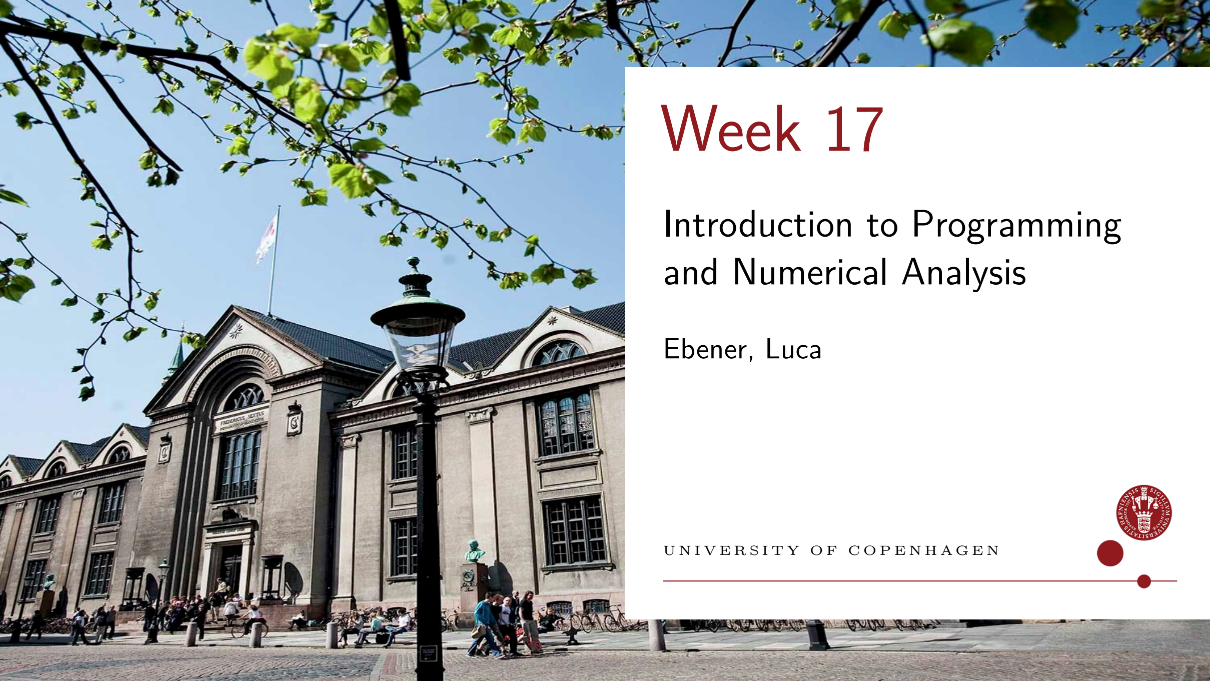

Week 17

Introduction to Programming and Numerical Analysis

Ebener, Luca

UNIVERSITY OF COPENHAGEN

Overview

- Work on problem set 7

Optimization Problem I

- A
  - Have a look at the notebook on unconstraint optimization part 3 for guidance for the 3D and contour plot
  - You don't have to guess the same levels as the answer in the contour plot but the way that it is done is pretty smart
- B
  - Lambdify the derivatives and collect them in numpy arrays
  - The Hessian needs to be a 2x2 array of the form `np.array([[f11,f12][f21,f22]])`
- C
  - Check the notebook for similar code and the documentation for more details about what is going on under the hood for different optimizers
  - Using analytical gradients typically saves computing power because of fewer iterations and evaluations, but requires some computations in sympy or by yourself
  - The notebook includes a summary of pros and cons of different scipy optimizers

Optimization Problem II

- A
  - You can mostly reuse code from the previous problem, loop through the different starting points, marking the optimization if it is better than the previous best
  - Also store all optimizations and results
- B
  - 3D scatter plots are a lot like plotting surfaces like you already did, but instead of: `cs = ax.plot_surface(x1_grid,x2_grid,f_grid,cmap=cm.jet)`, and the lines creating the grids, just write: `cs = ax.scatter(xs[:,0],xs[:,1],fs,c=fs)`
- C
  - Here, you just plot x0s instead of xs

Solve the consumer problem with income risk I

- A
  - Make sure you understand the functions for solving the problem that are given
  - You also need to create the v2_interp argument using the scipy.interpolate function RegularGridInterpolator, by solving period 2 first as in the notebook on dynamic optimization
- B
  - You have to define a new v1-function
  - This can be interpreted as higher risk in future periods increasing savings

Solve the consumer problem with income risk II

- Depending on the power of your computer, the `solve_period_2()`-function might run a bit slowly, you can turn down precision (replace 200 with 100 for example) if it becomes a problem
- The RegularGridInterpolator works for functions with multiple arguments also. It is fine to check the solution for the correct syntax. The x-values are a list of the two arguments as vectors, while the function-values should be in a grid format
- It is not explicitly noted, but $d_1 = d_2$, as it is durable consumption without depreciation
- Scipy might give you the warning `RuntimeWarning: Values in x were outside bounds during a minimize step, clipping to bounds warnings.warn("Values in x were outside bounds during a "`. This seems to be an error with the scipy package, so don't worry to much about it