# Exercise class 9

(week 16)

## Introduction to Programming and Numerical Analysis

## Class 4 and 8

Rosa Haslund Meyer

Spring 2024

KØBENHAVNS UNIVERSITET

## Solving equations – linear and non-linear

## Symbolic math / Python

## Problem set 6

# Linear equations

Linear equations refer to anything that can be written in matrix form, i.e.:

$$Ax = b \iff \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

If A in invertible (a square matrix such that the product of the matrix and its inverse generates the identity matrix), the solution is $x = A^{-1}b$. There are standard routines for inverting matrices, however, these can sometimes be computationally expensive, which is where the following algos come in:

- **Gauss-Jordan** elimination (not that much more efficient)
- **Gauss-Seidel** iterations are faster (but may not always converge)
- **LU-factorization** has no matrix inversion, i.e. very fast

# Non-linear equations

With non-linear equations, it's often necessary for us to be more *general* in our approach.

As such, we often use a root finder/solver:

$$f(x) = b \iff \hat{f}(x) = 0, \quad \hat{f}(x) = f(x) - b$$

Root-finding algos include (among others):

- **Bisection** (as you worked with in ps5), no gradient, but may sometimes be slow
- **Newton** or **Halley**, use gradient (and Hessian) to update guesses effectively
- **Brent** finds an efficient combination

Different algos work well under different conditions – feel free to experiment!

# Symbolic math / Python

Symbolic Python (sympy-package) can do complicated math and solve equations analytically!

Everything is much like what we know from micro and macro courses, which is nice as:

- More familiar, recognisable and intuitive
- Can find solutions to problems with more than one solution
- Can solve the problems exactly, not just down to a numerical approximation
- The code is nice to look at and easy to read

Not so nice as:

- Doesn’t provide a deeper understanding of numerical methods… :(
- Many models have no analytical solution

# Problem set 6 – task 1-4

Task 1-4: linear algebra, solving matrix equations – general tips:

- Go through the tasks slowly and think logically.
- If you forgot some of the math, look that up instead of looking at solutions
- Look at the `scipy` package `linalg`, it has almost everything you would need
- For task A4, remember to import the given module `numecon_linalg`
- Hint for A4: you need to stack the two arrays F and e to get X:

```
F:
 [[ 2.   1. -1.]
 [-3. -1.   2.]
 [-2.   1.   2.]]

e:
 [  8. -11.  -3.]
```

```
X:
 [[  2.    1.   -1.    8.]
 [ -3.   -1.    2.  -11.]
 [ -2.    1.    2.   -3.]]
```

# Problem set 6 – task 5-6

Task 5-6: symbolic math / Python – general tips:

- Go through the tasks slowly
- Think logically, these tasks should be quite straight forward
- Look up the most useful methods/documentation of `sympy`, it has the main functions you will use: `sm.limit()` and `sm.diff()`
- To use `sympy`, you need to “define” your variables: `x = sm.symbols('x')`

# Problem set 6 – Problem: the Solow model

Task 5-6: symbolic math / Python – general tips:

- Make sure to understand the setting and if needed refresh the Solow model:

  ”A standard Solow model predicts that in the long run, economies converge to their balanced growth equilibrium and that permanent growth of per capita income is achievable only through technological progress.”

- Dived expressions into multiple entities to make the code more readable
- Symbolic equations in `SymPy` are represented as `sm.Eq(x, y)`
- For task A8 you should use `sm.lambdify`, google the documentation
- Define each function one at a time

# Next time…

**Video lectures:**

- Unconstrained optimization
- Constrained optimization
- Dynamic optimization

**Exercises – Problem set 7. Solving the consumer problem with income risk**

Note: ps7 is quite long and covers some difficult concepts, so I suggest you have a quick look before class. Ps7 is also the last problem set!

**Remember to give peer feedback on data projects by April 21st.**