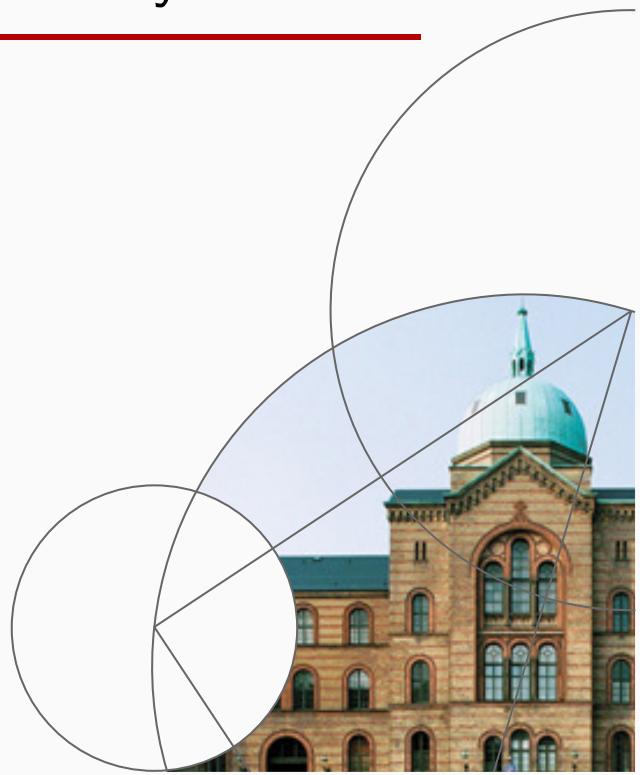
CENTER FOR
ECONOMIC
BEHAVIOR &
INEQUALITY

1. Introduction

Introduction to Programming and Numerical Analysis

Jeppe Druedahl

Spring 2020



Plan

1. Intended learning goals
2. Numerical analysis in action
3. Infrastructure
4. Work-flow
5. Projects
6. More examples
7. 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
  - I can show you examples
  - I can guide you in terms of where to start
  - I 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
  - I can show you examples
  - I can guide you in terms of where to start
  - I 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
  - I can show you examples
  - I can guide you in terms of where to start
  - I 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
- **Second generation:** All of your feedback is very important for optimizing and improving the course!

Who I am

- **Name:** Jeppe Druedahl (ph.d. polit)
- **Web-page:** www.econ.ku.dk/druedahl
- **Position:** Assistant Professor at Department of Economics, Center for Economic Behavior and Inequality (CEBI)
- **Research interests:**
  1. Macro-questions
  2. Micro-data
  3. Numerical methods
- **Modern macro-models**
  1. Heterogeneous agents (households and firms) take decisions under uncertainty and imperfect information
  2. Markets are not complete
  3. The dynamic equilibrium path is found approximately on a (large) computer

Who you are

[results from questionnaire]

Numerical analysis in action

Numerical analysis in action

- We work with **Python 3.7**
- **Suggested environment:**
  1. **Distribution:** Anaconda
  2. **Documents:** JupyterLab
  3. **Editor/IDE**: VSCode
- **I will show** how to
  1. Run Python in JupyterLab
  2. Solve the consumer problem from microeconomics

Infrastructure

Getting started

- **Web-page:** The course is organized around
  www.numeconcopenhagen.netlify.com
  [copy of all material on Absalon...]
- **DataCamp:** Online courses on Python (requires no installation)
  $\Rightarrow$ you get 6 months free access (see e-mail with details)
- **Install and run Python:** Follow these guides
  1. Installing Python and VSCode
  2. Running Python in JupyterLab
  3. Running Python in VSCode

Lectures, classes and exam

- **Lectures and classes:**
  1. **Lectures:** Monday 15-17
  2. **Classes:** Tuesday/Wednesday 15-17
- **Exam requirements** (deadlines):
  1. Basic programming test (on DataCamp.com, see e-mail)
  2. Inaugural project
  3. Data analysis project
  4. 2x useful peer feedback on data analysis projects
  5. Model analysis project
  6. 2x useful peer feedback on model analysis projects
- **Exam:** Portfolio of projects + exam problem (48 hours)
- **Grading:** Pass or fail
- **Groups:** All projects can be done in *fixed* groups (maximum of 4)

Course plan - lectures

1. Introduction
2. Fundamentals: Primitives
3. Fundamentals: Optimize, print and plot
4. Fundamentals: Random numbers and simulation
5. Fundamentals: Workflow and debugging
6. Fundamentals: Recap and overview
7. Working with Data: Load/save and structure data
8. Working with Data: Basic data analysis
9. Algorithms: Searching and sorting
10. Algorithms: Solving equations
11. Algorithms: Numerical optimization
12. Further Perspectives: The need for speed
13. Further Perspectives: Other programming languages

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. **lectures-2020**: slides, course plan, guides etc.
  2. **exercises-2020:** problem sets, solutions etc.

- **Git:** A version-control system for tracking changes in computer files and coordinating work on those files among multiple people.
  $\Rightarrow$ integrated in VSCode
  $\Rightarrow$ we will talk more about it in week 5

Download guide

1. Follow the installation guide
2. Open VScode
3. Pres »Ctrl+Shift+P«.
4. Write: »git: clone«
5. Write »https://github.com/NumEconCopenhagen/lectures-2020«
   Or: »https://github.com/NumEconCopenhagen/exercises-2020«
   $\Rightarrow$ the repository will be downloaded to your computer
6. You can update to the newest version of the code with »git: sync«
7. Create a copy of the cloned folder, where you work with the code (otherwise you can not sync with updates)

Alternative: Use the download button on the github-page.

Work-flow

Your work-flow

- **Lectures:** *Listen to me and ask questions on Socrative*
  1. Overview of topic
  2. Introduction to new concepts
  3. Live coding
  4. Presentation of problem set

Your work-flow

- **Lectures:** *Listen to me and ask questions on Socrative*
  1. Overview of topic
  2. Introduction to new concepts
  3. Live coding
  4. Presentation of problem set
- **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:** *Listen to me and ask questions on Socrative*
  1. Overview of topic
  2. Introduction to new concepts
  3. Live coding
  4. Presentation of problem set
- **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

Socrative

- **Socrative:**
  1. Web: www.socrative.com → student login → room: NumEcon
  2. App (Socrative Student): room: NumEcon
- **Structure:**
  1. 5 min to run code just presented
  2. Ask question on Socrative
  3. Vote on most important question

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 [see this guide]
  4. Ask question online using GitHub issues [see this guide, example]

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 [see this guide]
  4. Ask question online using GitHub issues [see this guide, example]
- **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. Intro to Python for Data Science
  2. Intermediate Python for Data Science
  3. Python Data Science Toolbox (Part 1)
  4. Python Data Science Toolbox (Part 2)
- **First 3 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

More examples

More examples

- **I will show** how to
  1. Simulate the AS-AD model
  2. Write modules in VSCode
  3. Run Python code in VSCode

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 (see info in e-mail)

Your to-do list

1. **First priority:** Login to DataCamp (see info in e-mail)
2. **Second priority:**
   - 2.1 Install **Anaconda with VSCode**
     See the guide: Installing Python and VSCode
   - 2.2 Open **JupyterLab** and experiment
     See the guide: Running Python in JupyterLab

Your to-do list

1. **First priority:** Login to DataCamp (see info in e-mail)
2. **Second priority:**
   - 2.1 Install **Anaconda with VSCode**
     See the guide: Installing Python and VSCode
   - 2.2 Open **JupyterLab** and experiment
     See the guide: Running Python in JupyterLab
3. **Third priority:** Download slides and exercises with git
   (see previous »Download guide«-slide)

Your to-do list

1. **First priority:** Login to DataCamp (see info in e-mail)
2. **Second priority:**
   - 2.1 Install **Anaconda with VSCode**
     See the guide: Installing Python and VSCode
   - 2.2 Open **JupyterLab** and experiment
     See the guide: Running Python in JupyterLab
3. **Third priority:** Download slides and exercises with git (see previous »Download guide«-slide)
4. **Fourth priority:** Read the guides on:
   - Searching for answers using Google and Stackoverflow
   - Asking questions using GitHub issues

Your to-do list

1. **First priority:** Login to DataCamp (see info in e-mail)
2. **Second priority:**
   - 2.1 Install **Anaconda with VSCode**
     See the guide: Installing Python and VSCode
   - 2.2 Open **JupyterLab** and experiment
     See the guide: Running Python in JupyterLab
3. **Third priority:** Download slides and exercises with git (see previous »Download guide«-slide)
4. **Fourth priority:** Read the guides on:
   - Searching for answers using Google and Stackoverflow
   - Asking questions using GitHub issues

- **Next time:** Introduction to the fundamentals of Python