UPPSALA
UNIVERSITET

Department of Information Technology

Scientific Computing for Data Analysis

Davoud Mirzaei

Course Overview

Course overview

Teachers:

- Main teacher: Davoud Mirzaei, Associate Professor, Department of IT, Uppsala University
  (All lectures, 3 problem solving, Assignments, Exam, Grading)
  Email: davoud.mirzaei@it.uu.se

- TA: Aleksandr Karakulev, PhD student, IT Department, Uppsala University, Email: aleksandr.karakulev@it.uu.se
  (3 problem solving and 3 lab sessions, Assignments, Python)

Course Structure

The structure of the course:

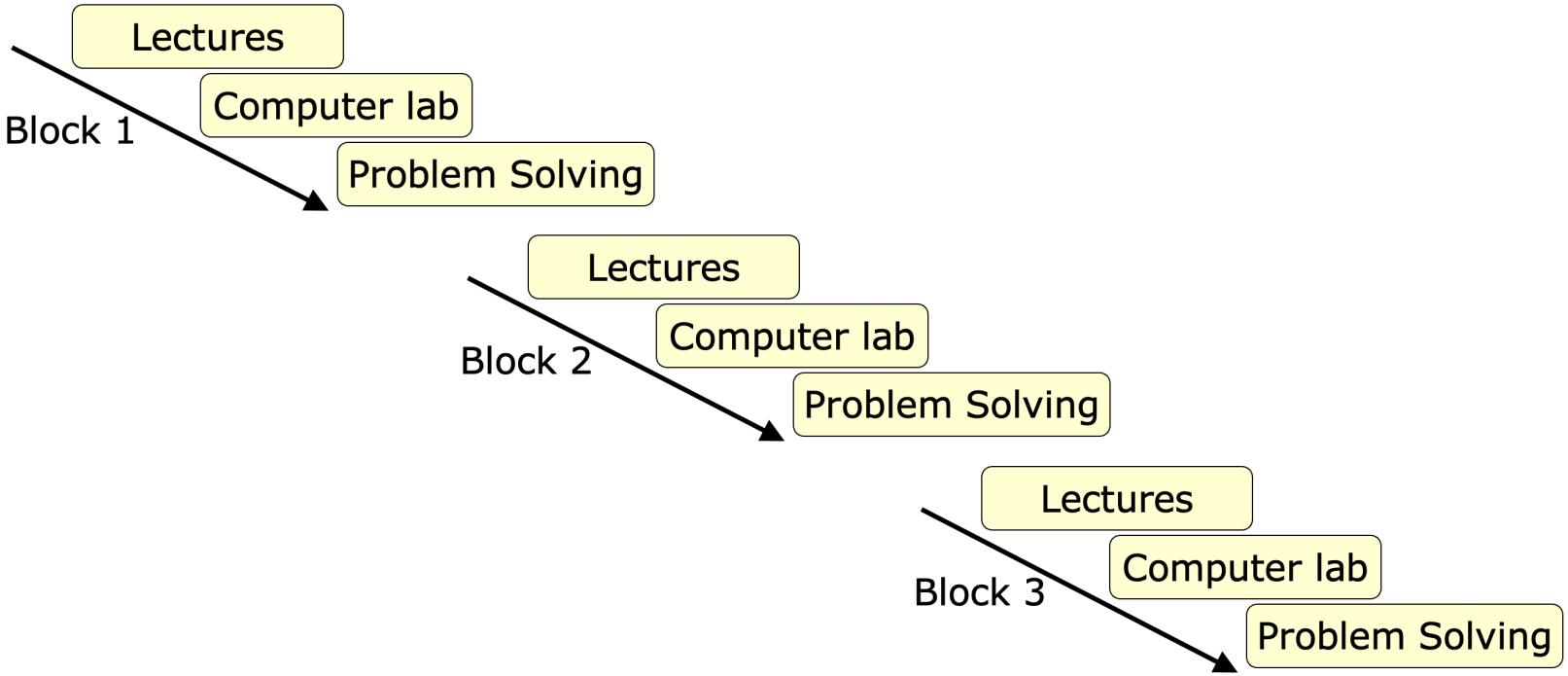
- Block 1: Stochastic simulation and Monte Carlo method: 4 lectures, 1 problem solving, and 1 lab exercise $\longrightarrow$ miniproject 1
- Block 2: Regression, Least squares problem, QR: 3 lectures, 1 problem solving, and 1 lab exercise
- Block 3: Eigenvalues and SVD: 3 lectures, 1 problem solving, and 1 lab exercise $\longrightarrow$ miniproject 2
- Exam preparation: 2 lectures

In each module we look at important computer methods/algorithms that are used and their characteristics

Download lecture presentations from Studium (will be posted gratually during the period)

Course Structure

Each block begins with 3-4 lectures at the lecture halls



All parts of a block are tightly linked together - the three parts together gives a “full” understanding

- Miniproject 1 in Block 1
- Miniproject 2 in Block 3

Learning outcomes (goals)

After the course the student should be able to:

- account for and perform tasks that require knowledge of the key concepts included in the course;
- describe, use and implement the algorithms included in the course;
- analyze the convergence and the computational cost of different algorithms;
- solve technical and scientific problems given the mathematical model, by structuring the problem, choosing the appropriate numerical method, and generating a solution using software and your own code;
- present, explain, summarize, evaluate and discuss solution methods and results in a small report;

Learning outcomes

Mainly tested on final exam:

You are supposed be able to understand, describe and use

- Important concepts
- Algorithms
- Analysis

Mainly tested on mini-project and problem-solving sessions:

Solve problems (coding included) and present the problem and solution in the class and in a report

Assessment

Examination:

- Coursework (2 credits): Miniprojects and Problem-Solvings Grade: **fail/pass**
- Final written exam (3 credits): 2024-03-08 Grade: **fail/3/4/5**

Important: The exam grade is solely based on the final written examination

Obligatory parts:

- Problem solving sessions (3 sessions: present your solutions at the end of each session)
- Mini-projects (2 mini-projects: 2-3 students in each group, register in mini-project groups as soon!)

Active participation in Lectures and Lab sessions is strongly recommended!

Exam Setup

When? Where? How?

- Exam date: 2024-03-08, Time: 14:00 - 17:00
- Exam hall: ...
- Exam in Inspera

Exam structure

- Part A: grade 3: usually 6 questions (2 questions per each objectives concepts, algorithms, analysis). You must collect points per each objective.
- Part B: higher grades: usually 3 questions from all objectives.

More detailed info concerning points will follow - this is the rough structure

*Last two lectures: Exam preparation

Problem Solving (PS)

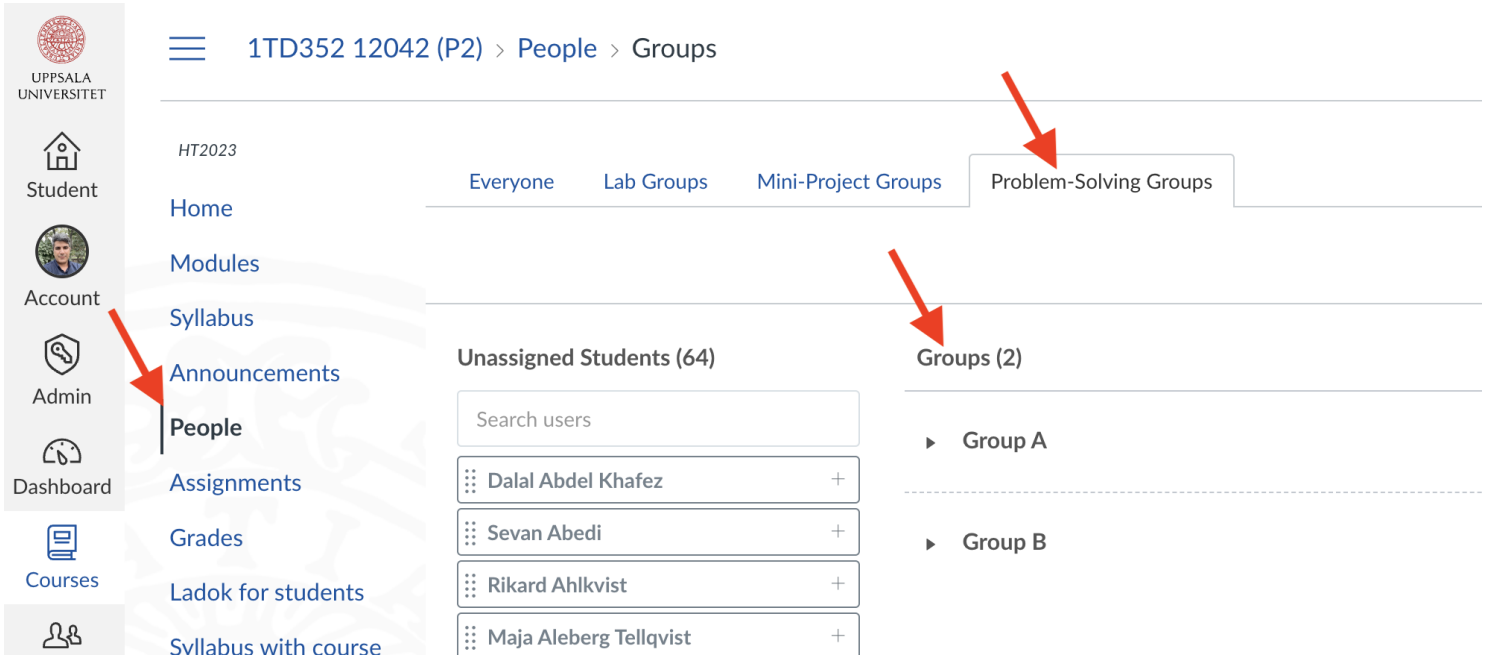
- 3 PS sessions, will be run in ALC
- It is a group work, group size: 4-7 students
- Participation is mandatory*
- You solve together 4-6 workouts during the session
- Nothing needs to be submitted but groups will present their solutions during the session
- PS1 (In Evelyn Sokolowski class, the same timeslot), PS2 and PS3 (in 2 ALCs, different timeslot)
- Choose a groups (A or B) in Studium for PS2 ans PS3 (limited to 50 students)

*If you have to miss a problem-solving class, upload the solutions to all mandatory problems no later than 2 days after the session (deadline is strict, then solutions will be released)

Problem Solving

To do now:

- Choose a PS group in Studium
  Problem Solving Group A
  Problem Solving Group B



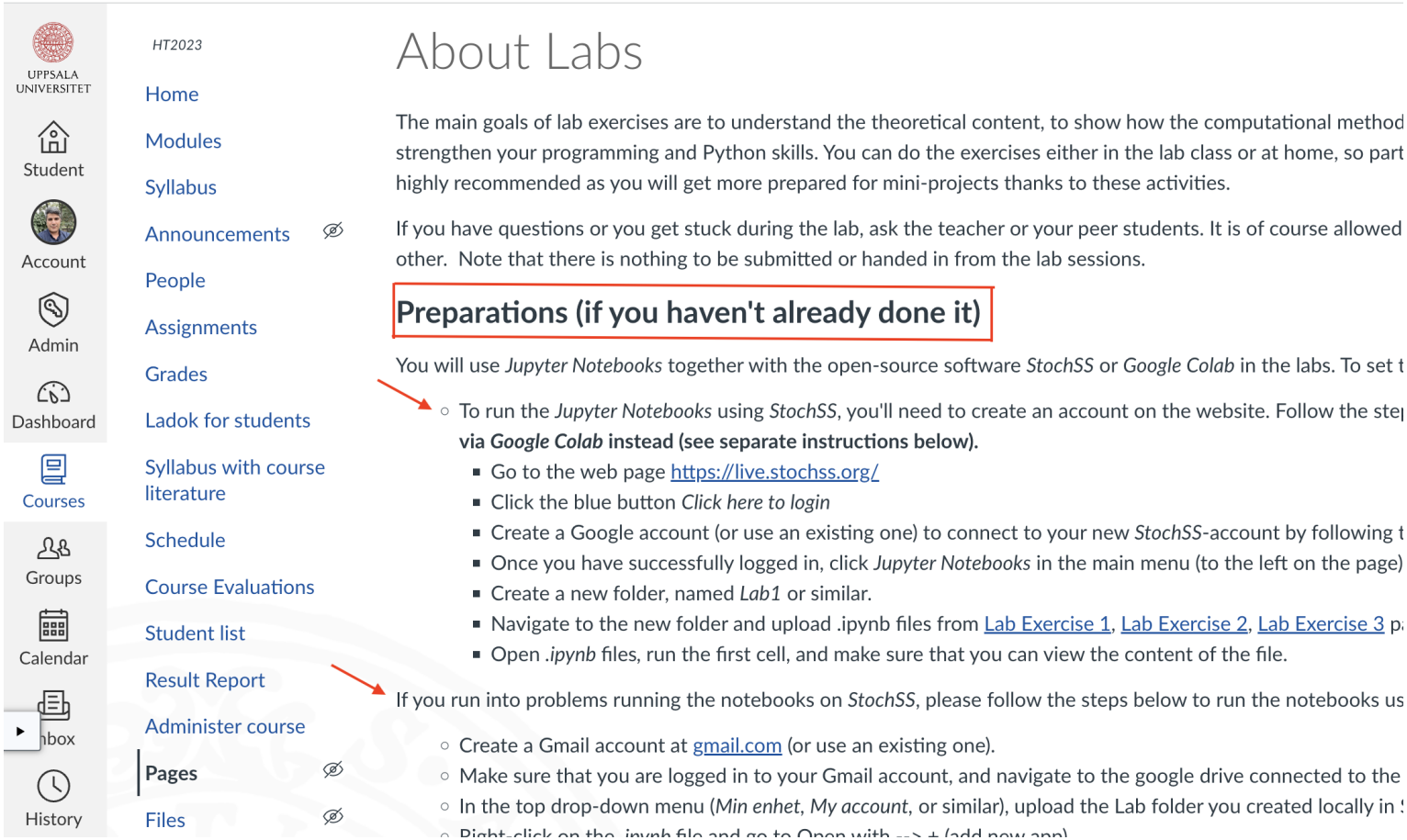
Computer Labs

Lab exercises:

- Important to prepare for the mini-project
- Not mandatory, but highly recommended to do
- Nothing needs to be submitted
- We will be using Python and Jupyter Notebook
- You can run the labs online using StochSS or Google Colab
- StochSS serves as a platform for stochastic biochemical simulations
- If preferred, you can run the labs on your own laptop by installing Jupyter Notebook
- Follow the instructions provided on the "About Labs" page in Studium for guidance

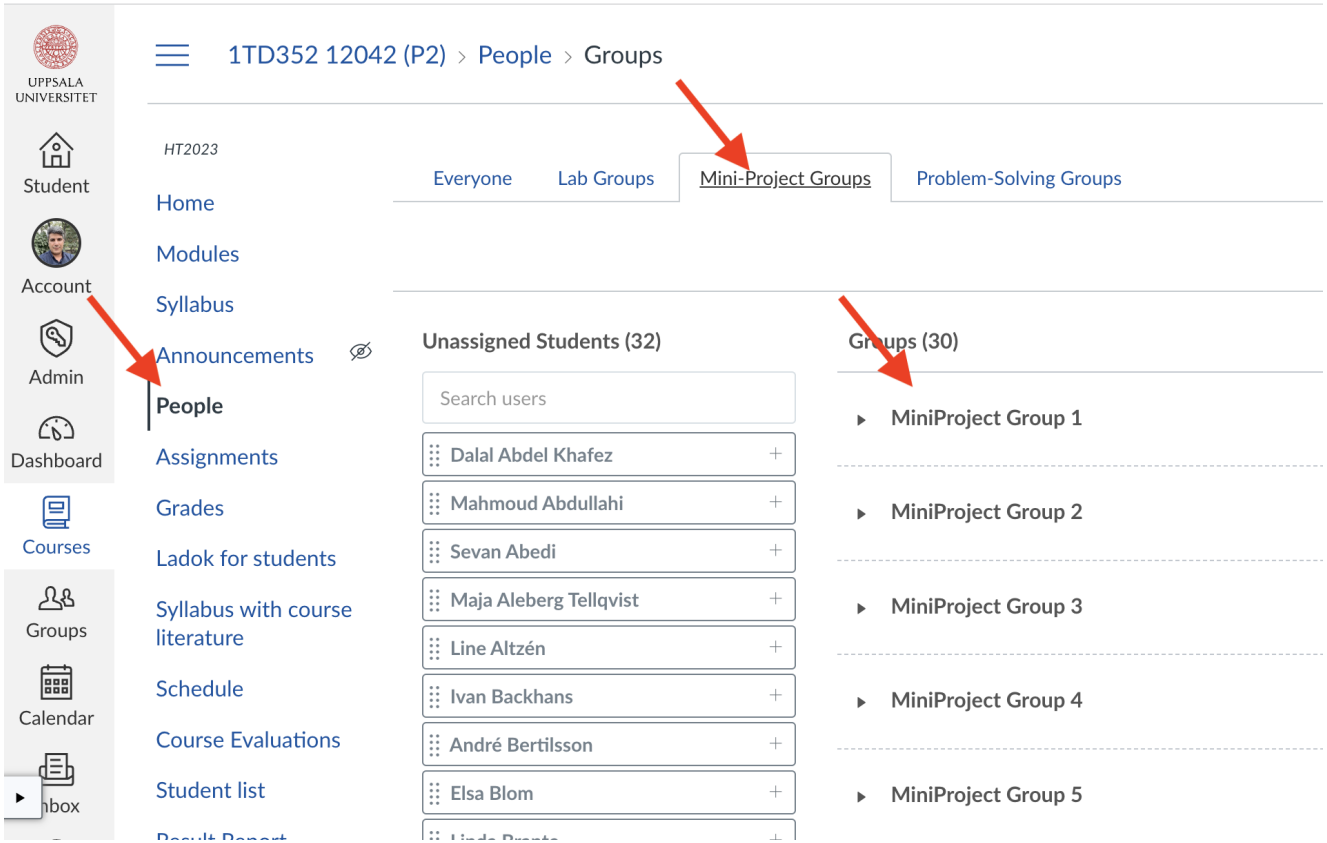
To do now:

- Do the lab preparation (set up StochSS etc.)



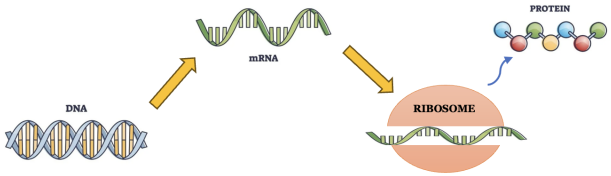
Mini-project organization

- You form groups of 2 or 3 students and you work together outside of class
- You register your group by Thu. 2024-01-25, otherwise we put you in random groups



Mini-Project

- We have a project in Block 1 (Monte Carlo) and a project in Block 3 (Classification of handwritten digits)

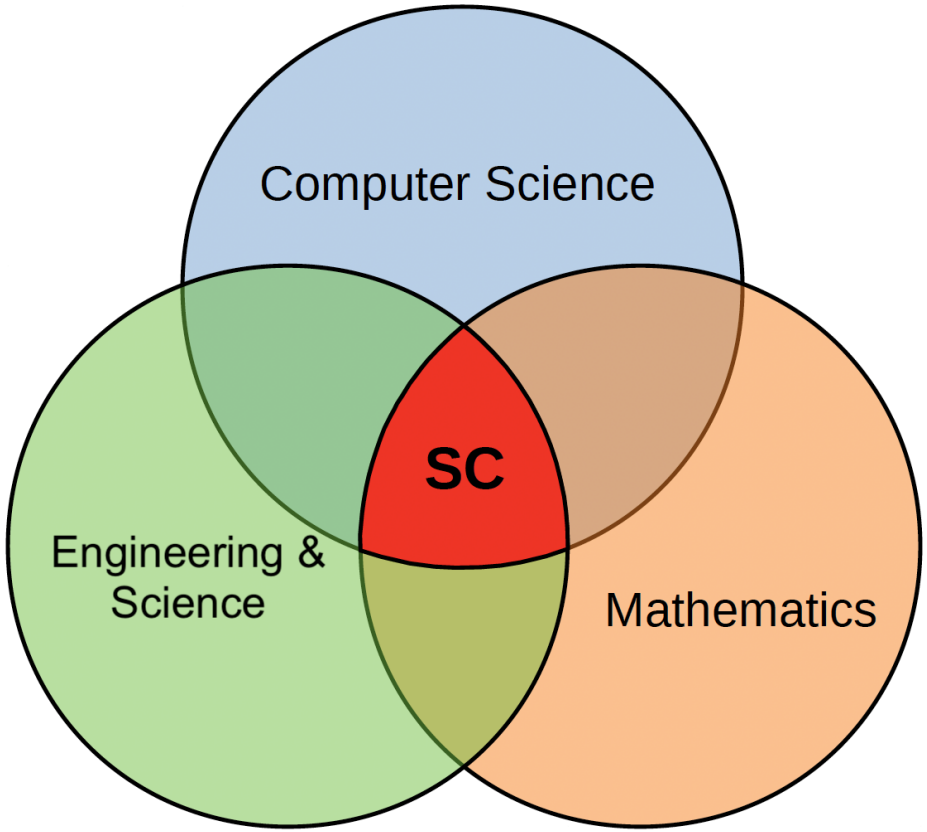


- Practice report writing (learning outcome)
- Peer review according to predefined categories (mandatory)
- Feedback from teachers
- You can use some part of codes from Labs
- The first deadline for each project is strict, if you miss it you will miss the peer review
- See the page "About miniprojects" in Studium

	Deadline	Feedback (peers and teacher)	Final submission
Mini-project 1	Feb. 5	Feb. 6--9	** Feb. 15
Mini-project 2	Feb. 26	Feb. 27--March 1	March 8

What is scientific computing?

What is scientific computing?



Scientific computing concerns with the **design and analysis of algorithms** for solving **mathematical problems** that arise in **science and engineering**

Problem Solving Process

Problem solving process consists of 4 steps:

1. Develop a mathematical model, usually expressed by equations of some type, of a physical phenomenon or system of interest
2. Discretize the model and develop algorithms to solve the equations numerically
3. Implement and execute the algorithms in computer software
4. Interpret and validate the computed results, repeating any or all of the preceding steps, if necessary.

Steps 2 and 3 (designing, analyzing, implementing, and using numerical algorithms and software) are the main subject matter of scientific computing.

Problem Solving Process (a funny example: birds on the wire)

modelling

$$y'' = f(x),\ 0 \leqslant x \leqslant \ell$$
$$y(0) = y(\ell) = 0$$

interpretation

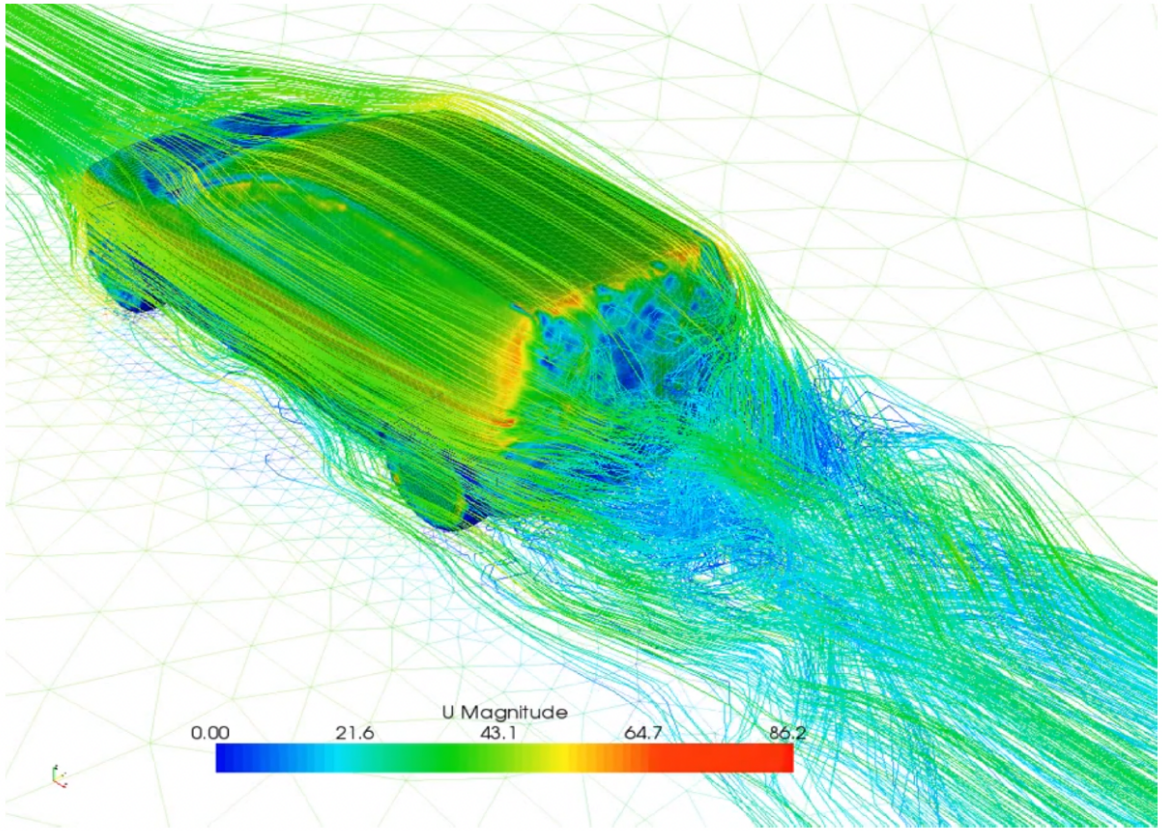
discretization

```
def dijkstra(graph,src,dest,visited=[],distances={},predecessors={}):
    """ calculates a shortest path tree routed in src
    """
    if src not in graph:
        raise TypeError('The root of the shortest path tree cannot be found')
    if dest not in graph:
        raise TypeError('The target of the shortest path cannot be found')
    if src == dest:
        path=[]
        pred=dest
        while pred != None:
            path.append(pred)
            pred=predecessors.get(pred,None)
            readable=path[0]
        for index in range(1,len(path)): readable = path[index]+'--->'+readable
        print('shortest path - array: '+str(path))
        print("path: "+readable+",   cost="+str(distances[dest]))
    else :
        if not visited:
            distances[src]=0
        for neighbor in graph[src] :
            if neighbor not in visited:
                new_distance = distances[src] + graph[src][neighbor]
                if new_distance < distances.get(neighbor,float('inf')):
                    distances[neighbor] = new_distance
                    predecessors[neighbor] = src
```

code implement.

$$y_{k+1} - 2y_k + y_{k-1} = h^2 f_k$$
$$y_0 = y_N = 0$$

A real example: Volvo car simulation



- The mathematical model: Navier-Stokes equations in fluid (air) and elasticity equations in solid (car body)
- Computational methods: FVM in fluid and FEM in solid
- A deep mathematical theory + high performance computing

Block 1
Stochastic Simulations

Lecture 1: Monte Carlo method - I

Agenda

- Deterministic vs. Stochastic
- Monte Carlo method
- Review of prior concept

Deterministic model/method

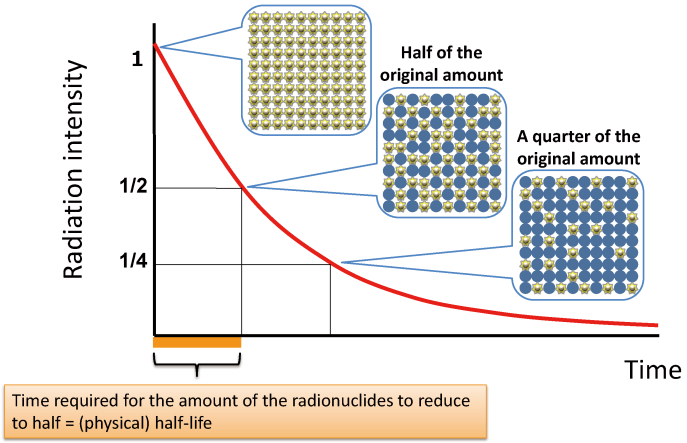
Determinism: “The theory that everything that happens must happen as it does and could not have happened any other way.”

Cambridge dictionary

- A **deterministic model** always yields the same result for a given set of initial conditions and parameters.
  Examples: Newtons laws on motion, everything described by an differential equation, ...
- A **deterministic method** produces the same results, given certain set of input data
  Examples: Euler’s method for ordinary differential equations, trapezoid and Simpson’s rules for integration, Newton-Raphson’s method for non-linear equations, ...

A deterministic model

An example: a deterministic model describing radioactive decay

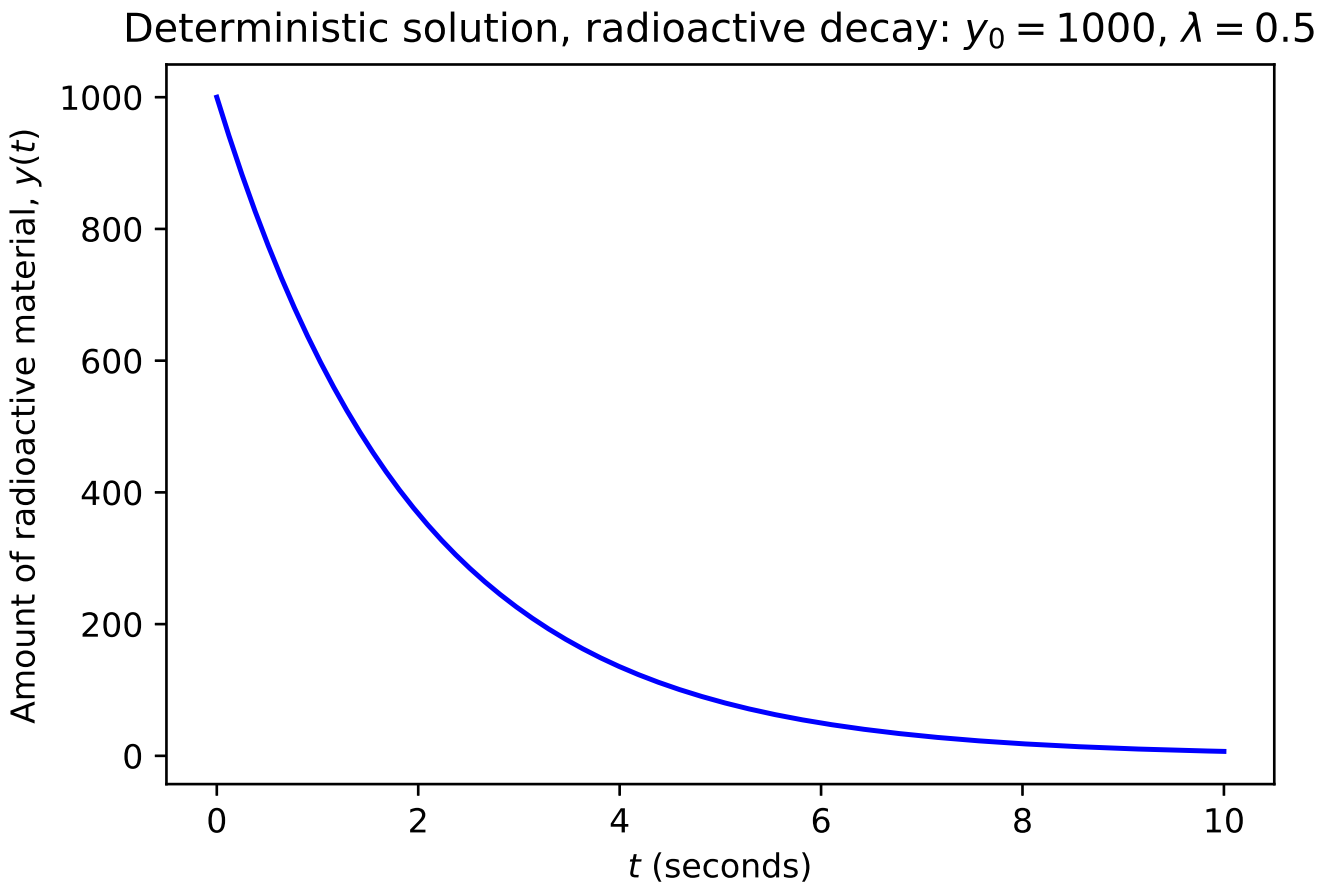


The model is an ordinary differential equation, ODE:

$$\begin{cases} y'(t) = -\lambda y(t) \\ y(0) = y_0 \end{cases}$$

- $y(t)$ amount of radioactive material at time t, λ constant rate (a positive constant depending on the radioactive substance).
- Analytic solution exist in this case: $y(t) = y_0 e^{-\lambda t}$
- Given a certain amount of material y_0 at time 0 and a certain rate λ we always get the same solution
- We can also solve this model using a numerical method (next page)

A deterministic method



Numerical solution using ODE-solver in Python, `scipy.integrate.solve` (Runge-Kutta 45) which is a deterministic method

The deterministic solution vs. reality:

- If we repeat this experiment with the same initial amount of radioactive material, would we observe exactly the same reductions at each time as before?
- Each individual decay happens randomly with a certain probability. Shouldn't the amount vary slightly?
- The unit is "amount of radioactive material", not number of particles. For example, at $t = 4s$ we have $y(4) = 135.3353$, but number of particles must be an integer
- The solution should be seen as an average that agrees with reality when we have large number of particles

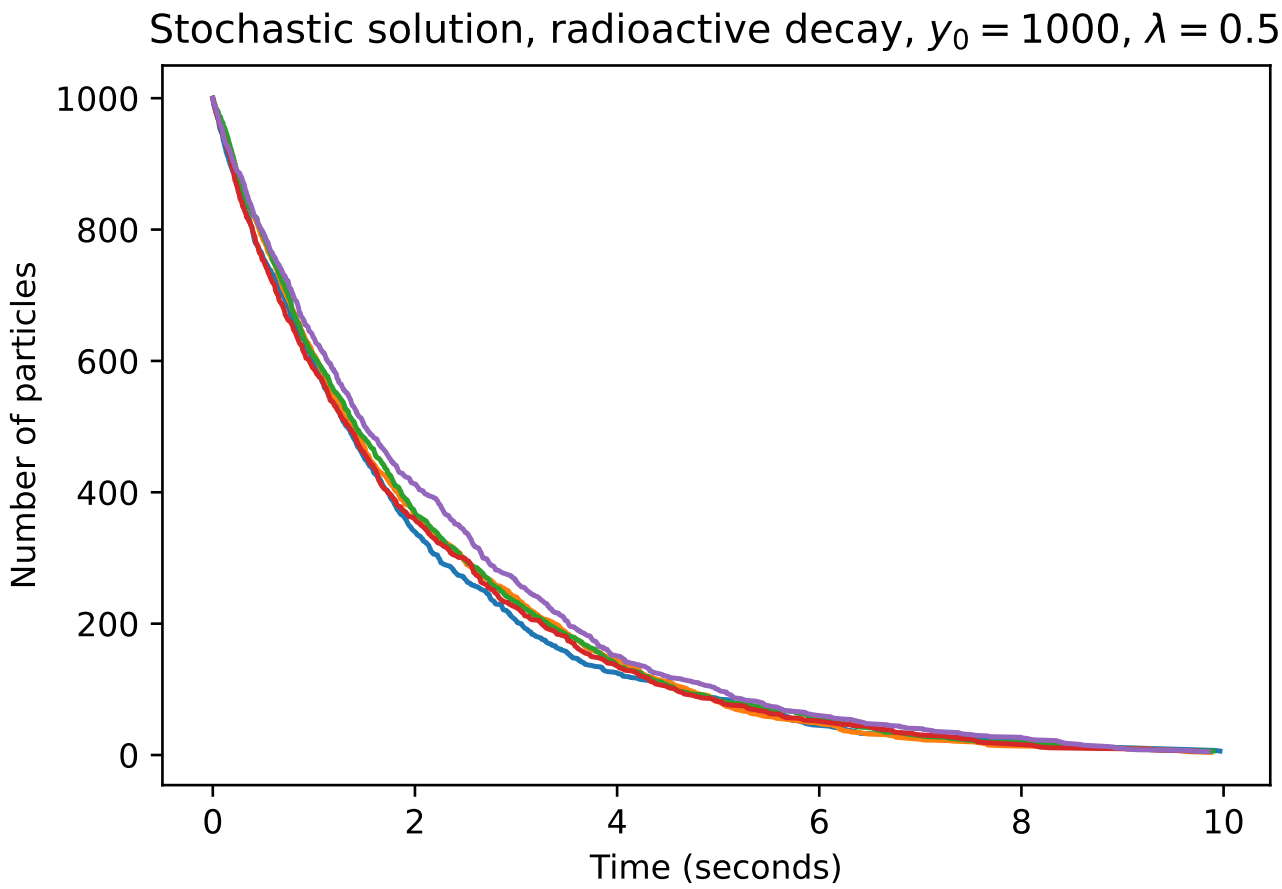
A stochastic model

Stochastic model, radioactive decay:

$$y \xrightarrow{\lambda} z, \qquad \lambda y \textit{ the probability for one decay}$$

- Meaning: "y undergoes the reaction ..." , here a radioactive decay to isotope z.
- λ is the *propensity* and λy the *propensity function* (tells us the probability that a decay will happen)
- We can simulate the stochastic model using a stochastic method (here Gillespies Algorithm, described later)
- Run it many times and calculate the mean - Monte Carlo simulation

A stochastic method



Numerical simulation with Stochastic Simulation Algorithm (SSA) (later in the course). In this case 5 simulations

The stochastic solution vs. reality:

- Different solutions despite same input data - for both stochastic model and stochastic method
- Unit is number of particles (integers):

$$y = 1000, 958, 919, 864, ...$$

- Better description of reality when a relatively small number of atoms
- If we obtain many simulations the mean of solutions at certain time will be normally distributed with expected value $=$ deterministic solution ("*Law of large numbers*" and "*Central limit theorem*" from Statistics)

Monte Calro Method

Monte Carlo methods

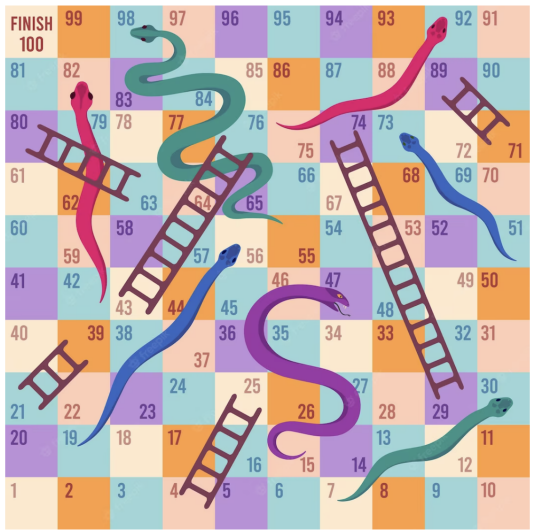
- Invented by *John von Neumann* and *Stanislaw Ulam* during World War II to improve decision making under uncertain conditions
- The name *Monte Carlo methods* is motivated by the randomness similar to games in the Monte Carlo casino

Let's play a game: Snakes-and-Ladders

- We play using a 6-sided dice (game's rules below)
- Let X be the number of tosses required to reach the finish
- What is the expected number of tosses required to finish? (Expectation of X?, math notation $\mathbb{E}(X) = ?$)

Game's rules:

- start from space 1, toss and move forward the number of spaces shown on the dice
- If you land at the base of a ladder move up to the top of the ladder
- If you land at the head of a snake slide down to the bottom of the snake
- to finish you will need to toss the exact number to get you to the last space 100. For example, if you are at space 99 then you should toss until getting 1 to finish



Monte Carlo method for solving snakes-and-ladders game

- One student finished the game after $x_1 = 51$ tosses. If N students play the game then we will have N observations (realizations) $x_1, x_2, \ldots, x_N$ of X
- Then, the expected number (expectation of X) is the average (mean):

$$\mathbb{E}(X) \approx \frac{1}{N}(x_1 + \cdots + x_N)$$

  This is the Monte Carlo method: do an experiment many times and finally take average!
- More observations (larger N), better estimation for mean
- No time to play this game many times, write an algorithm to do this job!
- We must simulate the dice rolling (generate random numbers from discrete uniform distribution $\mathcal{DU}\{1, 2, \ldots, 6\}$ with probabilities $1/6$)
- My simulation with $N = 10000$ shows the expected number is $\mathbb{E}(X) \approx 44$

MC method for solving snakes-and-ladders game

Other possible questions:

- What is the probability of finishing the game by exactly 30 tosses?

$$\mathbb{P}(X = 30) \approx \frac{\#30s}{N} \approx 0.02$$

- What is the probability of finishing the game by at most 30 tosses?

$$\mathbb{P}(X \leqslant 30) \approx \frac{\#30s + \#29s + \cdots + \#1s}{N} \approx 0.34$$

- What is the probability of finishing the game by more than 30 tosses? (31 and more)

$$\mathbb{P}(X > 30) \approx \frac{\#31s + \#32s + \cdots}{N} = 1 - \mathbb{P}(X \leqslant 30) \approx 0.66$$

- Approximate the standard deviation of X:

$$\sigma_X \approx s_N = \sqrt{\frac{1}{N-1}\sum_{k=1}^{N}(x_k - \bar{x})^2} \approx 27$$

General structure of Monte Carlo algorithm

$$
\begin{aligned}
&\textit{Input } N: \ (\textit{Number of observations})\\
&\textbf{for } k = 1:N\\
&\quad \textit{perform one stochastic simulation/process}\\
&\quad \textit{result}[k] = \textit{result of the simulation}\\
&\textbf{end}\\
&\textit{FinalResult} = \textit{mean}(\textit{result}) \textit{ or other statistical calculation}
\end{aligned}
$$

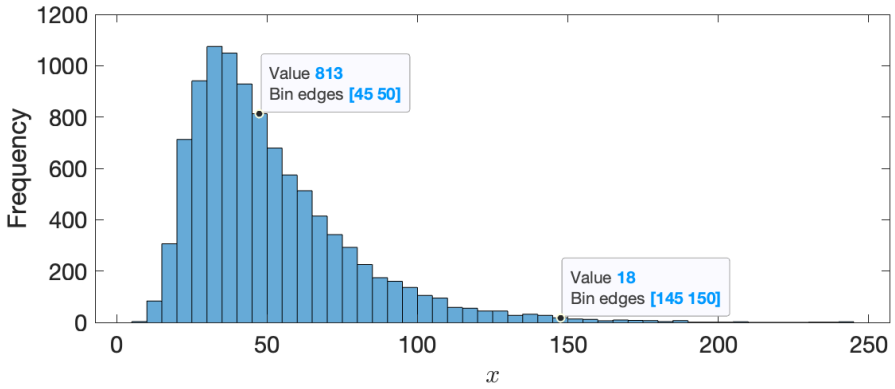
- Not only the mean but also some other statistical conclusions
- What the stochastic simulation/process is, can differ, e.g. a dice or Brownian motion

Review of prior concepts

- Random experiment: an experiment whose outcome cannot be determined in advance (Snakes-and-ladders game)
- Outcome: an observation (realization) of the experiment ($x_1 = 51$ for the above game), usually denoted by lowercase letters like $x, y, \ldots$
- Sample space: all possible outcomes of the random experiment (In this example $\{7, 8, 9, \ldots, 43, 44, \ldots\}$)
- Random variable: any function defined on the sample space, usually denoted by capital letters $X, Y, \ldots$
  - Discrete random variable
  - Continuous random variable
- Probability density function (pdf): the probability model of X, usually denoted by f. For discrete case $f(x) = \mathbb{P}(X = x)$
- Cumulative density function (cdf): The probability that X will take a value less than or equal to x, denoted by capital letters $F(x) = \mathbb{P}(X \leqslant x)$.

The pdf of X (Snakes-Ladders game)

If 10000 people play the snakes-and-ladders game then the frequency of X looks like this figure



This means for example 813 people finished the game using 45-50 tosses and 18 unlucky people did it using 145-150 tosses. The pdf looks like: