Introduction

Maksim Bolonkin

Moscow State University

Few minutes of honesty

- Course compiled from scratch $\rightarrow$ inherent incoherency
- Diverse group of students $\rightarrow$ possibly redundant topics
- Not enough time to rewrite slides $\rightarrow$ using slides from other sources (in English)
- Trying my best $\rightarrow$ expecting same from you
- Putting slide credits wherever possible (some sources might be lost)

Course outline

"Computational and numerical tools" (known as *"Optimization and numerical methods"* in your curriculum)

What this course is about:

- Some of the most relevant computational and numerical tools commonly used in research and engineering
- Tools = algorithms and implementations

Course outline

"Computational and numerical tools" (known as *"Optimization and numerical methods"* in your curriculum)

What this course is about:

- Some of the most relevant computational and numerical tools commonly used in research and engineering
- Tools = algorithms and implementations

What this course is **not** about:

- Minimal math component: some math for understanding, rare derivations, almost no analysis/proofs
- Not a programming course: you are expected to know basic programming
- Breadth, not depth

Logistics

This course consists of **lectures and programming seminars**. Usualy one lecture and one seminar will be every week with rare exceptions. In addition to that this year I plan to have **research presentation** as part of the class.

Lecture notes and video links will be posted online:
`https://www.mbolonkin.info/teaching/cs170-2022`

Homeworks: basic and advanced version will be posted online
Research presentation: papers will be posted online

Getting your grade:

- Grade "5"
  + Solve the advanced homework (automatic grade) OR
  + Solve the basic homework (80%) and attend the oral examination
  + Do a good research presentation and be be able to answer quesions
- Grade "4"
  + Solve the basic homework (80%) and attend the oral examination
  + Do a decent research presentation
- Grade "3"
  + Attend the classes and solve the basic homework (50%)
  + Do a research presentation
- Non-passable grade
  - Attendance below threshold, no solved homeworks, failed oral exam
  - Severe violation of an honor principle

Honor principle

Any work you are submitting must be your own. Collaboration without sharing any code is acceptable.

In case I am **suspicious** of plagiarism in your assignments (in full or partialy, based on in-person grading), I reserve the right to not accept your solutions as well as give you a non-passable grade for the course. Plagiarized work will not be counted towards any positive grades.

Burden of proof is entirely on you: I do not have to prove that your work is plagiarized, you have to prove it is not.

Corollary: In case you decided to plagiarize your peer's solution, make sure you understand it thoroughly.

Expected programming tools for this course:

- Matlab/Octave
- Python + Jupyter Notebooks (Recommended)

I have Anaconda packages for Windows downloaded.

Why Python?

Python has well supported scientific libraries

- NumPy - supports wide variety of matrix operations
- SciPy - a lot of computational methods implemented
- MatPlotLib - library for plotting and visualizing
- Scikit-Image, Scikit-Learn, etc.
- Very good documentation

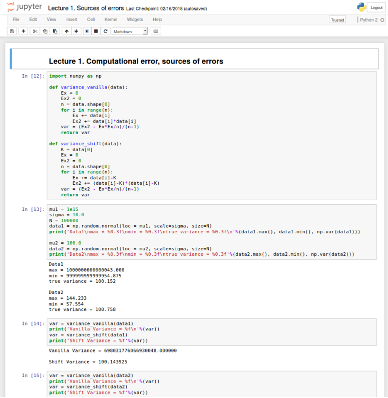
Why Python?

Python has well supported scientific libraries

- NumPy - supports wide variety of matrix operations
- SciPy - a lot of computational methods implemented
- MatPlotLib - library for plotting and visualizing
- Scikit-Image, Scikit-Learn, etc.
- Very good documentation

Jupyter Notebooks:

- Convenient tool for coding and debugging
- Good for reproducible research
- Getting more and more popular



Any questions?

Why bother?

- Computation allows exploring models without analytical solution
- This is usually the case for real-world problems
- Advances in hardware and software make it easier to use computational models

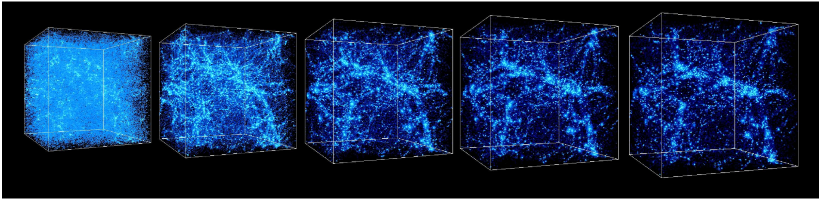
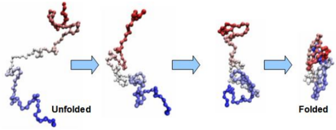


Figure: Galaxy formation (cosmicweb.uchicago.edu)

Why bother?



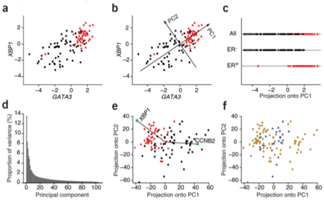


Figure: Protein unfolding simulation and analysis of gene expression

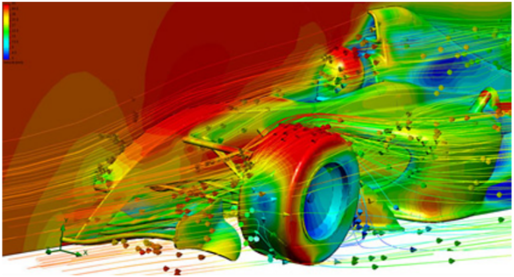


Figure: Fluid dynamics simulation

Top 10 Algorithms of 20th Century

In 2000 January-February issue of *Computing in Science and Engineering* editors selected 10 algorithms that influenced science and engineering in 20th century.

1. **Metropolis algorithm for Monte Carlo**

   1946: John von Neumann, Stan Ulam, and Nick Metropolis

2. **Simplex method for linear programming**

   1947: George Dantzig, at the RAND Corporation

3. **Krylov subspace iteration methods**

   1950: Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos

4. **The decompositional approach to matrix computations**

   1951: Alston Householder of Oak Ridge National Laboratory

5. **The Fortran optimizing compiler**

   1957: John Backus, IBM

6. **QR algorithm for computing eigenvalues**

   1959–61: J.G.F. Francis

7. **Quicksort algorithm for sorting**

   1962: Tony Hoare

8. **Fast Fourier transform**

   1965: James Cooley of the IBM T.J. Watson Research Center and John Tukey of Princeton University and AT&T Bell Laboratories

9. **Integer relation detection**

   1977: Helaman Ferguson and Rodney Forcade of Brigham Young University

10. **Fast multipole method**

   1987: Leslie Greengard and Vladimir Rokhlin of Yale University

In 2016 editors of the CiSE updated the list of top 10 algorithms, adding the following:

1. **Newton and quasi-Newton methods**
2. **Jpeg**
3. **PageRank**
4. **Kalman filter**

What we are going to cover in this course:

- **Data fitting**
  Some topics: linear least squares, non-linear least squares, maximum likelihood, maximum a posteriori, expectation maximization
- **Optimization**
  Some topics: gradient-free optimization, gradient descent, non-linear optimization, heuristics
- **Dimensionaity reduction and component analysis**
  Some topics: PCA, ICA, t-SNE, LDA
- **Signal Processing**
  Some topics: signals, filters, DFT
- **Randomized methods**
  Some topics: random number sampling, Monte-Carlo methods, RANSAC
- **Differential equations**
  Some topics: numerical solutions for Ordinary Differential Equations (ODE), Partial Differential Equations (PDE)

- Replacing difficult problem by simplified problem with the same or close enough solution

  Infinite $\rightarrow$ Finite
  Differential $\rightarrow$ Algebraic
  Continuous $\rightarrow$ Discrete
  Non-linear $\rightarrow$ Linear
  ...

- Solution obtained only approximates the exact solution of the original problem
  - Approximation before computation (empirical measurements, previous computation)
  - Approximation during computation (covered in next part)
  - Final result accuracy reflects all those errors, possibly amplified by an agorithm/problem