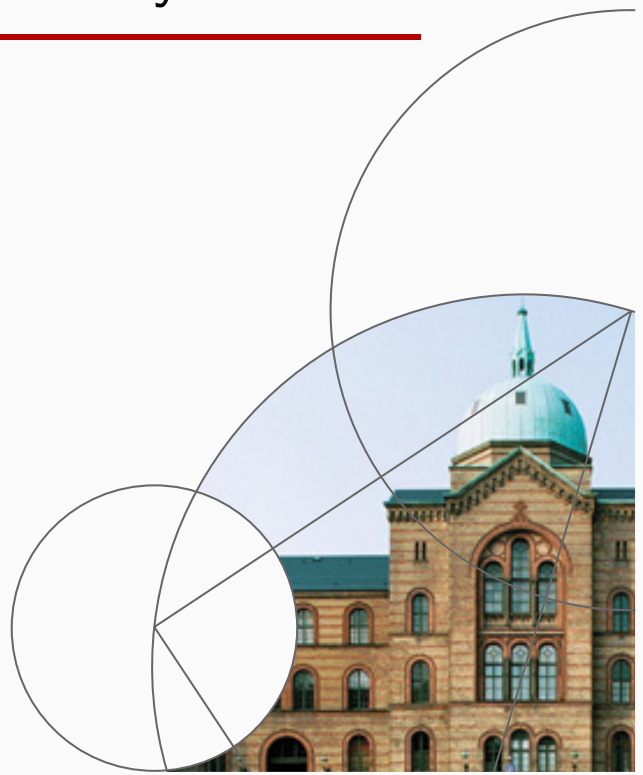
CENTER FOR
ECONOMIC
BEHAVIOR &
INEQUALITY

# 0. Introduction

## Introduction to Programming and Numerical Analysis

Jeppe Druedahl
Asker Nygaard Christensen



# Plan

1. Intended learning goals
2. Numerical analysis in action
3. Infrastructure
4. Work-flow
5. Projects
6. Summing up

# Intended learning goals

# Intended learning goals

- **In a nutshell:** *Learn how to use numerical analysis to improve your understanding of economic problems*
  1. Visualize solutions and simulations of well-known models
  2. Explore alternative assumptions regarding functional forms and parameter choices
  3. Solve more realistic models with constraints, uncertainty and non-convexities, where algebraic solutions are not available
  4. Work with online data and do programming based statistics and descriptive economics

# Intended learning goals

- **In a nutshell:** *Learn how to use numerical analysis to improve your understanding of economic problems*
  1. Visualize solutions and simulations of well-known models
  2. Explore alternative assumptions regarding functional forms and parameter choices
  3. Solve more realistic models with constraints, uncertainty and non-convexities, where algebraic solutions are not available
  4. Work with online data and do programming based statistics and descriptive economics
- Focus will be on **methods** rather than **economics**

  $\Rightarrow$ very relevant when writing your bachelor and master theses

  $\Rightarrow$ very relevant when using in your work-life

# Intended learning goals

- **In a nutshell:** *Learn how to use numerical analysis to improve your understanding of economic problems*
  1. Visualize solutions and simulations of well-known models
  2. Explore alternative assumptions regarding functional forms and parameter choices
  3. Solve more realistic models with constraints, uncertainty and non-convexities, where algebraic solutions are not available
  4. Work with online data and do programming based statistics and descriptive economics
- Focus will be on **methods** rather than **economics**
  $\Rightarrow$ very relevant when writing your bachelor and master theses
  $\Rightarrow$ very relevant when using in your work-life
- You will learn a **set of important tools**, but it is equally important that you **learn how to acquire new tools** for problems you will face in the future (in your studies or work-life)

# Scientific programming

- **Numerical analysis** is a ***complement*** not a substitute for **math**

# Scientific programming

- **Numerical analysis** is a ***complement*** not a substitute for **math**
- **Three central steps:**
  1. mathematical problem $\rightarrow$ construct algorithm
  2. algorithm $\rightarrow$ write code
  3. write code $\rightarrow$ present results

# Scientific programming

- **Numerical analysis** is a ***complement*** not a substitute for **math**
- **Three central steps:**
  1. mathematical problem $\rightarrow$ construct algorithm
  2. algorithm $\rightarrow$ write code
  3. write code $\rightarrow$ present results
- **The set of potential errors is infinite:**
  A good work-flow is very important
  1. Clear structure reduces the number of bugs
  2. Testing helps discovering bugs
  3. Documentation helps removing bugs

# Scientific programming

- **Numerical analysis** is a ***complement*** not a substitute for **math**
- **Three central steps:**
  1. mathematical problem $\rightarrow$ construct algorithm
  2. algorithm $\rightarrow$ write code
  3. write code $\rightarrow$ present results
- **The set of potential errors is infinite:**
  A good work-flow is very important
  1. Clear structure reduces the number of bugs
  2. Testing helps discovering bugs
  3. Documentation helps removing bugs
- **Programming is more than writing code:** Structuring, testing, documenting and collaborating on code is a central aspect of this course

# Active learning

- **Active learning:** To learn scientific programming you need to work on actual problems yourself
  - We can show you examples
  - We can guide you in terms of where to start
  - We can answer questions
  - But you need to work with the material on your own
  - Programming is not a spectator sport!

# Active learning

- **Active learning:** To learn scientific programming you need to work on actual problems yourself
  - We can show you examples
  - We can guide you in terms of where to start
  - We can answer questions
  - But you need to work with the material on your own
  - Programming is not a spectator sport!
- **High level:** Few (if any) econ bachelor programs provide education on numerical analysis on the level you will get

# Active learning

- **Active learning:** To learn scientific programming you need to work on actual problems yourself
  - We can show you examples
  - We can guide you in terms of where to start
  - We can answer questions
  - But you need to work with the material on your own
  - Programming is not a spectator sport!
- **High level:** Few (if any) econ bachelor programs provide education on numerical analysis on the level you will get
- **Work-in-progress:** All of your feedback is very important for optimizing and improving the course!

# Your teachers

- **Jeppe Druedahl**, Associate Professor
  research: macro questions, micro data, computational methods
  web: sites.google.com/view/jeppe-druedahl/
  e-mail: jeppe.druedahl@econ.ku.dk

- **Asker Nygaard Christensen**, PhD student
  e-mail: anc@econ.ku.dk

# Numerical analysis in action

# Numerical analysis in action

- We work with **Python 3.9**
- **Suggested environment:**
  1. **Distribution:** Anaconda
  2. **Editor/IDE**: VSCode
- **I will show** how to use VSCode
  1. Run python *code* and *notebooks*
  2. Solve the consumer problem from microeconomics

# Infrastructure

# Getting started

- **Web-page:** The course is organized around https://sites.google.com/view/numeconcph-intro prog/home
- **DataCamp:** Online courses on Python (requires no installation)
  $\Rightarrow$ you get 6 months free access (see e-mail with details)
- **Installation of Python:** Follow the installation guide

# Lectures, classes and exam

- **Lectures:** 3 physical lectures (see calendar)

  **Videos:** https://www.youtube.com/@numeconcph/

  **Questions?** Ask them as a Github-issue

- **Classes:** Week 6 to 20

# Lectures, classes and exam

- **Lectures:** 3 physical lectures (see calendar)

  **Videos:** https://www.youtube.com/@numeconcph/

  **Questions?** Ask them as a Github-issue

- **Classes:** Week 6 to 20
- **Exam requirements** (see deadlines)
  1. Basic programming test (on DataCamp.com, see e-mail)
  2. Inaugural project + 2x useful peer feedback
  3. Data analysis project + 2x useful peer feedback
  4. Model analysis project + 2x useful peer feedback
- **Exam:** Portfolio of projects + exam problem (48 hours)
- **Grading:** Pass or fail
- **Groups:** All projects can be done in *fixed* groups (maximum of 3)

# Course plan - lectures

Four parts:

1. **Fundamentals** (primitives, optimize, print and plot, random numbers and simulation, structure and documentation, workflow and debugging)
2. **Working with data** (load/save and structure data, basic data analysis)
3. **Algorithms** (searching and sorting, solving equations, numerical optimization)
4. **Further perspectives** (canonical economic models, structural estimation, speed-up with comprehensions, generators, vectorization and parallization, numba, EconModelClass, BabyMAKRO)

# Course plan - classes

1. DataCamp
2. DataCamp
3. DataCamp
4. Problem Set 1: Solving the consumer problem
5. Problem Set 2: Finding the Walras equilibrium in a multi-agent economy
6. Work on your inaugural project
7. Problem Set 3: Loading and combining data from Denmark Statistics
8. Problem Set 4: Analyzing data
9. Work on your data project
10. Problem Set 5: Writing your own searching and sorting algorithms
11. Problem Set 6: Solving the Solow model
12. Problem Set 7: Solving the consumer problem with income risk
13. Work on your model analysis project
14. Work on your model analysis project
15. Feedback on model project

# GitHub.com (code hosting platform)

- **All course materials will be shared on GitHub**
- **Organization:** www.github.com/NumEconCopenhagen

  **Repositories:**

  1. **IntroProg-lectures**: slides, course plan, guides etc.
  2. **IntroProg-exercises:** problem sets, solutions etc.

- **Git:** A version-control system for tracking changes in files and coordinating work $\Rightarrow$ integrated in VSCode

# Download course content guide

1. Follow the installation guide
2. Open *VScode*
3. Pres *Ctrl+Shift+P* to *command control palette*
4. Write *Git: Clone*
5. Use *https://github.com/NumEconCopenhagen/IntroProg-lectures*
6. Repeat with *https://github.com/NumEconCopenhagen/IntroProg-exercises*
7. Create copies of the folder to work in
8. You can update later with *Git: Sync*

# Work-flow

# Your work-flow

- **Lectures:**
  1. Watch videos and *try out the code yourself*
  2. Attend physical lectures and *participate actively*

# Your work-flow

- **Lectures:**
  1. Watch videos and *try out the code yourself*
  2. Attend physical lectures and *participate actively*
- **Classes:** Work on problem sets and ask questions to your fellow students and the teaching assistants
  1. Solve tasks and problems
  2. Fill the missing code
  3. Find the error
  4. Solve full problem

  **Note:** OK to peak at answers, but write the solution yourself

# Your work-flow

- **Lectures:**
  1. Watch videos and *try out the code yourself*
  2. Attend physical lectures and *participate actively*
- **Classes:** Work on problem sets and ask questions to your fellow students and the teaching assistants
  1. Solve tasks and problems
  2. Fill the missing code
  3. Find the error
  4. Solve full problem

  **Note:** OK to peak at answers, but write the solution yourself
- **In between classes and lectures:**
  1. Go through lecture notebooks (curriculum)
  2. Solve the problem set
  3. Experiment with your own ideas

# Getting help

- **Observation:** Programming is the slow and painful removal of tiny errors in your code – one at a time

# Getting help

- **Observation:** Programming is the slow and painful removal of tiny errors in your code – one at a time
- **Everybody often forgets the correct syntax** $\Rightarrow$ trial-and-error and testing is central, never a single correct approach

# Getting help

- **Observation:** Programming is the slow and painful removal of tiny errors in your code – one at a time
- **Everybody often forgets the correct syntax** $\Rightarrow$ trial-and-error and testing is central, never a single correct approach
- **Ask questions!!** In the following order
  1. Look in the documentation
  2. Talk about it in your group
  3. Search Google + Stackoverflow + ChatGPT
  4. Ask questions online using GitHub issues in IntroProg-lectures

# Getting help

- **Observation:** Programming is the slow and painful removal of tiny errors in your code – one at a time
- **Everybody often forgets the correct syntax** $\Rightarrow$ trial-and-error and testing is central, never a single correct approach
- **Ask questions!!** In the following order
  1. Look in the documentation
  2. Talk about it in your group
  3. Search Google + Stackoverflow + ChatGPT
  4. Ask questions online using GitHub issues in IntroProg-lectures
- **Help each other!!** You will learn a lot.
  Remember to be constructive and polite!

# Projects

# Basic programming test

- **You must complete the following courses on DataCamp**
  1. Introduction to Data Science in Python
  2. Intermediate Python
  3. Python Data Science Toolbox (Part 1)
  4. Python Data Science Toolbox (Part 2)
- **First 3 exercise classes:** Reserved for your work on DataCamp

# Inaugural project

- **Objectives:**
  1. Apply simple numerical solution methods
  2. Structure a code project
  3. Document code
  4. Present results
  5. Use GitHub
- **Content:**
  1. Solution of pre-specified economic model
  2. Visualization of solution
- **Structure:**
  1. A self-contained single notebook presenting the analysis
  2. Fully documented python files
- **Hand-in:** Create and commit folder called "inauguralproject" in your GitHub repository

# Data analysis project

- **Objectives:**
  1. Apply data cleaning and data structuring methods
  2. Apply data analysis methods
  3. Structure a code project
  4. Document code
  5. Present results in text form and in figures
- **Content:**
  1. Import data from an online source
  2. Present the data visually (and perhaps interactively)
  3. Apply some method(s) from descriptive economics (»samfundsbeskrivelse«)
- **Structure:**
  1. A self-contained single notebook presenting the analysis
  2. Fully documented python files
- **Hand-in:** Create and commit folder called "dataproject" in your GitHub repository

# Model analysis project

- **Objectives:**
  1. Apply model analysis methods
  2. Structure a code project
  3. Document code
  4. Present results in text form and in figures
- **Content:**
  1. Describe an algorithm on how to solve a simple economic model
  2. Solve (and perhaps simulate) a simple economic model
  3. Visualize results across e.g. parametrizations
  4. Analyze one or more extensions of the baseline model
- **Structure:**
  1. A self-contained single notebook presenting the analysis
  2. Fully documented python files
- **Hand-in:** Create and commit folder called "modelproject" in your GitHub repository

# Summing up

# Summing up

- **I hope your have:**
  1. An idea of why learning numerical analysis is important
  2. What you will learn in this course
  3. How you will learn it by working actively and interact with your fellow students
  4. How you will qualify for and pass the exam

# Your to-do list

1. **First priority:** Login to DataCamp.com (see info in e-mail)

# Your to-do list

1. **First priority:** Login to DataCamp.com (see info in e-mail)
2. **Second priority:** Install **Anaconda**, **Git** and **VSCode** (see the installation guide)

# Your to-do list

1. **First priority:** Login to DataCamp.com (see info in e-mail)
2. **Second priority:** Install **Anaconda**, **Git** and **VSCode** (see the installation guide)
3. **Third priority:** Download slides and exercises with git (see »Download course content guide«-slide)

# Your to-do list

1. **First priority:** Login to DataCamp.com (see info in e-mail)
2. **Second priority:** Install **Anaconda**, **Git** and **VSCode** (see the installation guide)
3. **Third priority:** Download slides and exercises with git (see »Download course content guide«-slide)
4. **Fourth priority:** Run the example code from this lecture yourself