159023 0.84757586 0.42016935 0.20991113]
```

In [2]:

```
# Solve using LU factorization -> Split A in an upper and lower triangular matrix -> Speed
# LU factorize A using linalg
LU,piv = linalg.lu_factor(A)
# Solve using linalg
x = linalg.lu_solve((LU,piv),b)
print(x)
```

```
[-15.33189031 -24.00998148  40.02675108  15.24193293   4.89008792]
```

In [3]:

```python
# Regular solve
print(linalg.solve(A,b))
```

```
[-15.33189031 -24.00998148  40.02675108  15.24193293   4.89008792]
```

SciPy linalg

What do we use it for?

In the first question of the exam 2020 you had to implement the OLS estimator using linear algebra. Recall that:

$$\hat{\beta} = (X'X)^{-1}X'y$$

and can thus, be solved using linalg

SymPy

SymPy is a Python library for symbolic mathematics and lets you solve equations analytically! (Kinda like WolframAlpha or Symbolab). Let's check it out

SymPy

Say that you want implement the utility function of standard OLG agent. We assume agents derive utility from consumption in both periods:

$$U_t = u(c_{1t}) + \frac{1}{1+\rho} u(c_{2t+1})$$

Furthermore, we assume log-preferences

In [4]:

```
import sympy as sm
# Initialize variabels in Sympy
c1,c2 = sm.symbols('c_1t'), sm.symbols('c_2t+1')
rho = sm.symbols('rho')

# Setup utility in sympy
uc1 = sm.ln(c1)
uc2 = sm.ln(c2)
U = uc1 + 1/(1+rho) * uc2
display(U)
```

$$\log\left(c_{1t}\right) + \frac{\log\left(c_{2t+1}\right)}{\rho + 1}$$

SymPy

With SymPy we are able to do many calculations. Say that we need the derivate of U w.r.t. c_{2t+1}:

SymPy

UNIVERSITY OF COPENHAGEN

With SymPy we are able to do many calculations. Say that we need the derivate of U w.r.t. c_{2t+1}:

In [5]:

```
# We just use SymPy's .diff() method:
sm.diff(U,c2)
```

Out[5]:

$$\frac{1}{c_{2t+1}\left(\rho+1\right)}$$

SymPy

Another cool feature is that you can turn your SymPy equations into python functions. This can really tie your model projects together:

- Solve model analytically with SymPy
- Convert your solution to a python function e.g. the law-of-motion in OLG
- Find steady state level of capital using an optimizer

How is it done?

In [6]:

```python
# Use SymPy's lambdify method which takes an iterable of arguments in our case the consumpti
# and of course the function in our case U
util = sm.lambdify((c1,c2,rho),U)

# Compute some utility
util(7,8,0.1)
```

Out[6]:

```
3.836311550582437
```

PS6

Start from 2.3 or watch the video on Absalon about solving system of equations