Modelling and Simulations of Systems

Lecture 1: Introduction

Piotr Kowalczyk & Adam Lamęcki

Materials & marks

Course slides & lab instructions are stored on platform:

https://enauczanie.pg.edu.pl

Course name:

Modelowanie i Symulacja Systemów - 21/22
Modelling and Simulations of Systems – 21/22

Enrolment key:

miss2021

Course description & marks

In this course we have:

- 15 lectures (15 x 2h)
- 5 laboratory classes (5 x 3h!), Matlab, in groups of 3 students

Maximum number of points to collect:

- 60 points during 5 laboratory classes (12 points each)
- 40 points during 2 short tests (lecture time)

The condition for passing the course is to obtain a minimum of **20** points (at least **10** points from each test) in the lecture part and a minimum of **30** points in the laboratory part. The final grade consists of the sum of points.

Course topics

1. Goal of modelling and applications
2. Model construction
3. Discretization of differentia equations
4. Numerical solution of discrete problems
5. Introduction to computer simulations (FD, FEM …)
6. Data compression and processing
7. Basics of stochastic methods
8. ….

Theoretical knowledge is consolidated by the implementation of the discussed algorithms in laboratory classes (<u>the Matlab environment should be known to all participants of the course before its commencement!</u>).

Consultation hours

- Piotr Kowalczyk

room 722 EA

Mon 15 – 16

Wed 10 - 11

- Adam Lamęcki

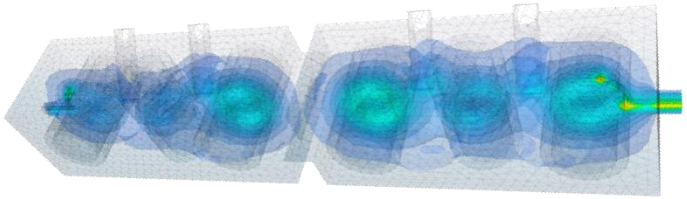
room 708 EA

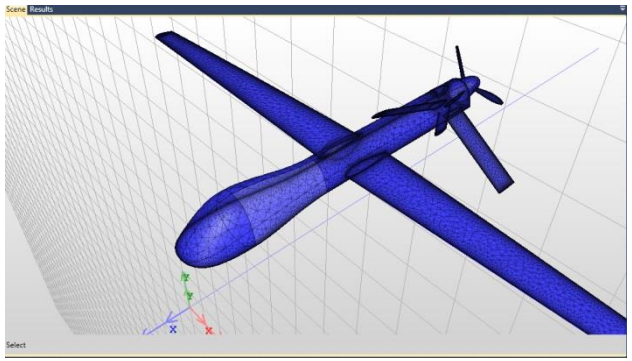
TBD???

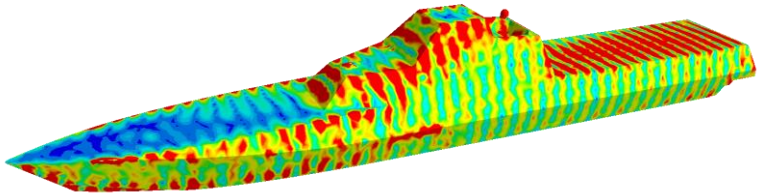
Department overview

Since many years we do R&D in computer simulation techniques:

- Finite differences (FD)
- Finite element methods (FEM)
- Method of moments (MoM)
- Surrogate (black-box) models
- Model order reduction techniques (MOR)
- Neural networks (ANN)
- Design optimization techniques (CAD)





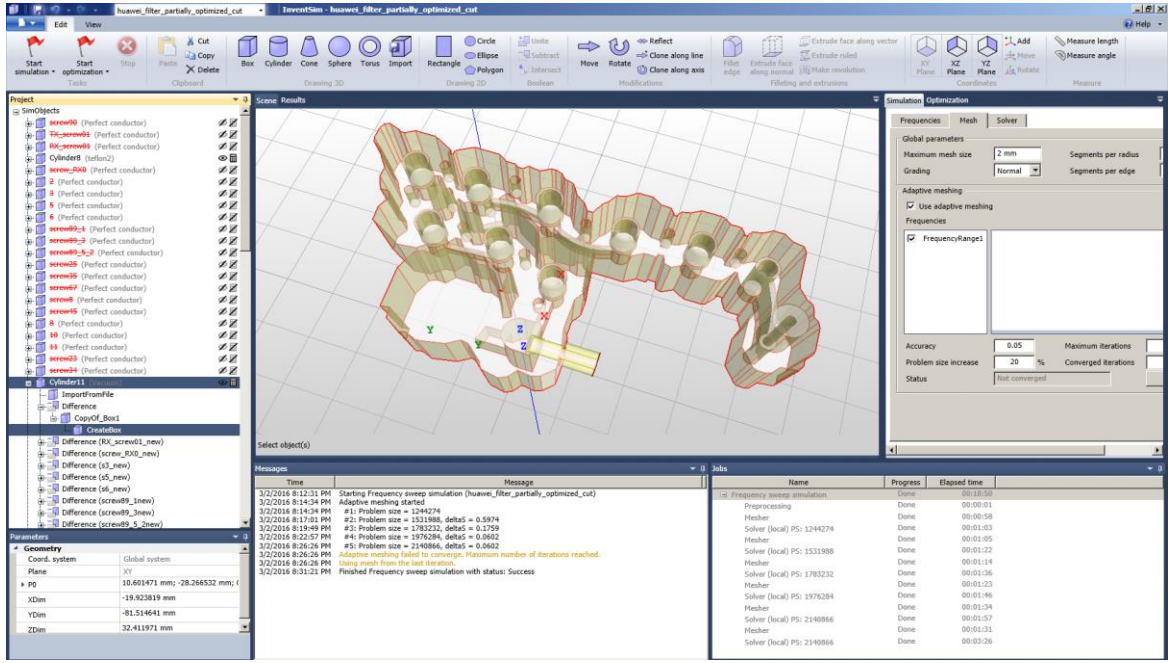


Department overview

- NVidia CUDA Research center – GPU Computing activity

NVIDIA
GPU
RESEARCH
CENTER

- Fast multithreaded solvers – C/C++ programming (including low level code optimization)



- Distrubuted computing (MPI)

- International & industry activities

Why do we need modeling?

- Mathematical models allow us to better understand some phenomenon, predict its process and results
- In result we can „predict the future” and have impact on it
- In most cases we do not want to use trial & error scheme
- Depending on how accurate we want to be, we use different complexity models
- In result models evolve in time

- Finally – we can not prove the model is correct, but we can prove that it is incorrect

Modern applications

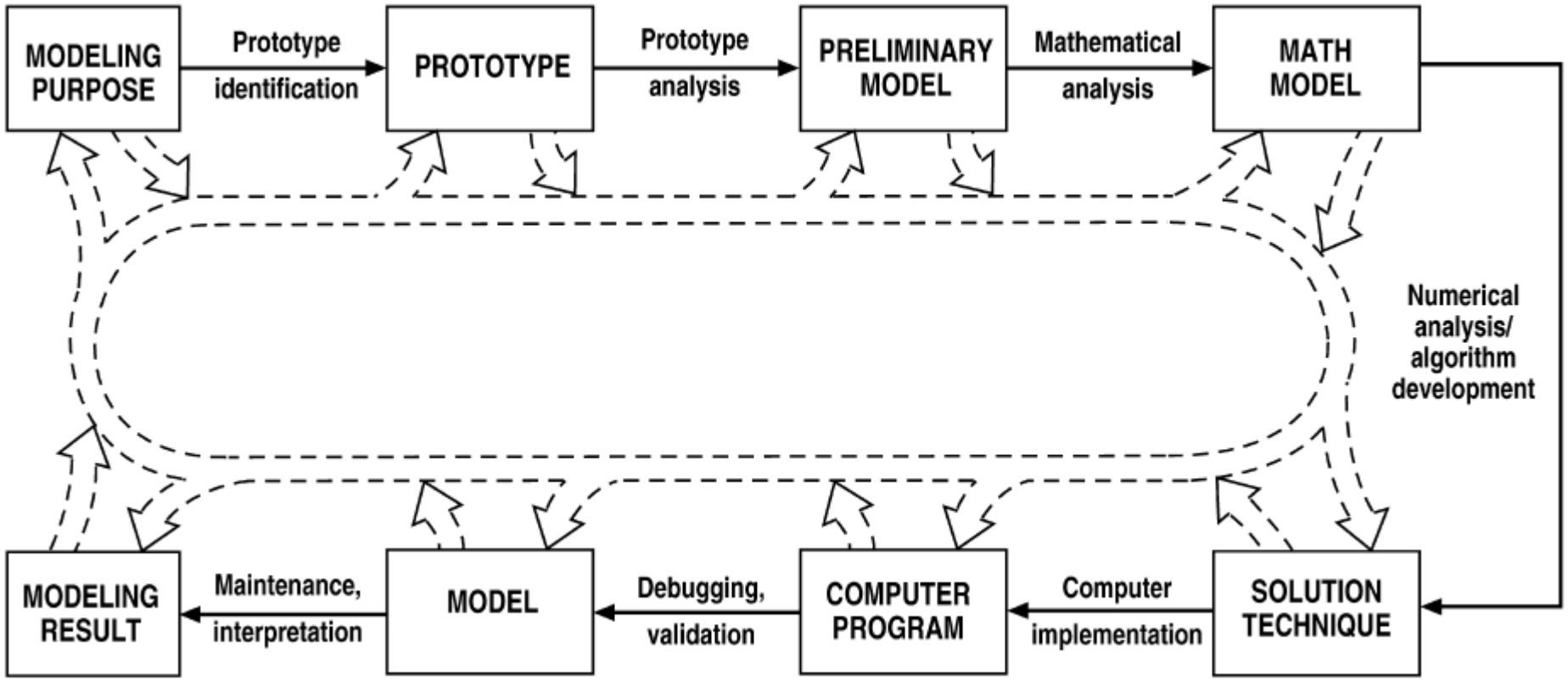
- Physics
- Industry: maritime, automotive, aerospace (high cost of prototype)
- Construction (stress analysis)
- Medicine (diagnosis and treatment)
- Weather prediction
- Economy & social science (statistic models)
- Video games (modelling of physical phenomenon)
- …

Informatics vs. Computer Science

- Informatics is a branch of information engineering. It involves the practice of information processing and the engineering of information systems, and as an academic field it is an applied form of information science (Wikipedia)

- Computer science is the study of computers and computation systems. **It is the scientific and practical approach to computation and its applications**.

Basic flow of modelling & simulation



Some history

- **Basic mathematics tools (e.g. geometry)**

- Pythagorean theorem (VI BC)
- Thales' theorem (VII/VI BC)
- …

- **Simple physical phenomenon**

- Aristotle (IV BC): simple observation-based theories
- Archimedes (III BC): Archimedes' principle

Some history

- **Key person – *Isaac Newton (1642-1727).*** Introduced three laws of motion, model of universal gravitational force, differential and integral calculus (is it a coincidence that in XVIII age industrial revolution started?)

- Model of body motion and interaction (N-body problem)

Some history – new planet?

- In 1781r. William Herschel accidentally discovers a new planet (Uranus), initially mistaken with comet

- Alexis Bouvard in 1821r. Develops a mathematical model of movement of 3 planets: Jupiter, Saturn and Uranus. For the latter the model does not work – even if other planets are taken into account

- There is a hypothesis that historical observation data are misleading or that some unknown force influences the movement

Some history – new planet?

- In 1845-1846 French mathematician Le Verrier develops new calculations – for the first time defines so-called „inverse problem” – he tries to find the mass and orbit of unknown object from obervation datasets
- He published his solution, but it was not commonly accepted
- 23 Sept. 1846 the correctness of his results is confirmed by

Johann Gottfried Galle – first observation of new planet: Neptune

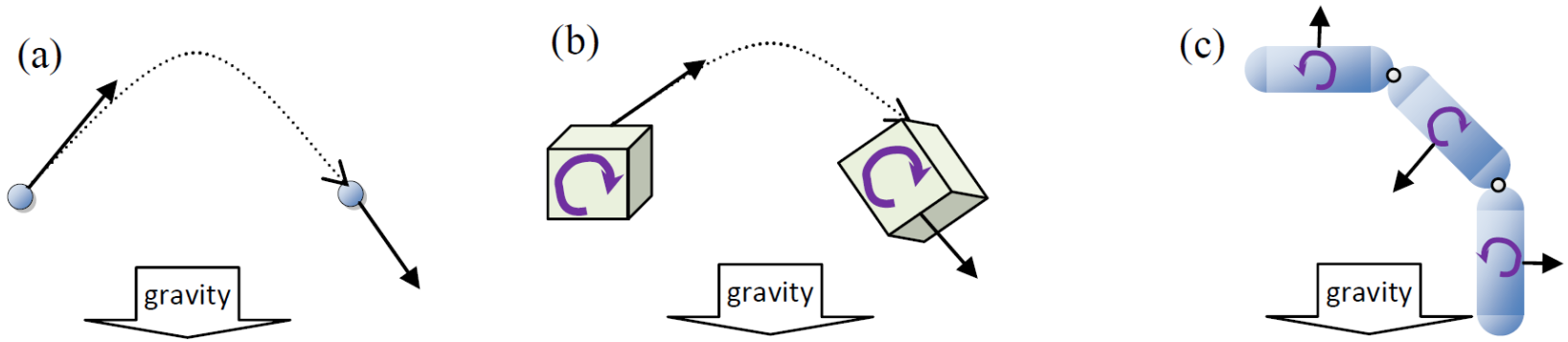
Some history…

- Development of new tools and mathematics caused further progress in science (especially electricity). Second industrial revolutions comes in XIX and XX age.
- Newton model appeared to be insufficient for high-speed objects. New theory of relativity is proposed by **Einstein**.
- On the other hand we have microscopic scale and development of quantum mechanics (**Schroedinger** i **Dirac**).

- Finally the last breakthrough: COMPUTER SCIENCE (Simulations). New opportunities to model the world. Third industrial revolution has started in 50’s

Physical simulations (entertainment)

Video games **(source: https://software.intel.com/en-us/articles/fluid-simulation-for-video-games-part-1/)**



Particles are points that have position, mass, and velocity but no size or shape (a). The relationship between forces and motion is linear. Particles are easy to simulate but rather uninteresting.

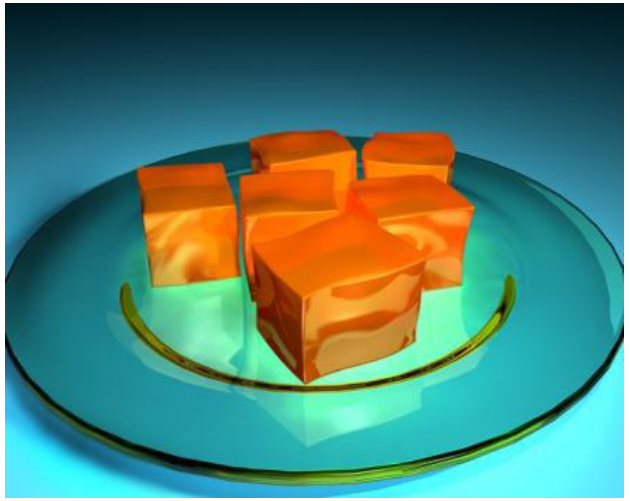
Rigid bodies have shape and orientation in addition to position, mass, and velocity (b). Rigid bodies are still easy to simulate: Most of the difficulty comes from detecting and responding to collisions. Stacks of bodies are usually the most difficult to solve, because everything in the stack continuously collides with everything else in the stack—even if nothing moves.

Articulated bodies are connected networks of rigid bodies (c). These bodies behave identically to rigid bodies that are continuously involved in a form of collision where the points of contact have a limited variety of ways in which they can move (called constraints).

Physical simulations (entertainment)

Video games (source: https://software.intel.com/en-us/articles/fluid-simulation-for-video-games-part-1/)

Deformable bodies can change shape but retain their connectedness and adjacency of various points on the body. Think of this as a model where the edges between vertices never change which vertices they connect, but the locations of the vertices can move.



Physical simulations (entertainment)

Video games (source: https://software.intel.com/en-us/articles/fluid-simulation-for-video-games-part-1/)

Fluids have lots of freedom of motion. The motion is nonlinear, and their shape and topology can change. Fluids require specialized simulation techniques – fluids take the shape of their container, they are always in collision with everything around them, including the fluid itself. So a collision with one part of the fluid effectively means that the whole body of fluid must respond.

Smoke seems to behave like a gas but also appears to have a kind of surface, although perhaps not as distinct as that of a liquid. Smoke is really a combination of a gas and tiny suspended particulates, and the combination of these particulates is called an aerosol. Those particulates follow the motion of the gas (and let game players see that motion) without necessarily influencing the motion.

Physical simulations (entertainment)

Video games (source: https://software.intel.com/en-us/articles/fluid-simulation-for-video-games-part-1/)

Similar to Newton's laws of motion, the Navier-Stokes equation expresses how velocity changes as a result of forces:

$$\frac{\partial \vec{\mathbf{v}}}{\partial t} + (\vec{\mathbf{v}} \cdot \vec{\nabla})\vec{\mathbf{v}} = \frac{-\vec{\nabla} p}{\rho} + \frac{\mu}{\rho}\nabla^2 \vec{\mathbf{v}} + \frac{\vec{f}_{ext}}{\rho}$$

Here, $\vec{\mathbf{v}}$ is the velocity at a point in time and space, t is time, p is pressure at a point, p is the density of the fluid at a point, μ is viscosity, and $\vec{f}_{ext}$ are external forces, such as gravity, acting on the fluid.

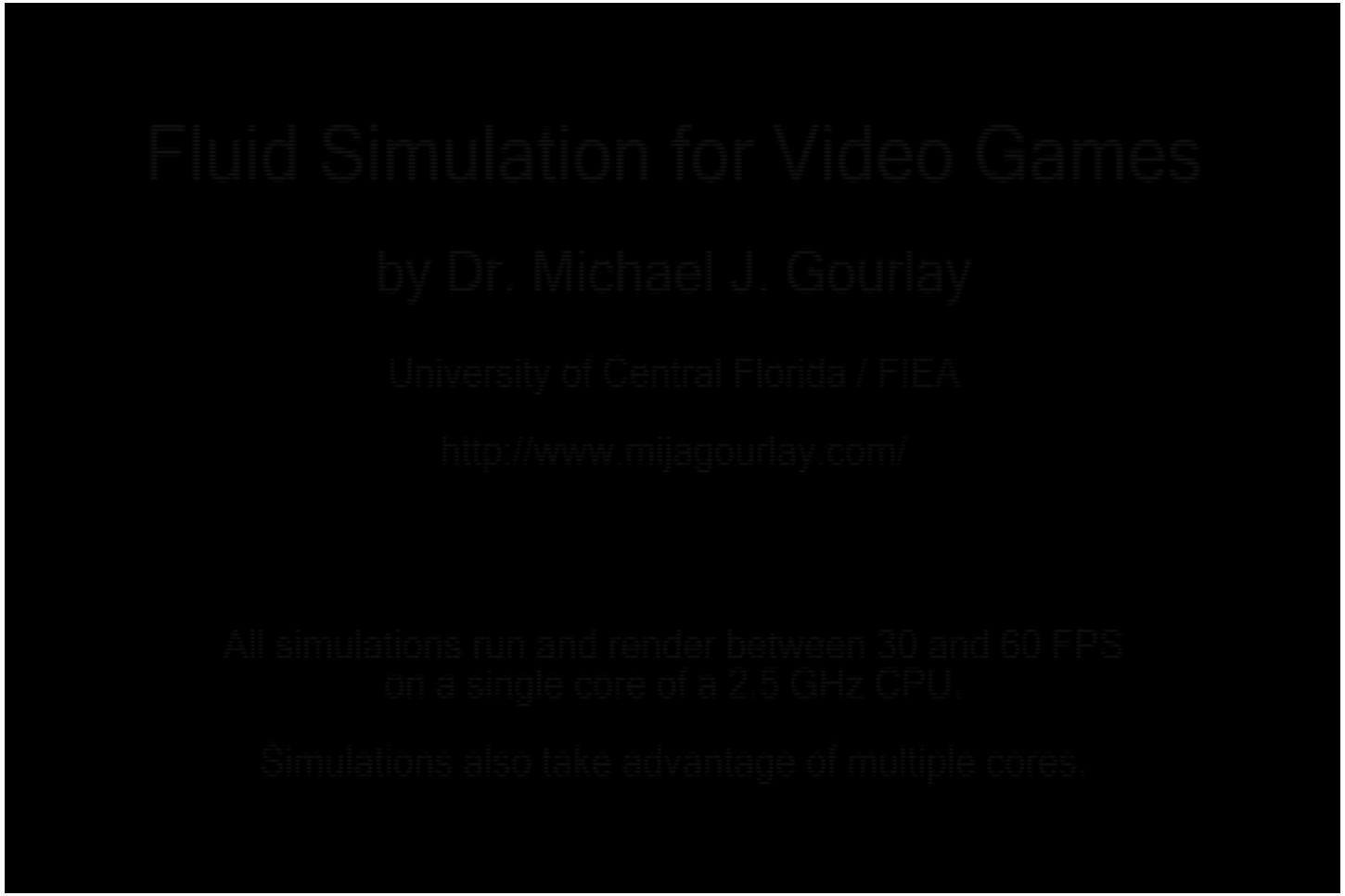
Note the similarities and differences to the equation for the block. Both express how velocity changes over time. Both include forces resulting from contact as well as external forces. But the fluid equation has an extra term on the left, $(\vec{\mathbf{v}} \cdot \vec{\nabla})\vec{\mathbf{v}}$, that takes some effort to understand.
The terms on the left express acceleration and have a special meaning. You can rewrite them as a new operator, $\frac{D}{Dt}$:

$$\frac{\partial \vec{\mathbf{v}}}{\partial t} + (\vec{\mathbf{v}} \cdot \vec{\nabla})\vec{\mathbf{v}} \equiv \frac{D\vec{\mathbf{v}}}{Dt}$$

This formula is called by many names, including the *substantive derivative, advective derivative*, *Lagrangianderivative*, and *particle derivative*. These names give a clue to its unusual meaning, and this is, in some sense, the heart of fluid motion. So, let's break it down, because to understand fluid motion, you must understand this derivative.

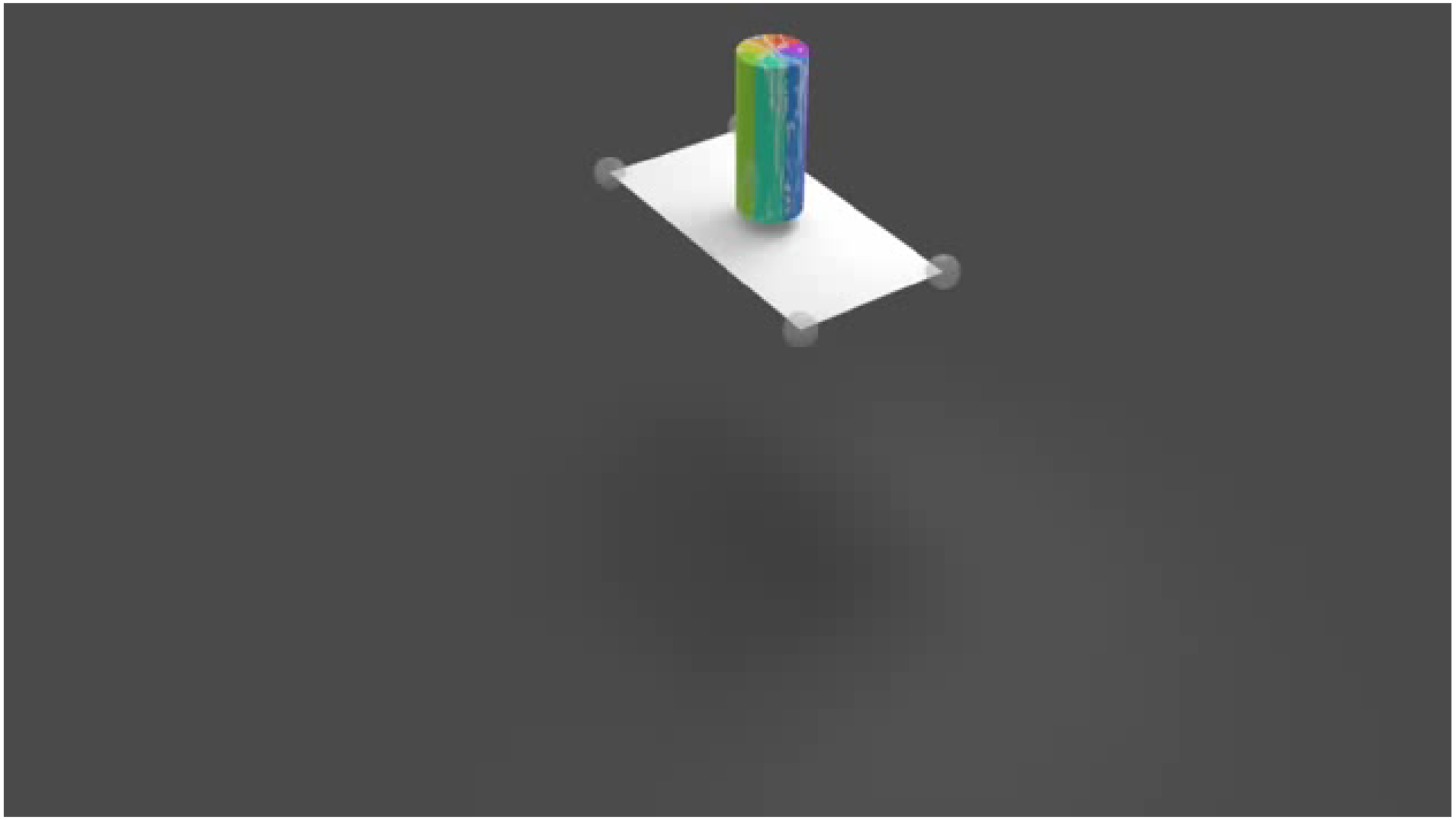
Physical simulations (entertainment)

Examples (https://software.intel.com/en-us/articles/fluid-simulation-for-video-games-part-3/



Physical simulations (entertainment)

Examples: SIGGRAPH 2017



Most common procedure in physical simulations

- Material properties – physical quantities (mass, force, velocity, momentum, energy, pressure, density, viscosity, temperature,…)
- Governing equations – partial differential equations (Newton's laws, Navier-Stokes equation, …)
- Initial and boundary conditions – define the initial state of the system and the computational domian limits (starting values and the boundary)
- Numerical techniques solving the problem – usually requires a discretization of the computational domain (different types of grids or meshes)

Physical simulations (science)

Cosmological N-Body Simulations

- Governing equations
  1. Newton’s laws of motion
  2. conservation of momentum and energy
  3. Newton's law of universal gravitation
- Initial conditions
  1. starting positions
  2. starting velocities

1 Teraflop Nvidia CUDA N-Body Simulation
N= 35000

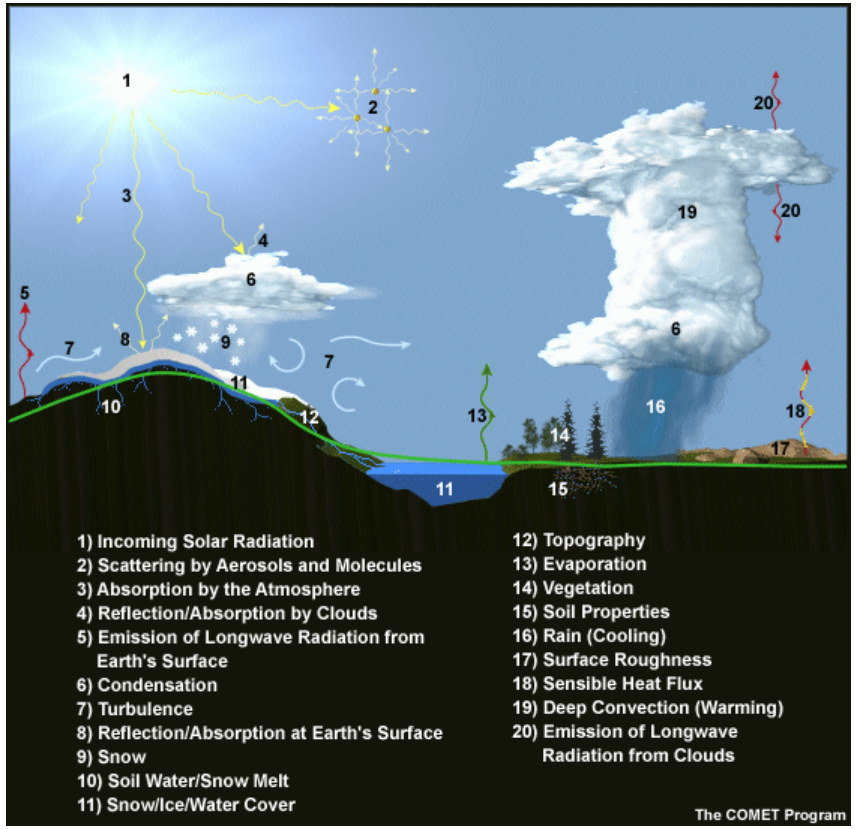
Source: https://www.youtube.com/watch?v=TvLGynskpms

For greater number of objects different models must be used (e.g. in practice a number of particles of gas or liquid is about 10^{23} then usually probability distribution is employed)!

Weather Models

Numerical Weather Prediction uses mathematical models of the atmosphere and oceans to predict the weather based on current weather conditions. Though first attempted in the 1920s, it was impossible until the advent of computer simulation in the 1950s that numerical weather predictions produced realistic results.

- Governing equations (Newton's laws, conservation of momentum, mass and energy, Ideal gas law)
- Initial and boundary conditions (define the atmosphere's current state - the starting point and domains' edges)
- Numerical methods (to solve the equations)



Source:
https://en.wikipedia.org/wiki/Numerical_weather_prediction
http://www.weather.gov/media/ajk/brochures/NumericalWeatherPrediction.pdf

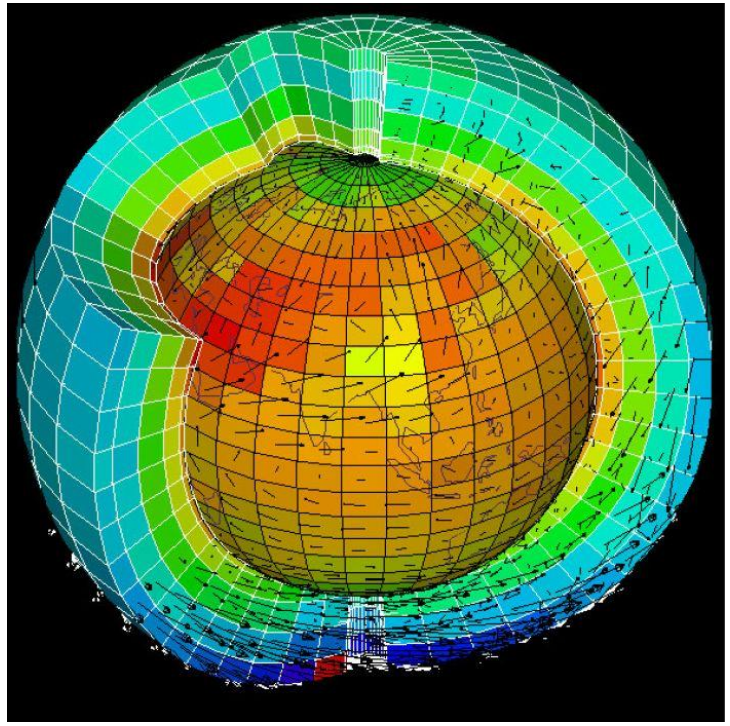
Weather Models

Models use systems of differential equations which are solved in 3D grid.

The computational complexity and memory requirement for a denser grid are higher but the results are more accurate.

Examples:

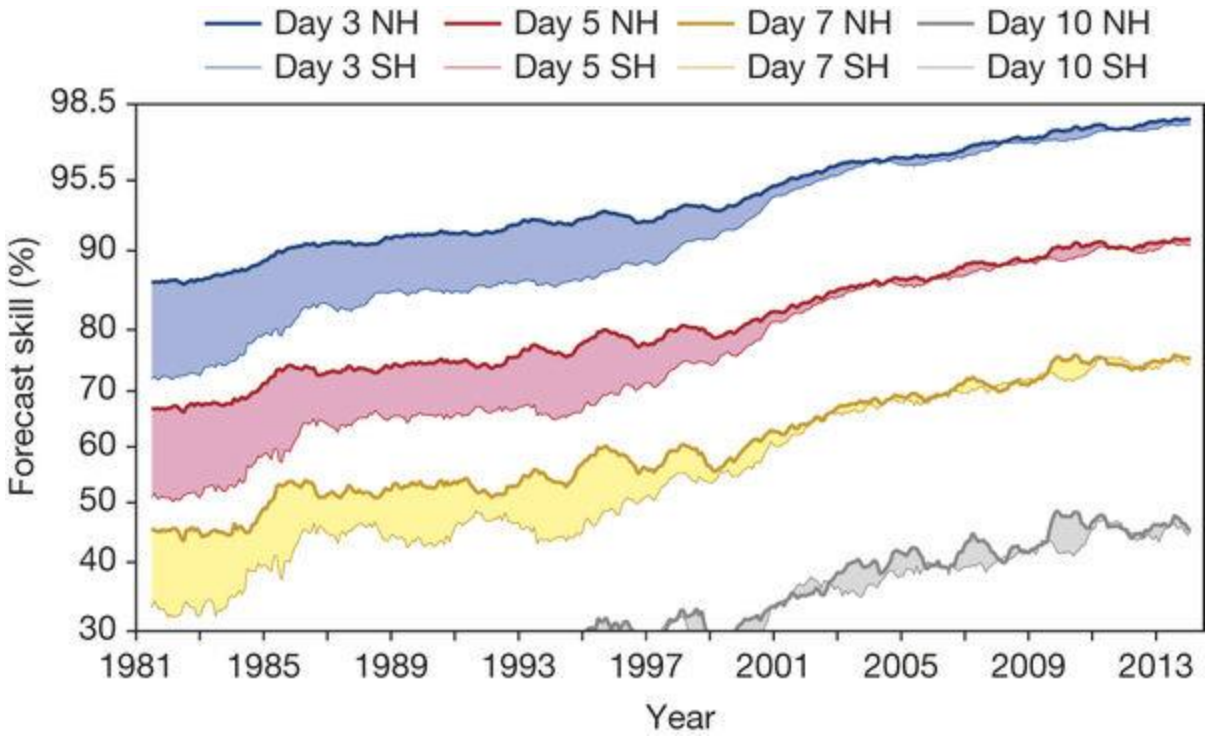
- http://www.meteo.pl/
- https://www.yr.no/
- https://www.ncdc.noaa.gov/about



Source: http://www.abc.net.au/radionational/programs/scienceshow/the-power-and-limits-of-weather-models/6507508

Weather Models

Today's 5-day weather predictions have the accuracy of 3-day predictions thirty years ago thanks to improved weather models.



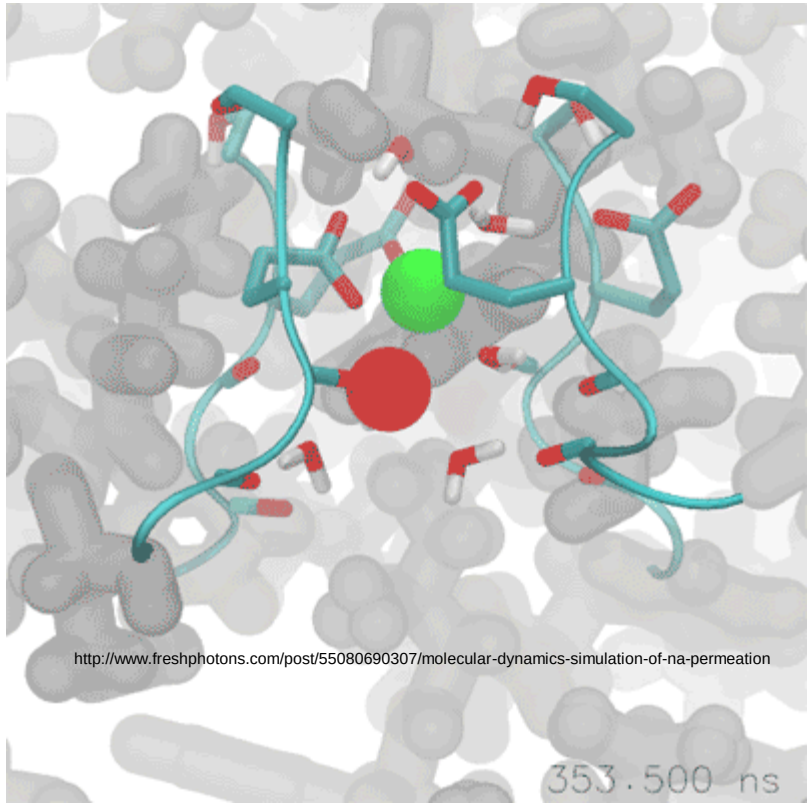
Forecast skill is the correlation between the forecasts and the verifying analysis of the height of the 500-hPa level, expressed as the anomaly with respect to the climatological height. Values greater than 60% indicate useful forecasts, while those greater than 80% represent a high degree of accuracy. The convergence of the curves for Northern Hemisphere (NH) and Southern Hemisphere (SH) after 1999 indicates the breakthrough in exploiting satellite data through the use of variational data.

Source: Peter Bauer, Alan Thorpe, Gilbert Brunet „The quiet revolution of numerical weather prediction", Nature 525, 47–55 (03 September 2015)

Medicine & Chemistry

In general pharmaceutical industry spends about $50 billion to produce ~20 new drugs each year- general success ratio is about 3%!

Drug discovery uses computational chemistry (eg. molecular dynamics techniques) to predict the interactions of various active particles.



Medicine & Chemistry

During simulation thousands to millions of individual atoms representing, for instance, all the atoms of a protein and surrounding water molecules, move in a series of short (e.g., 2 fs), discrete time steps.

In each step, the forces on each atom—determined from the ‘‘force field,’’ a collection of physics-based parameters that represent both bonded and non-bonded (e.g., van der Waals) inter-atomic forces—are computed and the atomic position and velocity are updated according to Newton’s laws of Motion.

This process is repeated billions of times to provide continuous atomic trajectories lasting as long as 1 micro second, or even longer.*

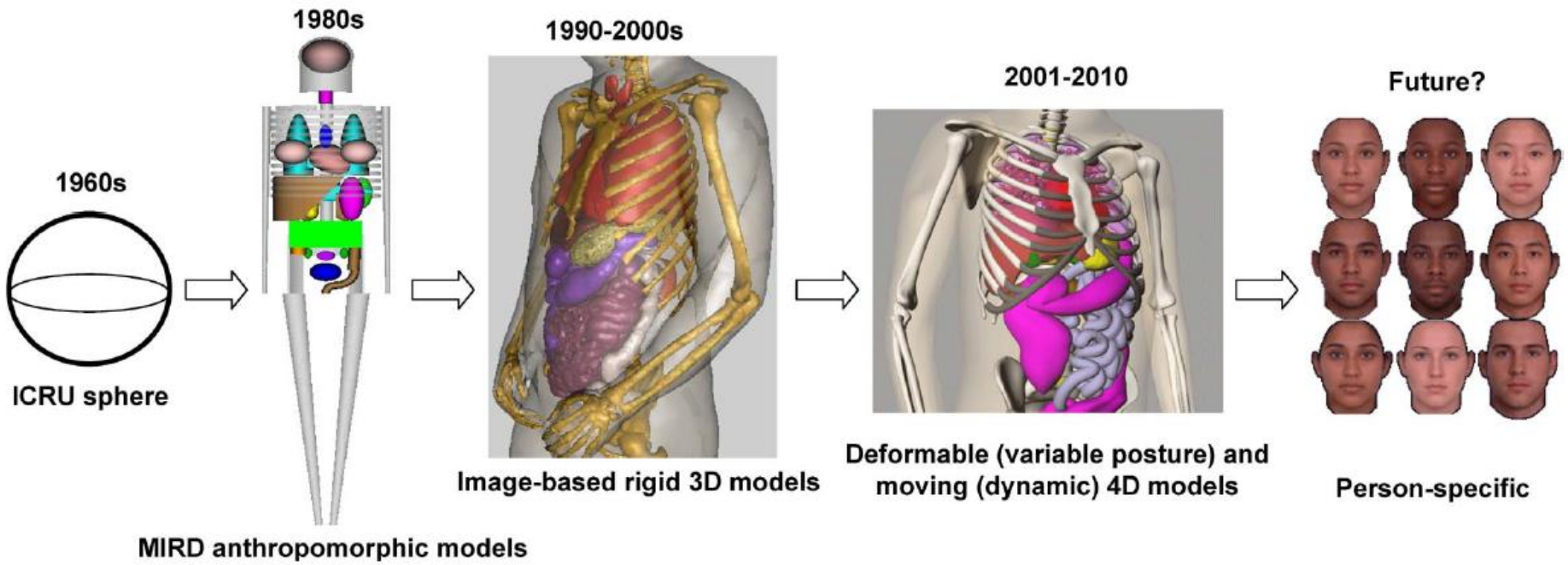
*David W. Borhani, David E. Shaw „The future of molecular dynamics simulations in drug discovery”, J Comput Aided Mol Des (2012)

Medicine - Dosimetry

- Permits a better understanding of how radiation interacts with biological tissues through various mechanisms of radiation interaction with matter and deposits energy (for radiation dosimetry applications) or to determine the best detection geometry configuration of a molecular imaging unit for imaging a particular region of the human body (e.g. brain, breast, prostate, etc.).

Medicine - Dosimetry

- Evolution of computer model of human body:



Habib Zaidi ; Benjamin M. W. Tsu, „*Review of Computational Anthropomorphic Anatomical and Physiological Models*," Proceedings of the IEEE, Vol. 97, Iss. 12, 2009

Structural design

- Up to 1940s the construction of suspension bridges was mostly done using static-force analysis
- Scale-models were also developed but tested in static conditions (including wind-tunnel tests)
- In 1938 the construction of Tacoma Narrows Bridge was started. The bridge was very long (840m) and narrow (17m)

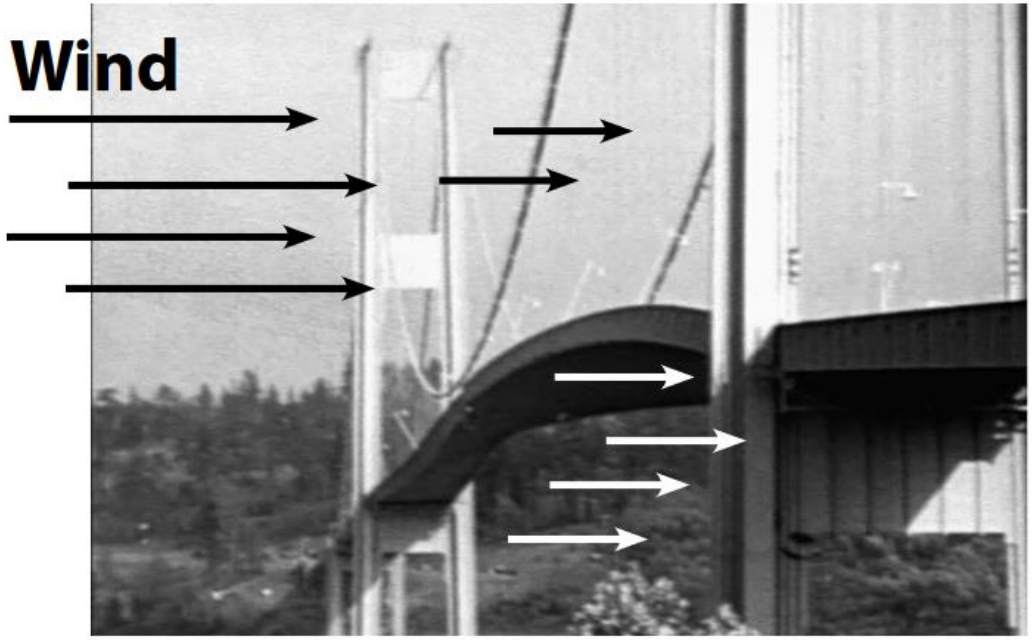
- From the time the bridge was built, it began to move vertically in windy conditions - shortly after construction it became a tourist attraction!

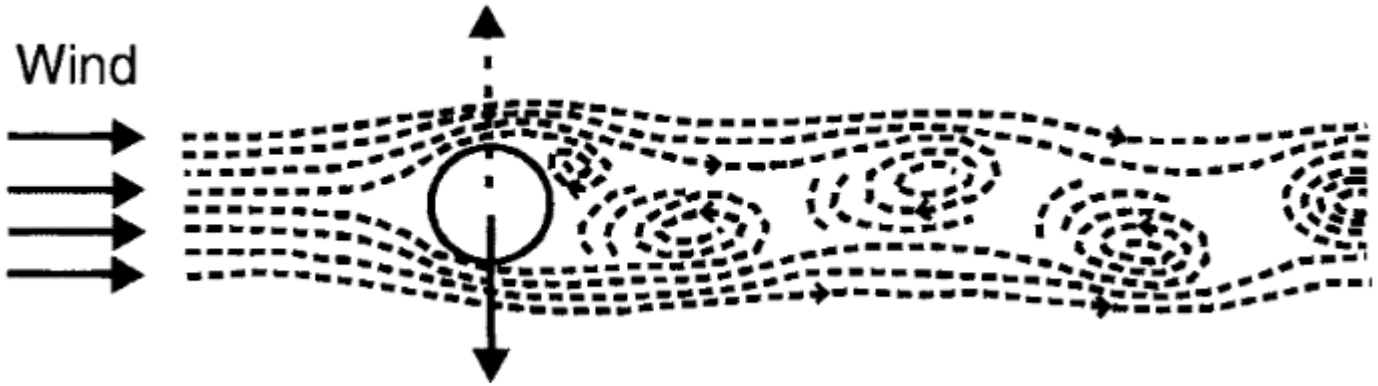
Structural design

- Tacoma Narrows Bridge Collapse: 07 Nov. 1940 (4 months after opening)
- Wind speed: 64 km/h

Structural design

- Tacoma Narrows Bridge





Simulation: fluid-structure interaction (CFD)

https://www.youtube.com/watch?v=YzvFxF5LrRA

Chemical kinetics

Kinetic modelling is a technique that is very useful in relation to food processing and food quality. The reason is twofold. First, changes in foods as a result of processing and storage lead to a change in quality (usually a quality loss). The processes involved are mainly (bio)chemical and physical reactions. Such changes proceed at a certain rate and with certain kinetics. Kinetic modelling enables us to describe these changes and their rates quantitatively. Second, with kinetic modelling we have a powerful tool that can help to unravel basic reaction mechanisms. The understanding of the basic mechanisms is vital for quality modelling and quality control.

Standard procedure

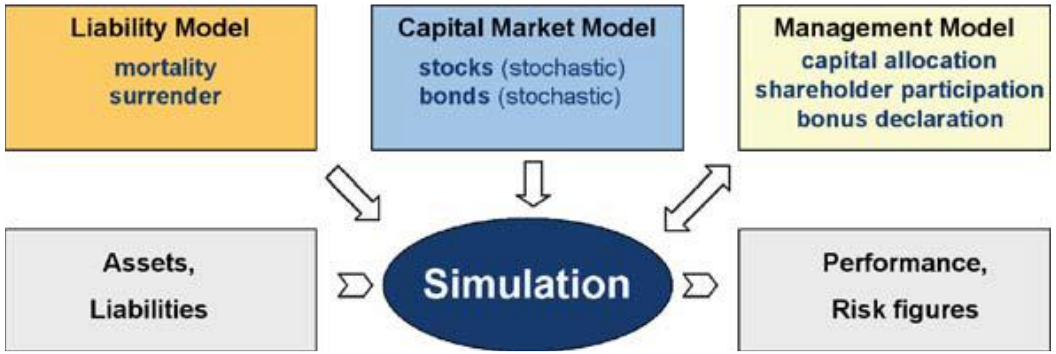
- Properties and quantities (nature of the reactants, physical state, surface area of solids, concentration, temperature)
- Governing equations (in general nonlinear differential equations)
- Numerical techniques to solve the problem

Source: L. M. M. Tijskens, M. L. A. T. M. Hertog B. M. Nicolai, „Food Process Modelling”, Woodhead Publishing 2001

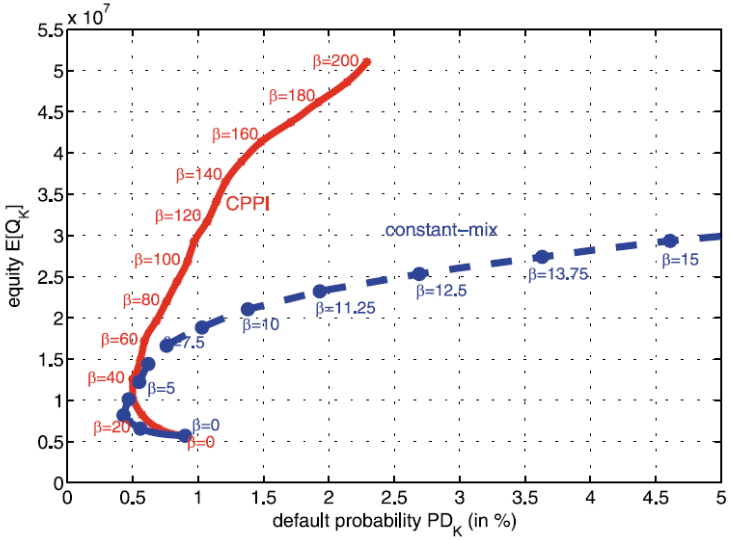
Economics

Numerical Simulation for Asset-Liability Management in Life Insurance

Overall structure of the model



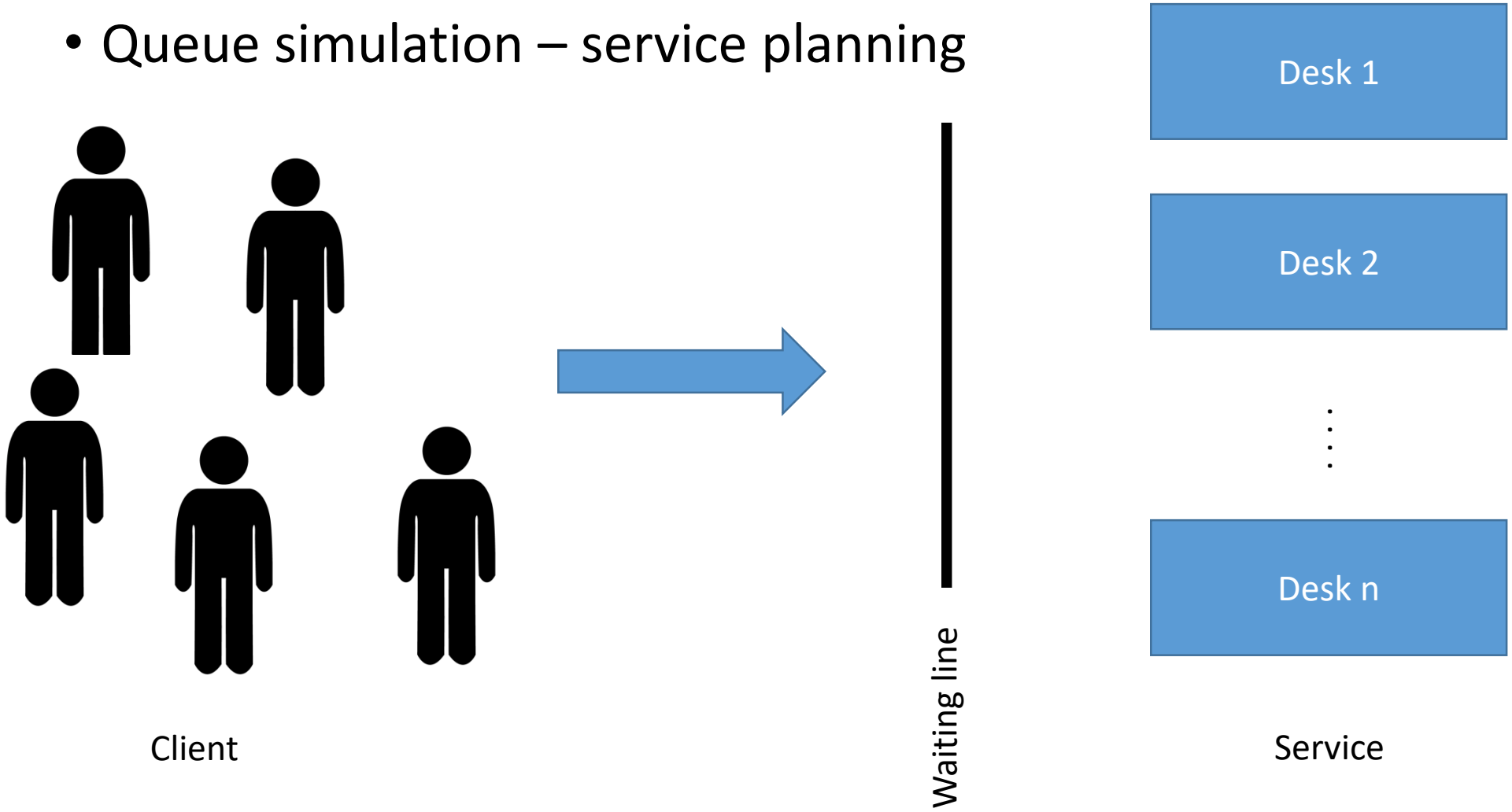
Risk-return profiles of the different capital allocation strategies



Source: T. Gerstner, M. Griebel, M. Holtz, R. Goschnick, M. Haep, „Mathematics -- Key Technology for the Future: Numerical Simulation for Asset-Liability Management in Life Insurance", Springer 2008

Discrete event simulations

- Queue simulation – service planning



Discrete event simulations

- Case Study: Ski resort
  - 4 sky lifts are run. 3 of them are ski-tows which are able to pull one skier or snowboarder uphill. One of them is a six-seat chairlift.
  - The skiers might sometimes miss the tow even when they are in the queue.

Table 1 Characteristics of the probability profile distributions used for the number of passengers in the ropeway cabin depending on the number of skiers in the queue

Probability in percentage		Number of skiers in the cabin						
		0	1	2	3	4	5	6
Number of skiers in the queue	0	100%	0%	0%	0%	0%	0%	0%
	1	5%	95%	0%	0%	0%	0%	0%
	2	5%	15%	80%	0%	0%	0%	0%
	3	4%	11%	30%	55%	0%	0%	0%
	4	3%	7%	20%	30%	40%	0%	0%
	5	2%	5%	13%	25%	30%	25%	0%
	6 and more	0,3%	0,7%	4%	5%	20%	40%	30%

  - The skiers can use each of the ski lifts and ski slopes in the center. There are four different ski slopes. The choices of the individual skiers are stochastic and based on a set of complicated probability rules when selecting one of the ski slopes or ski lifts.
  - Skier waiting for too long in one queue will next time choose another ski lift with higher probability. The intensity of the changes in probabilities depends on the skier's last queuing time. The length of skiing down a ski slope was set from 5 minutes to 10 minutes.

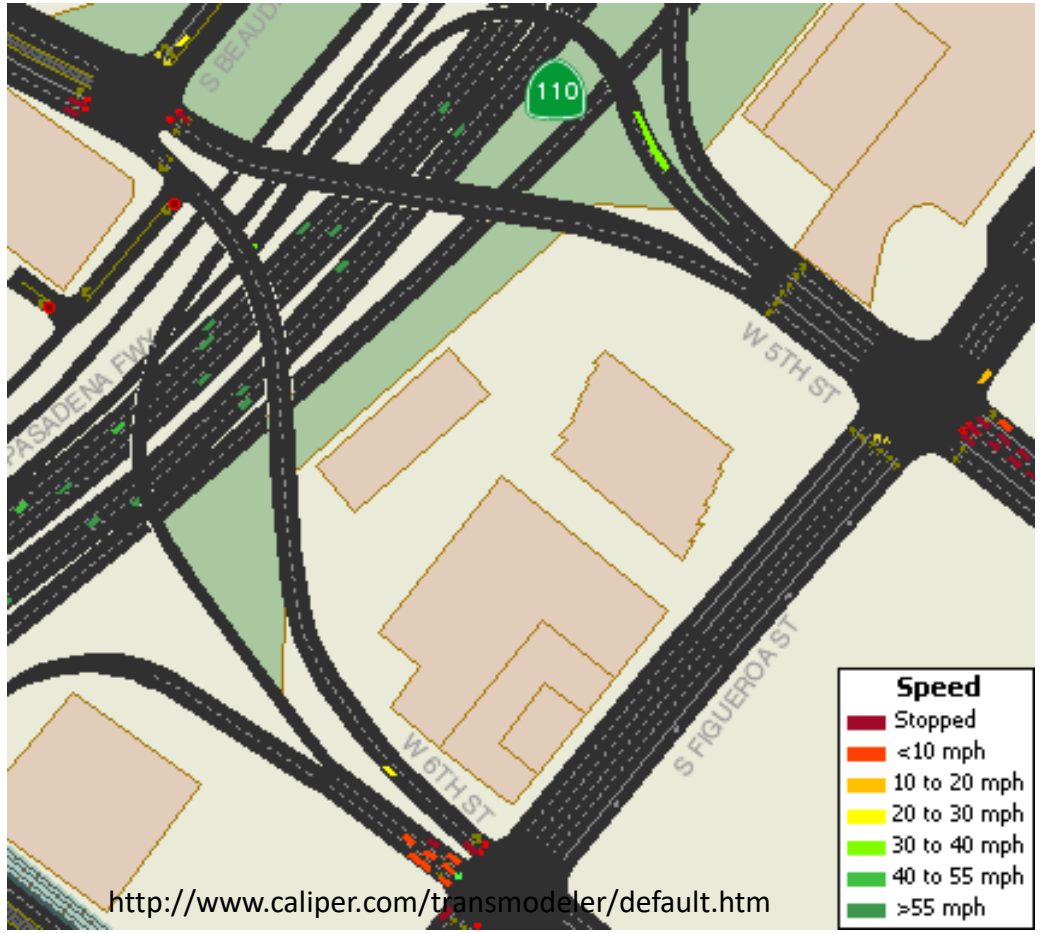
Discrete event simulations

- Case Study: Ski resort
  - The skiers can arrive at the ski center by cars before midday from eight a.m. to midday. Eighty percent of visitors come to the center from eight a.m. to ten a.m. and twenty percent of skiers arrive at the centre from ten a.m. to twelve a.m. The inter-arrival time of vehicles is modeled by exponential distribution.
  - The number of visitors who arrived in one car is modeled with a discrete probability profile distribution. A car contains one skier with the probability of ten percent; a car contains two skiers with the probability of twenty-five percent; a car contains three skiers with the probability of thirty-five percent; finally, a car contains four skiers with the probability of twenty-five percent.
  - The skiers can leave the ski center from 1 p.m. to 4 p.m. The intensity of leaving the ski center rises in time.
  - The maximum number of skiers pulled uphill in each ski tow is approximately nine hundred per one hour. The maximum number of people transported uphill with the six-seat chairlift is approximately two thousand and four hundred per hour.

Tomás Domonkos, „Computer Simulation as a Tool for Analyzing and Optimizing Real-Life Processes," International Scientific Journal of Management Information Systems 2010

Traffic simulation

- Network planning and optimization:



http://www.caliper.com/transmodeler/default.htm

Types of models

- Linear or Nonlinear: type of relation between simulation parameters and unknowns
- Static or Dynamic: influence of time
- Deterministic or Stochastic: randomness

Modeling and Simulation of Systems

Lecture 2: Mathematics - basic modeling tool, part I

Piotr Kowalczyk

Scalars and vectors

Scalars are quantities that are fully described by a magnitude (or numerical value) alone.

- mass
- temperature
- pressure
- energy
- …

Vectors are quantities that are fully described by both a magnitude and a direction.

- displacement
- velocity
- force
- acceleration
- …

Vectors

Let $\vec{v}_1, \vec{v}_2, \vec{v}_3, \dots \vec{v}_M \in R^N$ where $\vec{v}_m = [v_m^{(1)}, v_m^{(2)}, \dots, v_m^{(N)}]$ and $\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_M \in R$

- Linear combination

$$\alpha_1 \vec{v}_1 + \alpha_2 \vec{v}_2 + \alpha_3 \vec{v}_3 + \cdots + \alpha_M \vec{v}_M$$

- Length (norm)

$$\|\vec{v}_m\| = \sqrt{\left(v_m^{(1)}\right)^2 + \left(v_m^{(2)}\right)^2 + \cdots + \left(v_m^{(N)}\right)^2}$$

- Standard (natural) basis

$$\vec{e}_1 = [1,0,0, \dots, 0], \vec{e}_2 = [0,1,0, \dots, 0], \vec{e}_3 = [0,0,1, \dots, 0], \dots, \vec{e}_N = [0,0, \dots, 1]$$

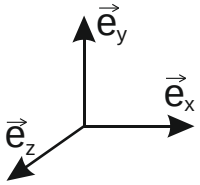
Vectors – scalar product

Let $\vec{a}, \vec{b}, \vec{e}_x, \vec{e}_y, \vec{e}_z \in R^3$ and $\vec{e}_x = [1,0,0], \vec{e}_y = [0,1,0], \vec{e}_z = [0,0,1]$, hence $\vec{a} = \left[a_x, a_y, a_z\right] = a_x\vec{e}_x + a_y\vec{e}_y + a_z\vec{e}_z$ and $\vec{b} = \left[b_x, b_y, b_z\right] = b_x\vec{e}_x + b_y\vec{e}_y + b_z\vec{e}_z$

Scalar product:

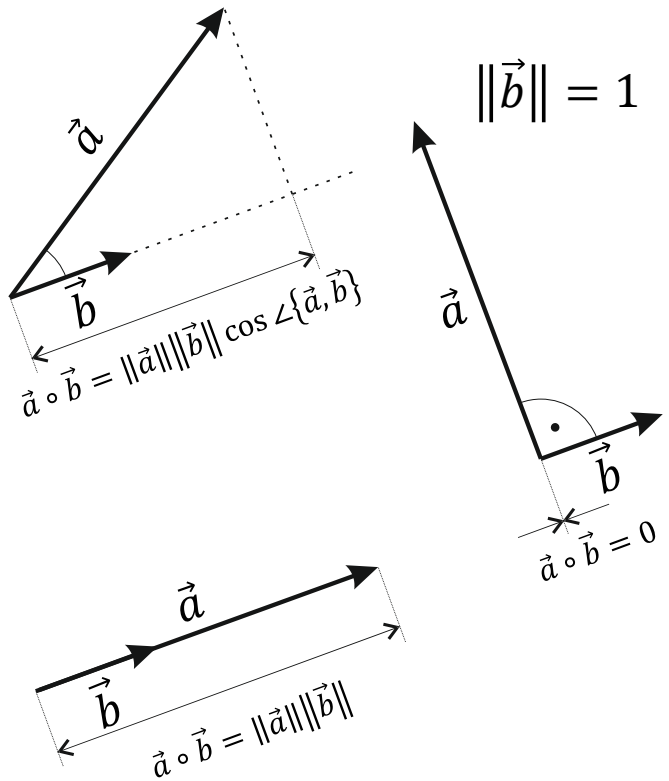
$$\vec{a} \circ \vec{b} = \|\vec{a}\|\|\vec{b}\| \cos \angle\{\vec{a}, \vec{b}\}$$

$$\vec{a} \circ \vec{b} = \left(a_x\vec{e}_x + a_y\vec{e}_y + a_z\vec{e}_z\right) \circ \left(b_x\vec{e}_x + b_y\vec{e}_y + b_z\vec{e}_z\right)$$



$$\begin{cases} \vec{e}_x \circ \vec{e}_x = 1, \vec{e}_x \circ \vec{e}_y = 0, \vec{e}_x \circ \vec{e}_z = 0, \\ \vec{e}_y \circ \vec{e}_x = 0, \vec{e}_y \circ \vec{e}_y = 1, \vec{e}_y \circ \vec{e}_z = 0, \\ \vec{e}_z \circ \vec{e}_x = 0, \vec{e}_z \circ \vec{e}_y = 0, \vec{e}_z \circ \vec{e}_z = 1, \end{cases}$$

$$\vec{a} \circ \vec{b} = a_x b_x + a_y b_y + a_z b_z$$

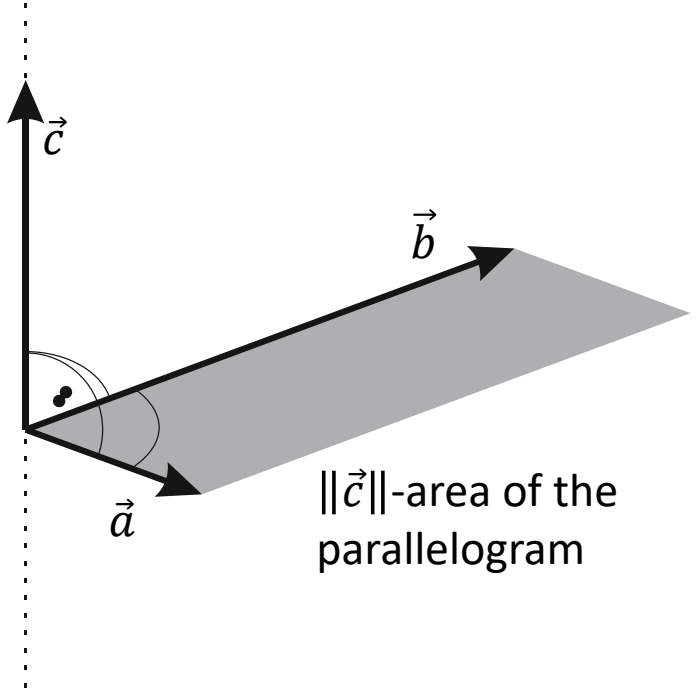


Vectors – cross product

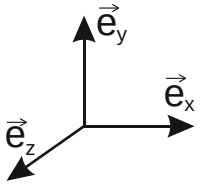
Cross product:

$$\vec{a} \times \vec{b} = \vec{c}$$

1. $\|\vec{c}\| = \|\vec{a}\|\|\vec{b}\| \, sin \angle\{\vec{a}, \vec{b}\}$
2. $\vec{c} \perp \vec{a}$ and $\vec{c} \perp \vec{b}$
3. *right-hand rule*



$$\vec{a} \times \vec{b} = \left(a_x\vec{e}_x + a_y\vec{e}_y + a_z\vec{e}_z\right) \times \left(b_x\vec{e}_x + b_y\vec{e}_y + b_z\vec{e}_z\right)$$



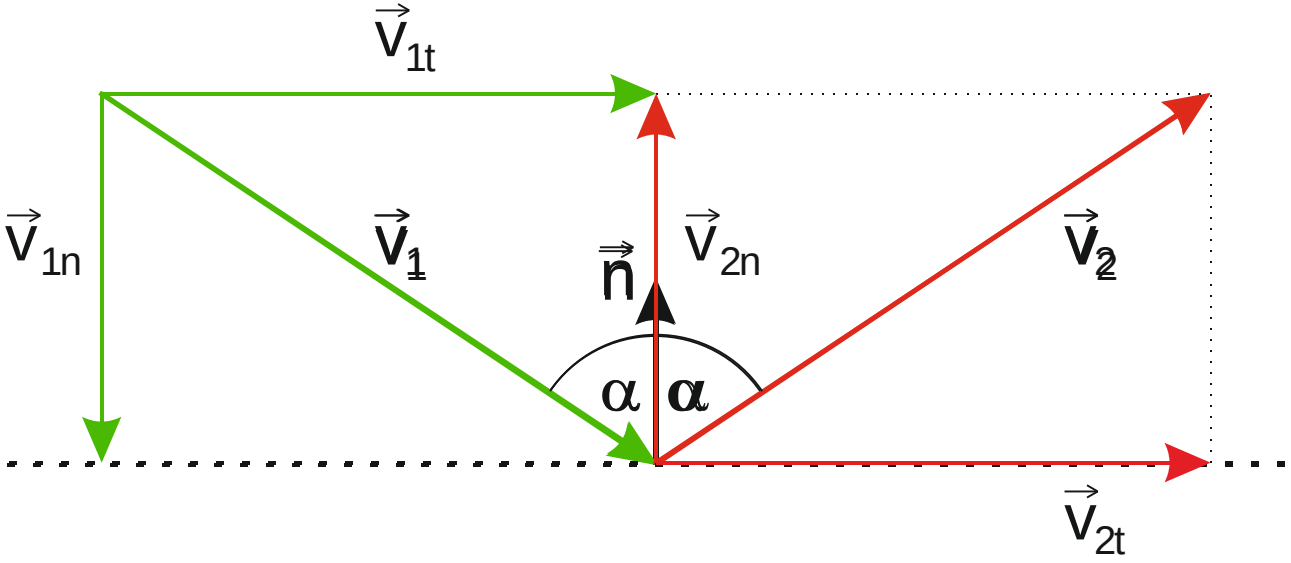
$$\begin{cases} \vec{e}_x \times \vec{e}_x = \vec{0}, \vec{e}_x \times \vec{e}_y = \vec{e}_z, \vec{e}_x \times \vec{e}_z = -\vec{e}_y, \\ \vec{e}_y \times \vec{e}_x = -\vec{e}_z, \vec{e}_y \times \vec{e}_y = \vec{0}, \vec{e}_y \times \vec{e}_z = \vec{e}_x, \\ \vec{e}_z \times \vec{e}_x = \vec{e}_y, \vec{e}_z \times \vec{e}_y = -\vec{e}_x, \vec{e}_z \times \vec{e}_z = \vec{0}, \end{cases}$$

$$\vec{a} \times \vec{b} = \left(a_y b_z - a_z b_y\right)\vec{e}_x - (a_x b_z - a_z b_x)\vec{e}_y + (a_x b_z - a_z b_x)\vec{e}_z = \begin{vmatrix} \vec{e}_x & \vec{e}_y & \vec{e}_z \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

Vectors – example

Problem: The ball bounces off a flat surface (perfectly elastic collision). Find the velocity vector after the collision if the incident velocity is equal $\vec{v}_1 = [-0.5, 0.5, -2]$ and the surface is defined by equation $6x + 3y + 2z - 6 = 0$. The ball dimension can be neglected.

Solution: The angle of incidence is equal to the angle of reflection and the speed value will not change (the impact is ideally elastic). Hence, the problem boils down to find the vector which is symmetrical to the normal vector.



$$\vec{v}_{2n} = -\vec{v}_{1n}$$
$$\vec{v}_{2t} = \vec{v}_{1t}$$

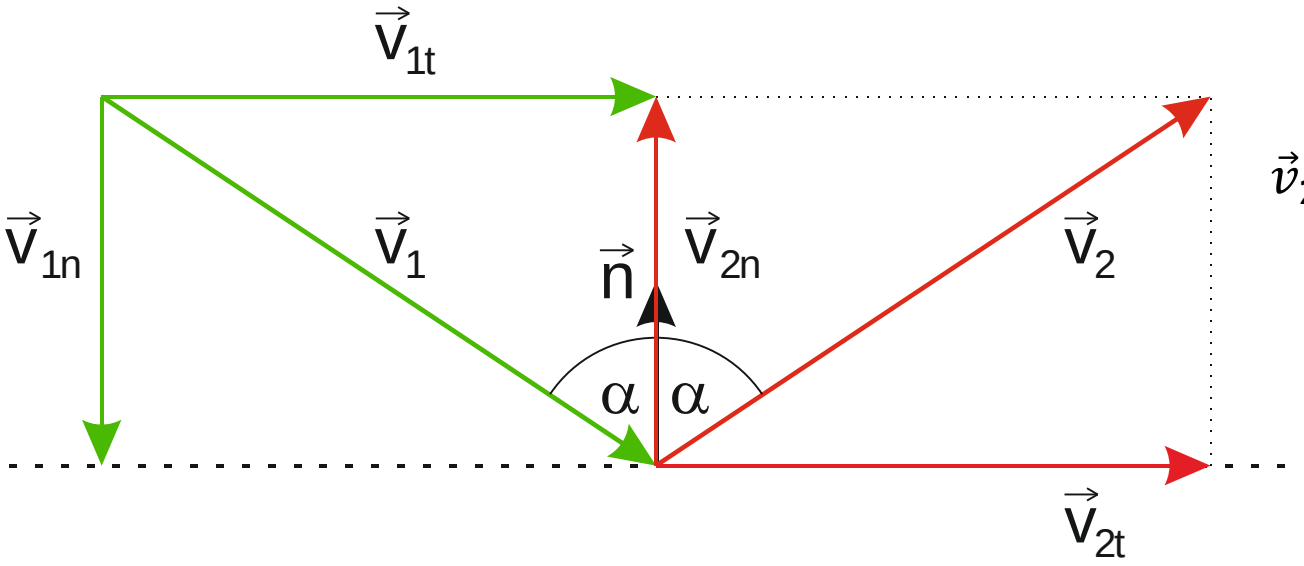
$$\vec{v}_{2n} = (\vec{v}_2 \circ \vec{n})\vec{n}$$
$$\vec{v}_{1n} = \left(\vec{v}_1 \circ (-\vec{n})\right)(-\vec{n}) = (\vec{v}_1 \circ \vec{n})\vec{n}$$
$$\|\vec{n}\| = 1$$

$$\vec{v}_{2t} = \vec{v}_2 - \vec{v}_{2n}$$
$$\vec{v}_{1t} = \vec{v}_1 - \vec{v}_{1n}$$

Vectors – example

Problem: The ball bounces off a flat surface (perfectly elastic collision). Find the velocity vector after the collision if the incident velocity is equal $\vec{v}_1 = [-0.5, 0.5, -2]$ and the surface is defined by equation $6x + 3y + 2z - 6 = 0$. The ball dimension can be neglected.

Solution: Finally, to solve the problem we need the incident velocity vector and the vector normal to the surface.



$$\vec{v}_{2n} = -\vec{v}_{1n} = -(\vec{v}_1 \circ \vec{n})\vec{n}$$

$$\vec{v}_{2t} = \vec{v}_{1t} = \vec{v}_1 - \vec{v}_{1n} = \vec{v}_1 - (\vec{v}_1 \circ \vec{n})\vec{n}$$

$$\vec{v}_2 = \vec{v}_{2n} + \vec{v}_{2t} =$$
$$= \vec{v}_1 - 2(\vec{v}_1 \circ \vec{n})\vec{n}$$

Vectors – example

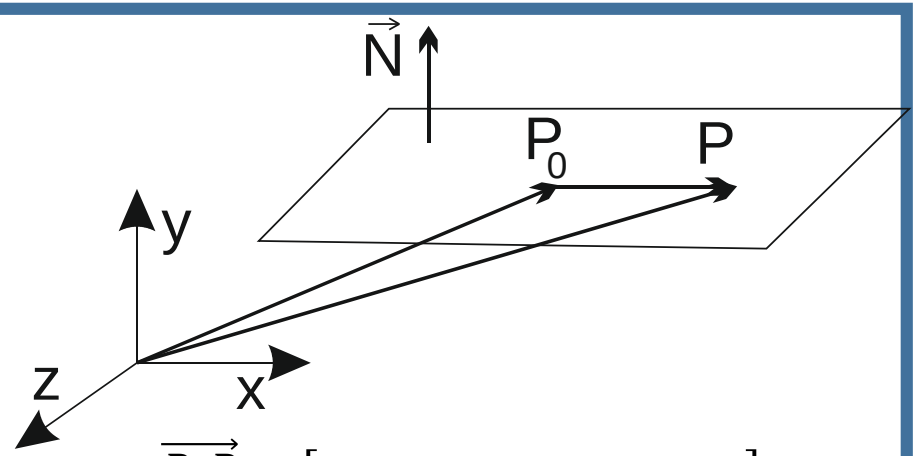
Problem: The ball bounces off a flat surface (perfectly elastic collision). Find the velocity vector after the collision if the incident velocity is equal $\vec{v}_1 = [-0.5, 0.5, -2]$ and the surface is defined by equation $6x + 3y + 2z - 6 = 0$. The ball dimension can be neglected.

Solution: The normal vector can be determined directly from the equation of the plane.

$$\vec{N} = [6,3,2]$$

$$\vec{n} = \frac{\vec{N}}{\|\vec{N}\|} = \frac{[6,3,2]}{7} = \left[\frac{6}{7}, \frac{3}{7}, \frac{2}{7}\right]$$

$\vec{N} = [A, B, C]$
$P_0 = (x_0, y_0, z_0)$
$P = (x, y, z)$



$\vec{N} \circ \overrightarrow{P_0P} = 0$ $\qquad$ $\overrightarrow{P_0P} = [x - x_0, y - y_0, z - z_0]$

$$A(x - x_0) + B(y - y_0) + C(z - z_0) = 0$$

$$Ax + By + Cz + D = 0$$

Vectors – example

Problem: The ball bounces off a flat surface (perfectly elastic collision). Find the velocity vector after the collision if the incident velocity is equal $\vec{v}_1 = [-0.5, 0.5, -2]$ and the surface is defined by equation $6x + 3y + 2z - 6 = 0$. The ball dimension can be neglected.

Solution: Finally, for

$$\vec{n} = \left[\frac{6}{7}, \frac{3}{7}, \frac{2}{7}\right]$$

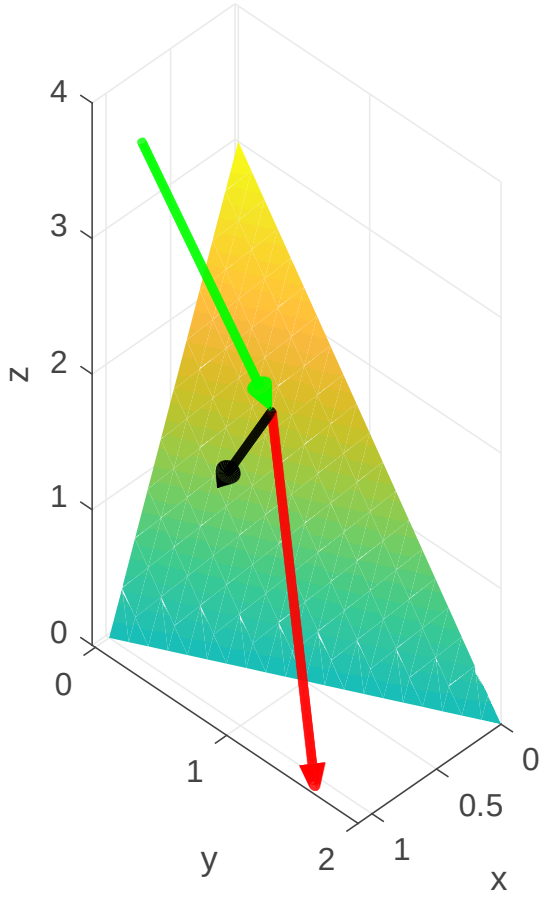
we obtain

$$\vec{v}_2 = \vec{v}_1 - 2(\vec{v}_1 \circ \vec{n})\vec{n} =$$

$$= \left[-\frac{1}{2}, \frac{1}{2}, -2\right] - 2\left(-\frac{6}{14} + \frac{3}{14} - \frac{8}{14}\right)\left[\frac{6}{7}, \frac{3}{7}, \frac{2}{7}\right] =$$

$$= \left[-\frac{1}{2}, \frac{1}{2}, -2\right] + \frac{11}{7}\left[\frac{6}{7}, \frac{3}{7}, \frac{2}{7}\right] = \left[-\frac{1}{2}, \frac{1}{2}, -2\right] + \left[\frac{66}{49}, \frac{33}{49}, \frac{22}{49}\right] =$$

$$= \left[-\frac{49}{98}, \frac{49}{98}, -\frac{196}{98}\right] + \left[\frac{132}{98}, \frac{66}{98}, \frac{44}{98}\right] = \left[\frac{83}{98}, \frac{115}{98}, -\frac{152}{98}\right]$$



Derivative of a function – introduction

What is a derivative of a function?

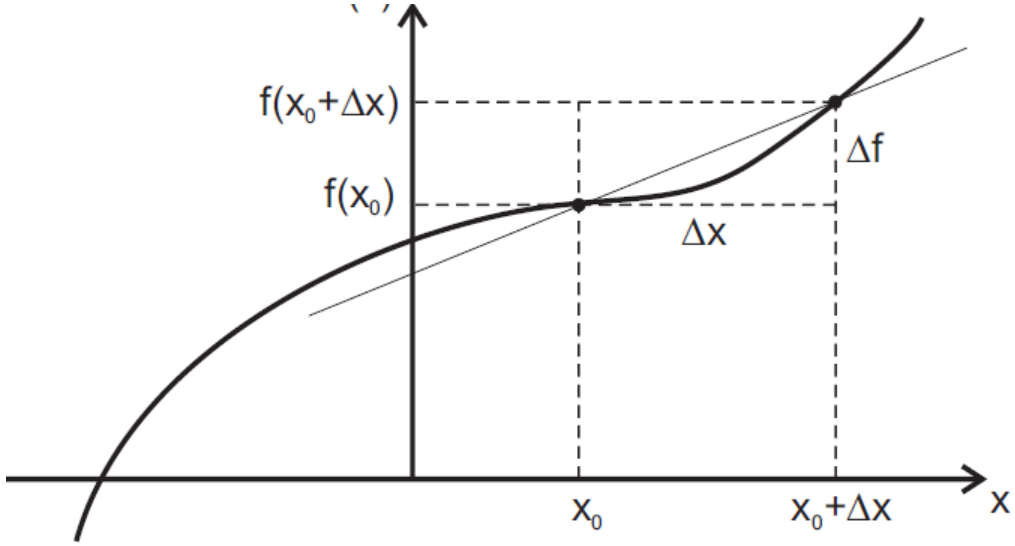
It measures the sensitivity to change of the function value with respect to a change in its argument. Derivatives are a fundamental tool of calculus!

How to calculate a derivative of a function?

It is not our problem, computer can do it faster and better. Usually, the brain is not needed for such process, so human can be replaced by a machine.

The problem is - why to calculate a derivative?

Derivative of a function – definition

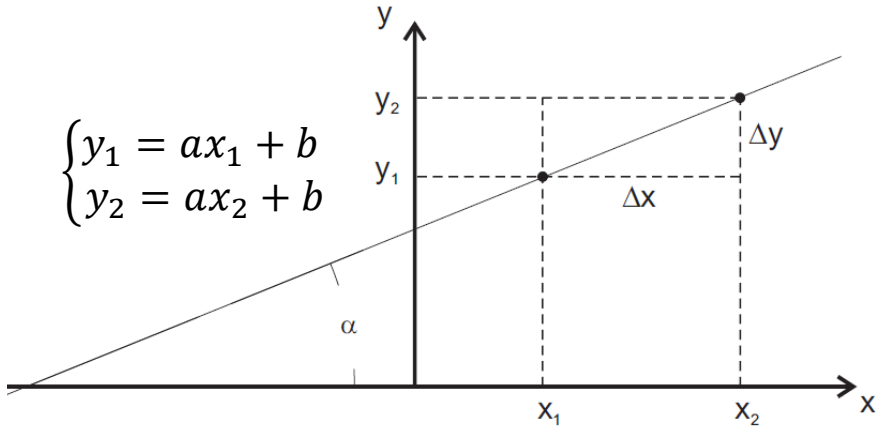


$$\lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x} = \frac{df}{dx} = f'(x_0)$$

The derivative of a function is the slope of the tangent line to the graph of the function at the fixed point x_0.

$$f'(x_0) = \tan \alpha(x_0) = a(x_0)$$

$$\begin{cases} y_1 = ax_1 + b \\ y_2 = ax_2 + b \end{cases}$$



Linear function $y = ax + b$

$$\tan \alpha = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{(ax_2 + b) - (ax_2 + b)}{x_2 - x_1} = a$$

a - slope of a line is a number that describes the steepness of the line

Derivative of a function – simple scheme

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{f\left(x_0 + \frac{\Delta x}{2}\right) - f\left(x_0 - \frac{\Delta x}{2}\right)}{\Delta x}$$

$$(x^n)' = nx^{n-1}$$

$$(\sin x)' = \cos x$$

$$(\ln x)' = \frac{1}{x}$$

$$(e^x)' = e^x$$

...

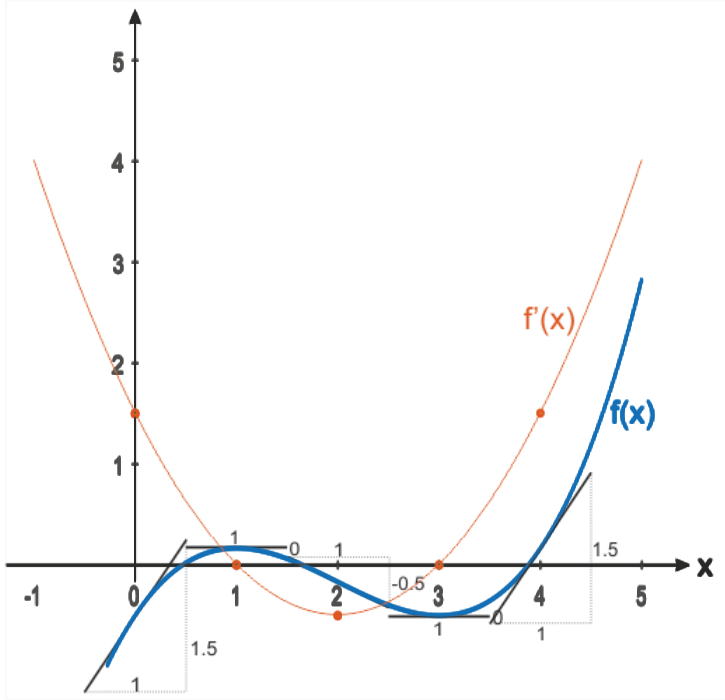
$$\big(f(g(x))\big)' = f'\big(g(x)\big)g'(x)$$

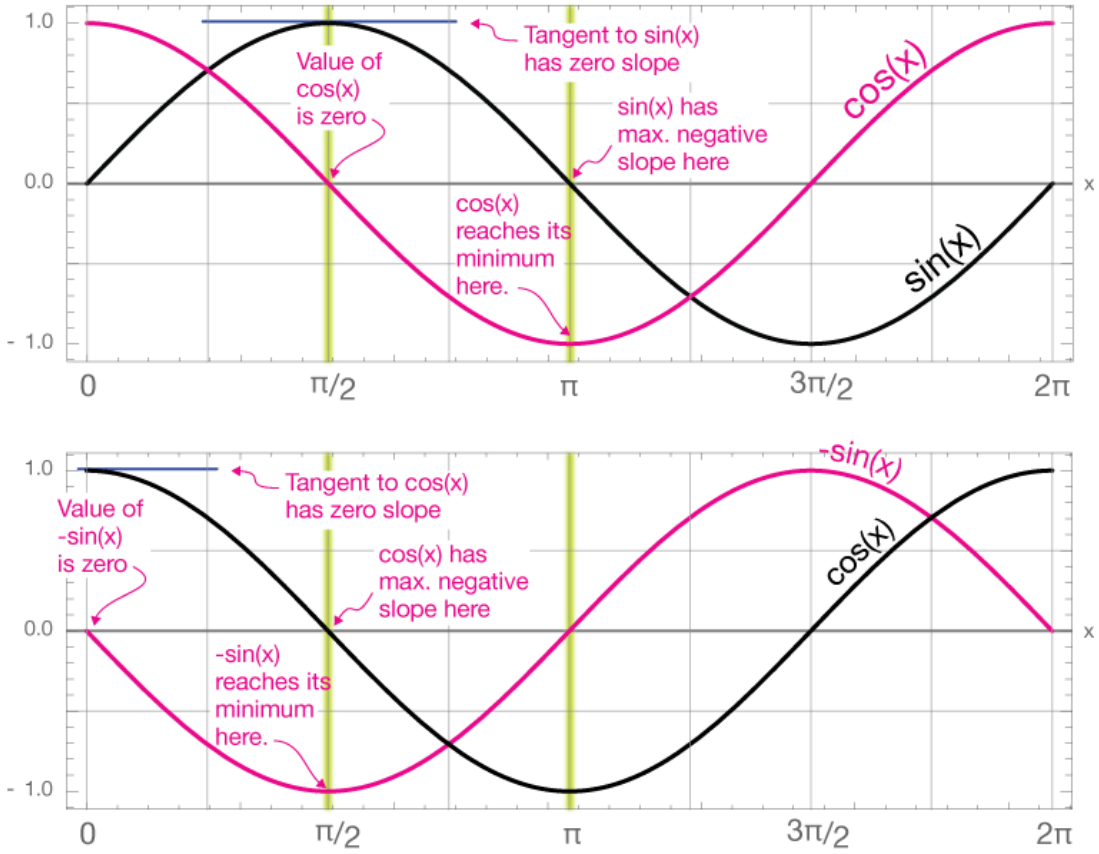
$$\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$$

$$\big(f(x)g(x)\big)' = f'(x)g(x) + f(x)g'(x)$$

Derivative of a function – monotonicity

To estimate a derivative of a function we need a pencil and a ruler

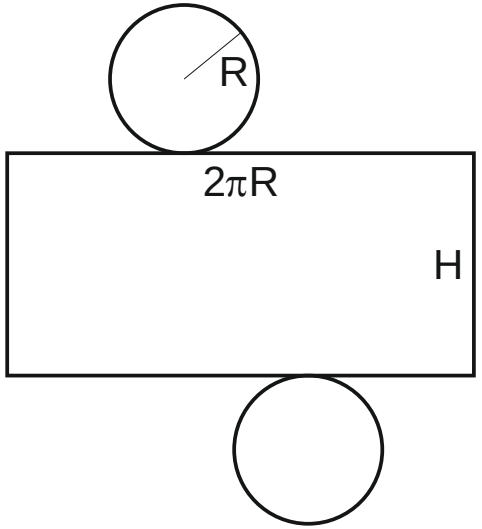




Source: http://www.drcruzan.com/DerivativesOfTrigFunc.html

Derivative of a function – optimization

Problem: A standard drum (metal barrel, capacity of 200 liters) must be light and chip. Are the dimensions of such container very important?



Solution: The weight and cost of production depend on the amount of metal used to make it! Hence, its total surface area is crucial!

$$\mathrm{P}(R, H) = 2\pi R^2 + 2\pi RH$$

Assuming $\mathrm{V} = \pi R^2 H = 200\ dm^3$, we get $H = \frac{200}{\pi R^2}$, so …

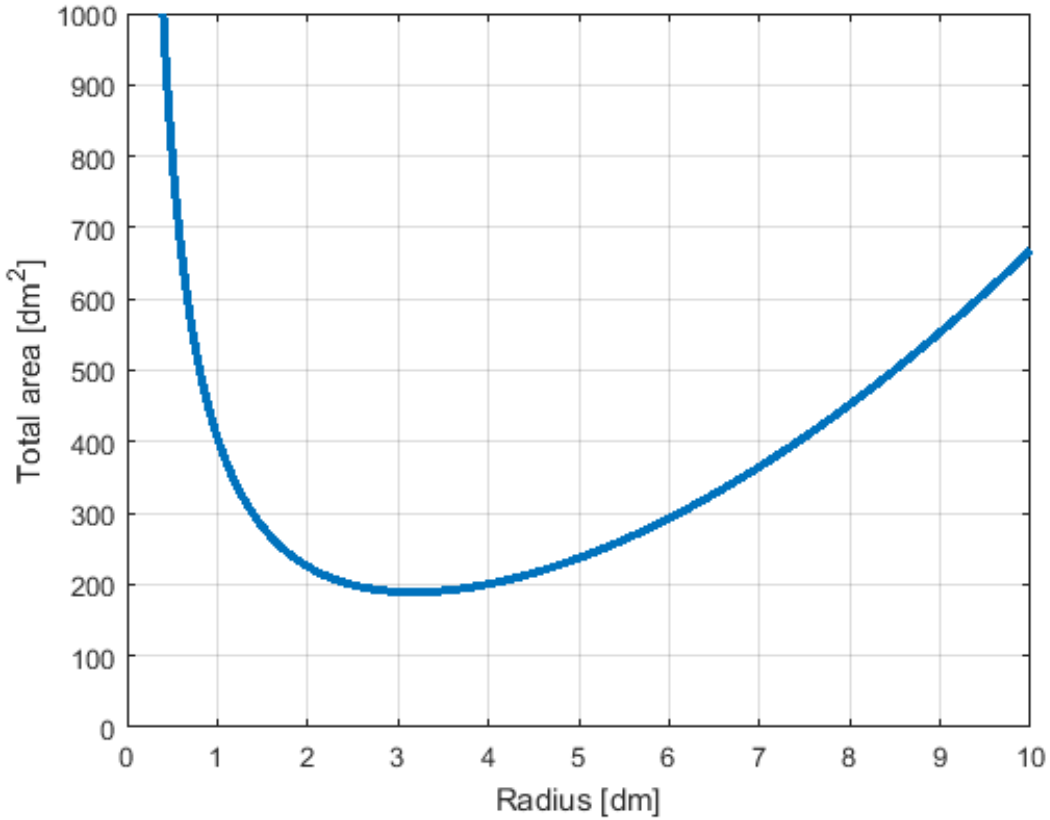
Derivative of a function – optimization

We need to find the smallest value of the function $\mathrm{P}(R) = 2\pi R^2 + \frac{400}{R}$

$$\mathrm{P}'(\mathrm{R}) = 4\pi R - \frac{400}{R^2}$$

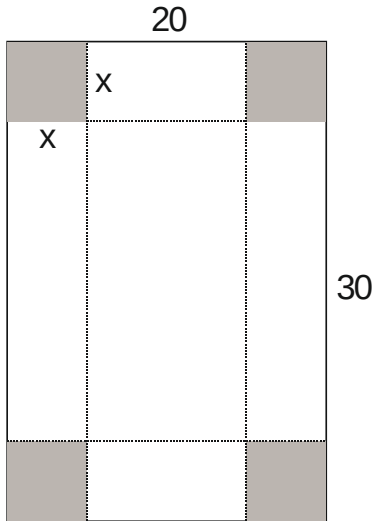
$$\mathrm{P}'(\mathrm{R}_0) = 4\pi R_0 - \frac{400}{R_0^2} = 0$$

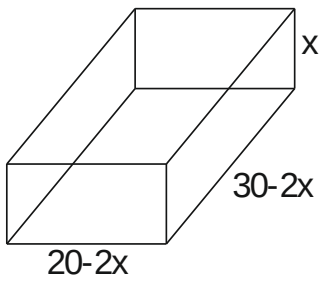
$$R_0 = \sqrt[3]{\frac{100}{\pi}} \approx 3.17\ dm$$



Derivative of a function – optimization

Problem: How big container can be built of a paper sheet 30x20cm (~A4)?





Solution: The volume of the container

$$V(x) = (20 - 2x)(30 - 2x)x$$

or in the simpler form

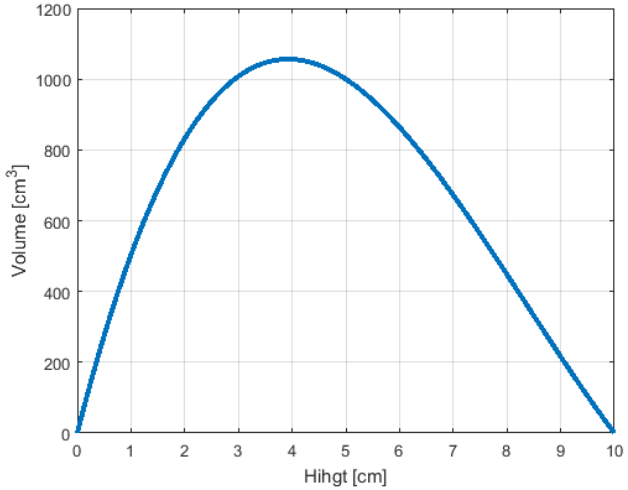
$$V(x) = 4x^3 - 100x^2 + 600x$$

The highest value of the function can be found from the derivative of the volume equal zero

$$V'(x_0) = 12x_0^2 - 200x_0 + 600 = 0$$

Finally

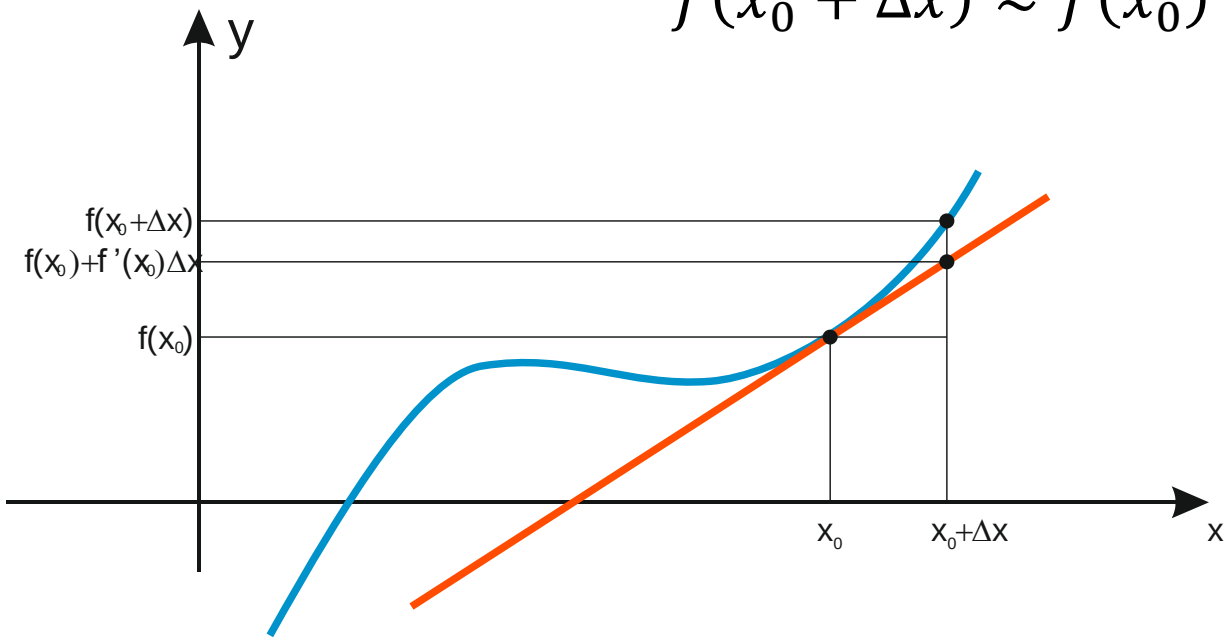
$$x_0 = \frac{25 - 5\sqrt{7}}{3} \approx 3.92\ cm$$



Derivative of a function – approximation

The tangent line is the best linear approximation of the function near the point x_0.
The derivative can be used to estimate a function value (linear approximation)

$$f(x_0 + \Delta x) \approx f(x_0) + f'(x_0)\Delta x$$



Example: $\sqrt{4.01} \approx ?$

$x_0 = 4, \Delta x = 0.01,$

$f(x_0) = \sqrt{x_0} = 2,$

$f'(x_0) = \frac{1}{2\sqrt{x_0}} = 0.25,$

$\sqrt{4.01} \approx 2 + 0.0025 = 2.0025$

Derivative of a function – approximation

What about: $\sqrt{3.55} = ?$, $\log 2.15 = ?$, $\sin\frac{\pi}{17} = ?$, $1.37^{1.26} = ?$, … ?

How do we know that? We can add, subtract, multiply, divide, but these … ?

Derivatives – Taylor series!!!

Taylor series is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a fixed point

$$f(x_0 + \Delta x) = f(x_0) + f'(x_0)\Delta x + f''(x_0)\frac{\Delta x^2}{2} + \cdots = \sum_{n=1}^{\infty} f^{(n)}(x_0)\frac{\Delta x^{\mathrm{n}}}{n!}$$

(in machines without hardware multipliers, for example in calculators, CORDIC algorithm is applied)

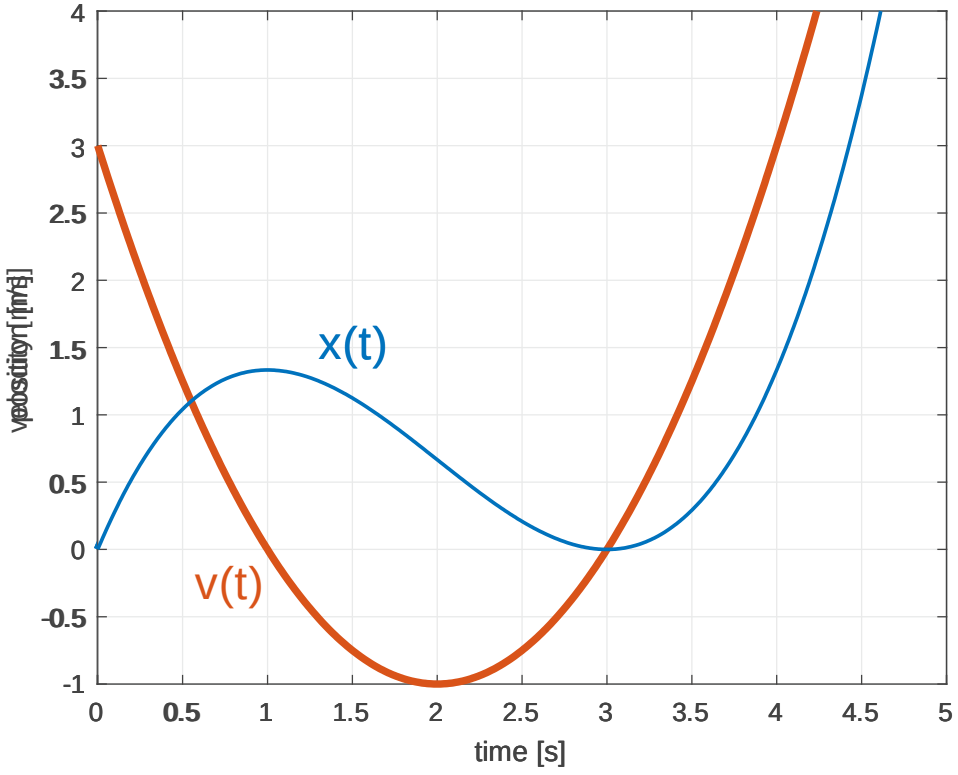
Derivative of a function – kinematics

Average velocity - rate of change of position Δx with respect to time Δt

$$\tilde{v} = \frac{\Delta x}{\Delta t}$$

Velocity (instantaneous) - is the limit of the rate of change of position over an infinitesimal interval of time

$$v(t_0) = \lim_{\Delta t \to 0} \frac{x(t_0 + \Delta t) - x(t_0)}{\Delta t} =$$
$$= \lim_{\Delta t \to 0} \frac{\Delta x}{\Delta t} = \frac{dx}{dt} = x'(t_0)$$



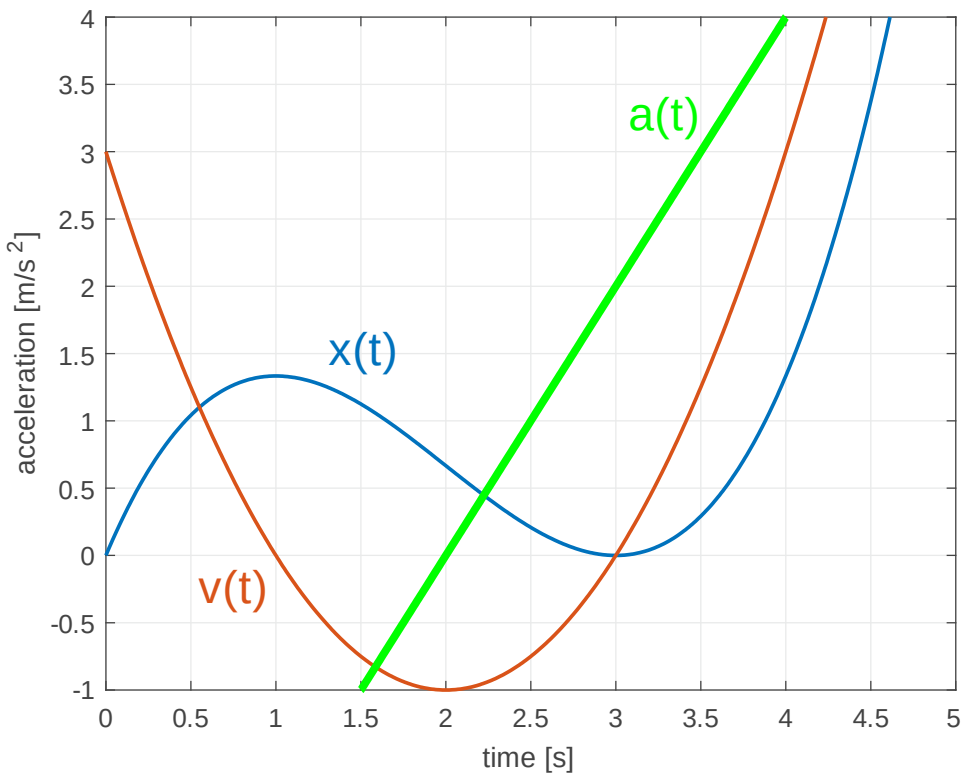
Derivative of a function – kinematics

Average acceleration - rate of change of velocity Δv with respect to time Δt

$$\tilde{a} = \frac{\Delta v}{\Delta t}$$

Acceleration (instantaneous) - is the limit of the rate of change of velocity over an infinitesimal interval of time

$$a(t_0) = \lim_{\Delta t \to 0} \frac{v(t_0 + \Delta t) - v(t_0)}{\Delta t} =$$
$$= \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t} = \frac{dv}{dt} = v'(t_0)$$



Derivative of a function – kinematics

Example of a constant acceleration

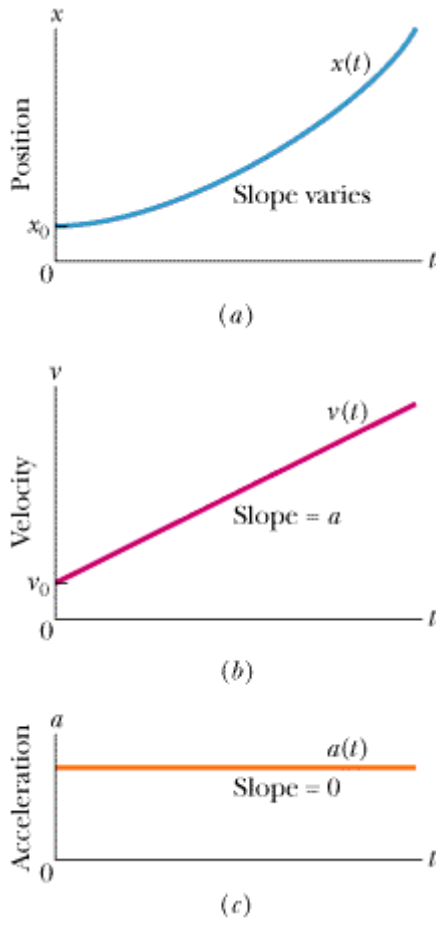
$$a(t) = a = const$$

then the velocity is

$$v(t) = at + v_0$$

and the position is

$$x(t) = \frac{1}{2}at^2 + v_0 t + x_0$$



Differential equation - one of the most powerful modeling tools

For algebraic equations (e.g. $x^2 + 3x - 4 = 0$) or for transcendental equations (e.g. $e^x = x^2 + 2$) the solutions are the values of the variable for which the equation is true.

For differential equation the solutions are the functions for which the differential equation is true.

For instance, the equation

$$f'(x) = 2f(x)$$

is satisfied by a set of functions

$$f(x) = Ce^{2x}$$

where C is an arbitrary number.

Without specification of initial/boundary conditions the solution is not unambiguous. For instance, if we impose initial condition $f(0) = 3$, the solution is

$$f(x) = 3e^{2x}$$

Differential equations - examples

Problem: A car with a turned off engine is rolling on a flat road. Due to air resistance and rolling friction the speed is decreasing from $v_1 = 108\ km/h$ to $v_2 = 72\ km/h$ in $\Delta t = 10\ s$. Find the motion resistance coefficient, if the mass of the car is $m = 1100\ kg$. What distance will the car travel in this time? Where and when it stops? What should be the engine power to reach maximum speed $v_{max} = 252\ km/h$.

Solution: According to the second Newton's law, the vector sum of the forces acting on an object is equal to the mass m of that object multiplied by the acceleration a of the object

$$ma(t) = F_{total}(t)$$

The only force acting on the car is motion resistance, which is proportional to its velocity (sign minus represents opposite directions of the velocity and the force)

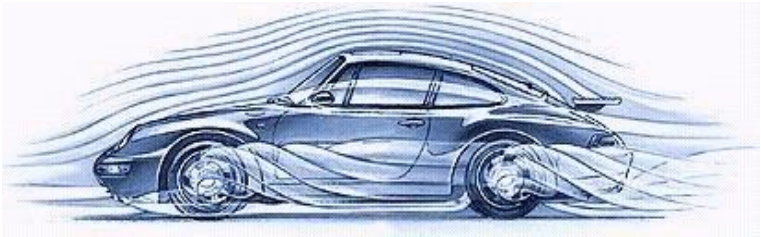
$$F_{total}(t) = -bv(t)$$

where b is the motion resistance coefficient. So, the differential equation governing of this motion is

$$ma(t) = -bv(t)$$

or in more useful form

$$v'(t) = -\frac{b}{m}v(t)$$



Source: http://www.autozine.org/technical_school/aero/tech_aero.htm

Differential s

Problem: A car with a tu air resistance and rolling friction the speed is decr h in $\Delta t = 10\ s$. Find the motion resistance coeffic What distance will the car travel in this time? Wher ne power to reach maximum speed $v_{max} = 252\ km/$

Solution: The solution of

using https://www.wolfr

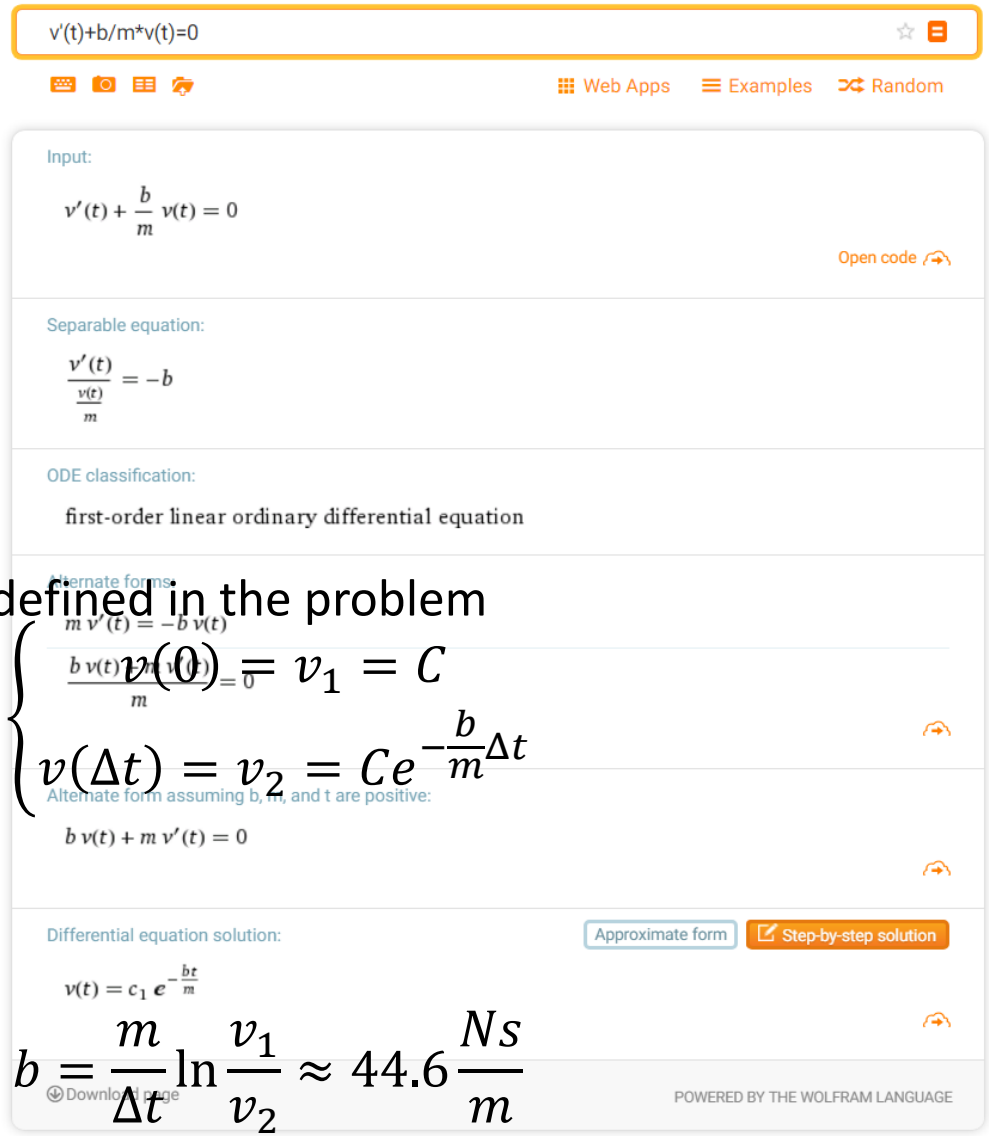
Applying the conditions defined in the problem

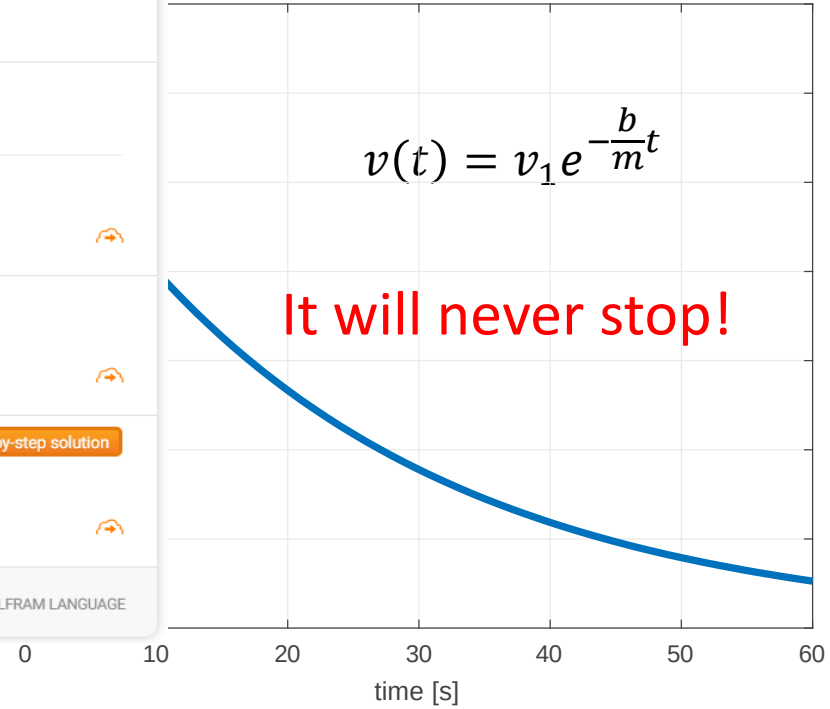
$$\begin{cases} v(0) = v_1 = C \\ v(\Delta t) = v_2 = Ce^{-\frac{b}{m}\Delta t} \end{cases}$$

we get $\frac{v_2}{v_1} = e^{-\frac{b}{m}\Delta t}$

Finally

$$b = \frac{m}{\Delta t}\ln\frac{v_1}{v_2} \approx 44.6\frac{Ns}{m}$$





Differential equations - examples

Problem: A car with a turned off engine is rolling on a flat road. Due to air resistance and rolling friction the speed is decreasing from $v_1 = 108\ km/h$ to $v_2 = 72\ km/h$ in $\Delta t = 10\ s$. Find the motion resistance coefficient, if the mass of the car is $m = 1100\ kg$. What distance will the car travel in this time? Where and when it stops? What should be the engine power to reach maximum speed $v_{max} = 252\ km/h$.

Solution: The distance can be evaluated from the velocity $v(t) = v_1 e^{-\frac{b}{m}t} = x'(t)$. Hence

$$x(t) = -\frac{mv_1}{b} e^{-\frac{b}{m}t} + C$$

Assuming that the position $x(t)$ is zero for $t = 0$

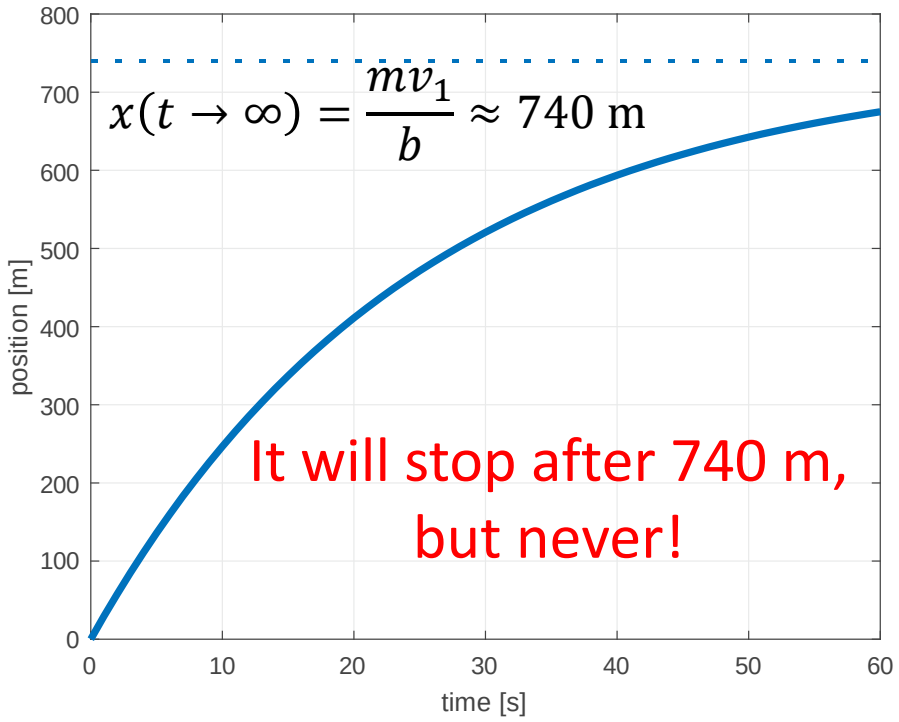
$$x(t = 0) = -\frac{mv_1}{b} + C = 0$$

and finally

$$x(t) = \frac{mv_1}{b}\left(1 - e^{-\frac{b}{m}t}\right)$$

In the first 10 seconds the distance is

$$\Delta x = x(\Delta t) - x(0) = \frac{mv_1}{b}\left(1 - e^{-\frac{b}{m}\Delta t}\right) \approx 246.6\ m$$



Differential equations - examples

Problem: A car with a turned off engine is rolling on a flat road. Due to air resistance and rolling friction the speed is decreasing from $v_1 = 108\ km/h$ to $v_2 = 72\ km/h$ in $\Delta t = 10\ s$. Find the motion resistance coefficient, if the mass of the car is $m = 1100\ kg$. What distance will the car travel in this time? Where and when it stops? What should be the engine power to reach maximum speed $v_{max} = 252\ km/h$.

Solution: As long as the force generated by the motor is greater than the motion resistance force, the car accelerates. However the resistance increases with the increase of speed. For the maximal speed the forces of the engine and the resistance are equal (magnitudes).

$$F_{engine} = -F_{resistance}$$

The engine power can be expressed as a product of vehicle velocity and generated force

$$P = vF_{engine}$$

For maximal speed it is

$$P = -v_{max}F_{resistance} = v_m^2 b$$

In this case

$$P \approx 218546\ W \approx 293hp$$

Differential equations - examples

Problem: In some isolated region lives a group of mice. Every day the population of mice increases by 1%. The same region is inhabited by a group of owls which eat 20 mice every day. How large will be the population of mice after two months (60 days), if today their number is $M_0 = 1000$.

Source: https://www.reddit.com/r/aww/comments/172xzt/owl_burrito/

Solution: Let $M(t)$ represents a number of mice (in time) and ΔM an increase of the population per day ($\Delta t = 1\text{day}$). Then

$$\frac{\Delta M}{\Delta t} = \frac{1}{100} M(t) - 20 \quad \rightarrow \quad \frac{dM}{dt} = \frac{1}{100} M(t) - 20$$

Such equation can be simply solved (variation of parameter, integrating factor)

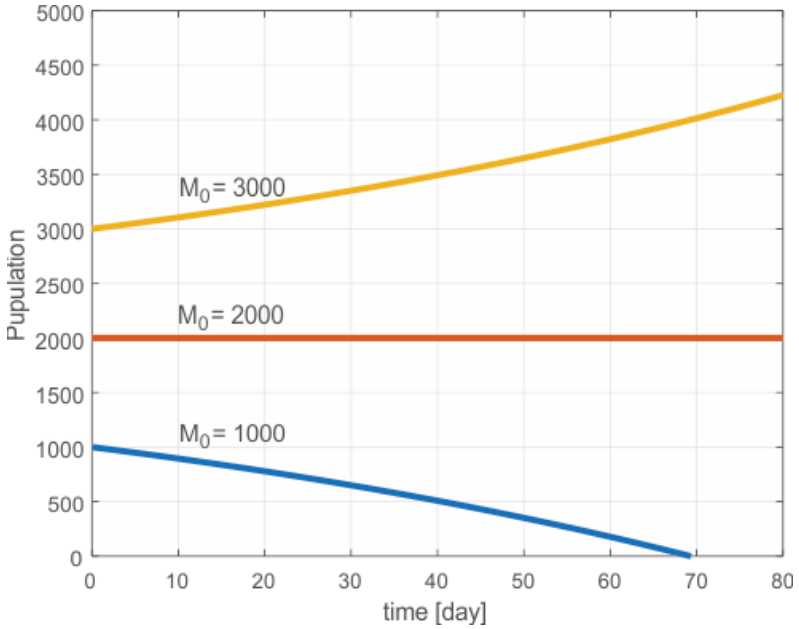
$$M(t) = 2000 + Ce^{\frac{1}{100}t}$$

From initial condition $M(0) = 2000 + C = M_0$ we get

$$M(t) = 2000 + (M_0 - 2000)e^{\frac{1}{100}t}$$

Finally

$$M(60) = 2000 - 1000e^{\frac{6}{10}} \approx 178$$



Differential equations - examples

Problem: What should be the constant average rate of drinking during a night of continual festivity to "not get drunk" (maintain stable level of alcohol in the blood)? The result depends on the individual person and requires an additional data. Therefore, lets consider the person which the level of alcohol decreases from $p_1 = 0.5$ ‰ to $p_2 = 0.35$ ‰ in $\Delta t = 1$ hour (when alcohol is not consumed). Assume that alcohol is detoxified in the liver at a rate proportional to its blood level.

Solution: Let $p(t)$ represent the level of alcohol in the blood. Then

$$\frac{dp}{dt} = -kp(t) + d$$

where k represents the rate of decay of alcohol per unit time and d the rate of intake per unit volume of the blood and per unit time. The general solution is

$$p(t) = \frac{d}{k} + Ce^{-kt}$$

and assuming that $p(t = 0) = 0$ we get

$$p(t) = \frac{d}{k}\left(1 - e^{-kt}\right)$$

For a longer period of time the desired level can be expressed by the following formula

$$p(t \to \infty) = \frac{d}{k} = p_\infty$$

Differential equations - examples

Problem: What should be the constant average rate of drinking during a night of continual festivity to "not get drunk" (maintain stable level of alcohol in the blood)? The result depends on the individual person and requires an additional data. Therefore, lets consider the person which the level of alcohol decreases from $p_1 = 0.5$ ‰ to $p_2 = 0.35$ ‰ in $\Delta t = 1$ hour (when alcohol is not consumed). Assume that alcohol is detoxified in the liver at a rate proportional to its blood level.

Solution: To determine the value of k, let us considered the case when alcohol is not consumed,

$$\frac{dp}{dt} = -kp(t)$$

Hence, $p(t) = Ce^{-kt}$ and from the conditions of the problem $\begin{cases} p(0) = p_1 = C \\ p(\Delta t) = p_2 = Ce^{-k\Delta t} \end{cases}$

we get $k = \frac{1}{\Delta \mathrm{t}} \ln \frac{p_1}{p_2} \approx 0.36 \frac{1}{h}$. For instance if we assume $p_\infty = 0.7‰ = \frac{d}{k}$ then $d = kp_\infty = 0.252 \frac{‰}{h}$. It is equivalent to $1.8\ ml$ of pure alcohol supplied to the blood (7 liters) within 1 hour. Taking into account that only 20% of the consumed alcohol is absorbed this value is about $9\ ml$ per hour.

Integral – introduction

What is an integral of a function?

Roughly speaking, the operation of (indefinite) integration is the reverse of differentiation!

How to calculate an integral of a function?

It is not our problem, computer can do it faster and better (like derivatives). Usually, the brain is not needed for such process, so once again human can be replaced by a machine.

The problem is - why to calculate an integral?

Integral – interpretation

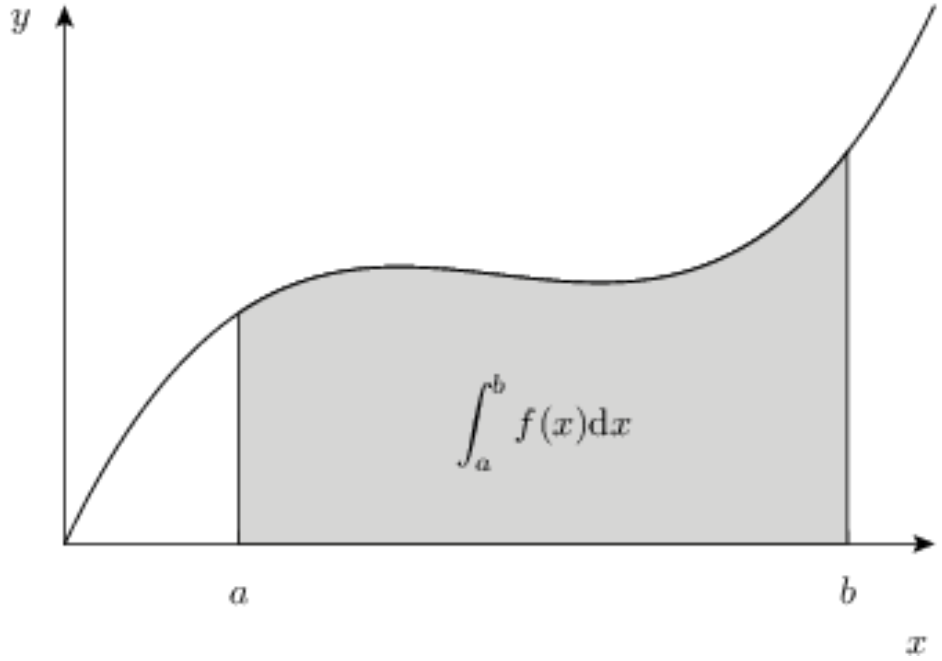
- indefinite integral represents the reverse of derivative

$$\int f(x)dx = F(x) \quad \Leftrightarrow \quad \mathrm{F}'(x) = f(x)$$

- definite integral represents the area of the region bounded by its graph

$$\int_a^b f(x)dx = P$$

Let's forget the fact that due to sign changes in a function this interpretation is not the most fortunate. But why the area? What does have in common reverse of derivative with the area?



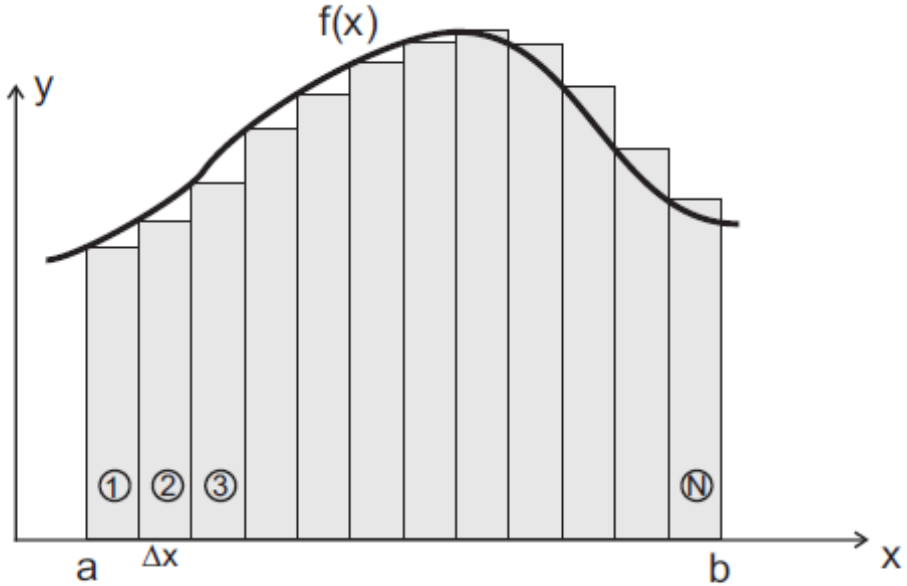
Integral – interpretation

$$P = f(a)\Delta x + f(a+\Delta x)\Delta x + f(a+2\Delta x)\Delta x + \cdots + f(b-3\Delta x)\Delta x + f(b-2\Delta x)\Delta x + f(b-\Delta x)\Delta x$$

$f(x) = F'(x)$ **antiderivative!**

$$f(x_0) = \frac{F(x_0+\Delta x) - F(x_0)}{\Delta \mathrm{x}}, \qquad \Delta x \to 0$$

$$f(x_0)\Delta \mathrm{x} = F(x_0+\Delta x) - F(x_0), \qquad \Delta x \to 0$$

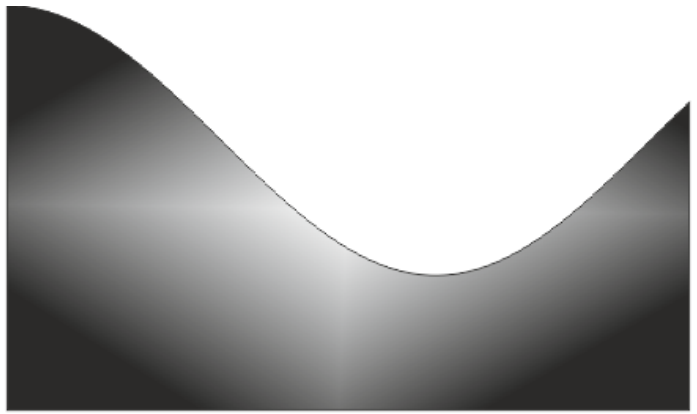


$$P = \boxed{F(a+\Delta x) - F(a)} + \boxed{F(a+2\Delta x) - F(a+\Delta x)} + \boxed{F(a+3\Delta x) - F(a+2\Delta x)} + \cdots$$
$$+F(b-2\Delta x) - F(\mathrm{b}-3\Delta x) + F(b-\Delta x) - F(\mathrm{b}-2\Delta x) + F(b) - F(\mathrm{b}-\Delta x)$$

$$= F(b) - F(a) = \int_a^b f(x)dx$$

Integral – examples

Problem: How much weighs a piece of steel sheet shown in the figure? We know that one square meter of this sheet weighs about 8 kg.

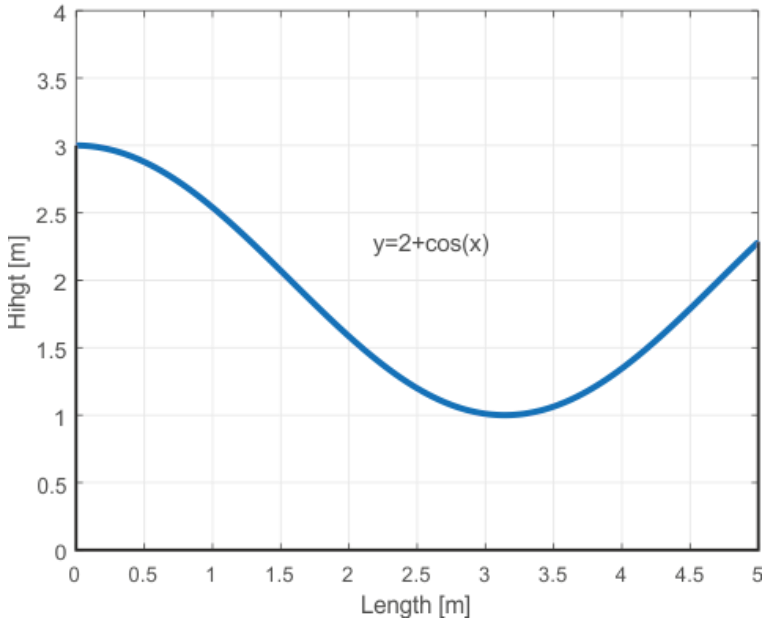


Solution: The area of the piece is

$$P = \int_0^5 (2 + \cos x)\, dx = [2x + \sin x]_0^5$$
$$= [2x + \sin x]^5 - [2x + \sin x]_0$$
$$= 10 + \sin 5 \approx 9.04m^2$$

Using the proportionality rule

$$m \approx 9.04m^2 \cdot 8kg/m^2 \approx 72.3kg$$

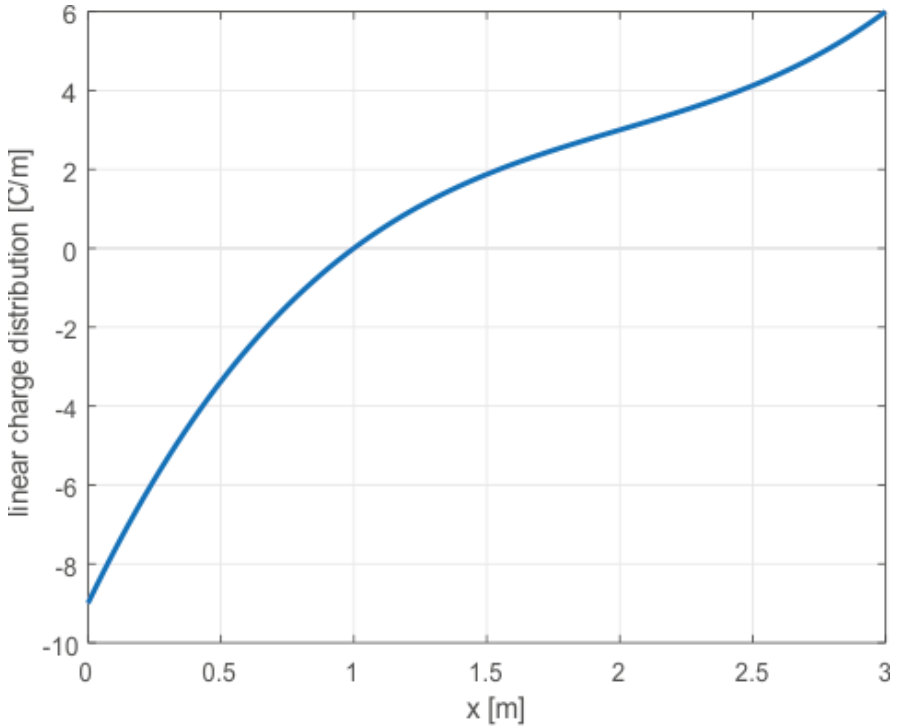


Integral – examples

Problem: The linear electric charge density on a thin fiber, located in $x \in [0,3]$, is expressed by

$$f(x) = x^3 - 6x^2 + 14x - 9\ C/m.$$

What is the total charge of the fiber?



Solution: If the charge distribution is constant the total charge is just a product of this constant charge density and the length of the fiber. Otherwise, the fiber must be divided into infinitesimal slices of length dx. At each slice the charge can be expressed by

$$dQ = f(x)dx$$

Hence, the total charge is the sum of all these charges

$$Q = \int_0^3 f(x)dx = \int_0^3 (x^3 - 6x^2 + 14x - 9)\ dx = \frac{9}{4}\ C$$

linear electric charge density
= charge/ infinitesimal length

Integral – examples

Problem: How to estimate an amount of water flowing through a pipe of known diameter (D=0.05m)? There are many methods to estimate the speed of the water (flow measurement) and we assume that this speed in function of time is known (presented in the graph).

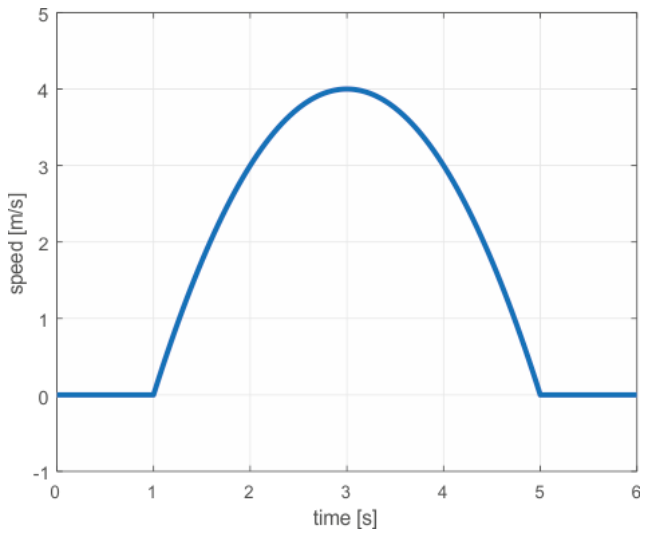
Solution: The flow (m^3/s) is a product of speed and cross section area of the pipe

$$\Phi(\mathrm{t}) = s \cdot v(t) = \frac{\pi D^2}{4} \cdot v(t)$$

where $v(t) = -t^2 + 6t - 5$ for $t \in (1,5)$ and zero elsewhere.

The total amount of water can be estimated by integration

$$V = \int_1^5 \Phi(\mathrm{t})dt = \frac{\pi D^2}{4} \int_1^5 (-t^2 + 6t - 5)\, dt = \frac{\pi D^2}{4} \frac{32}{3}$$
$$\approx 0.021 m^3$$



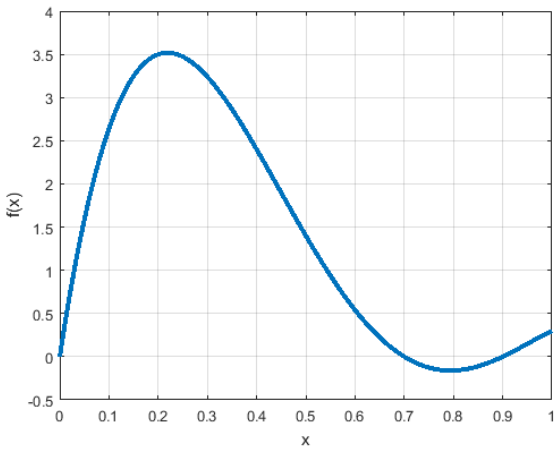
Source: https://www.youtube.com/watch?v=5sHjGtvE-Gc

Modeling and Simulation of Systems

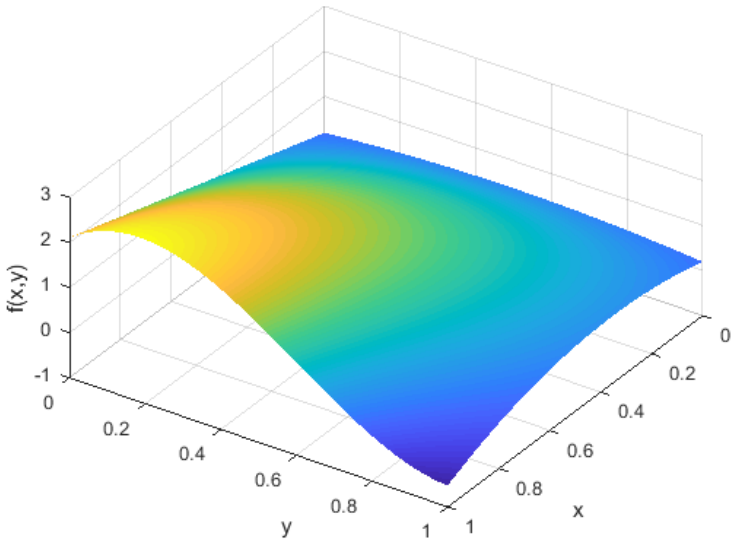
Lecture 3: Mathematics - basic modeling tool, part II

Piotr Kowalczyk

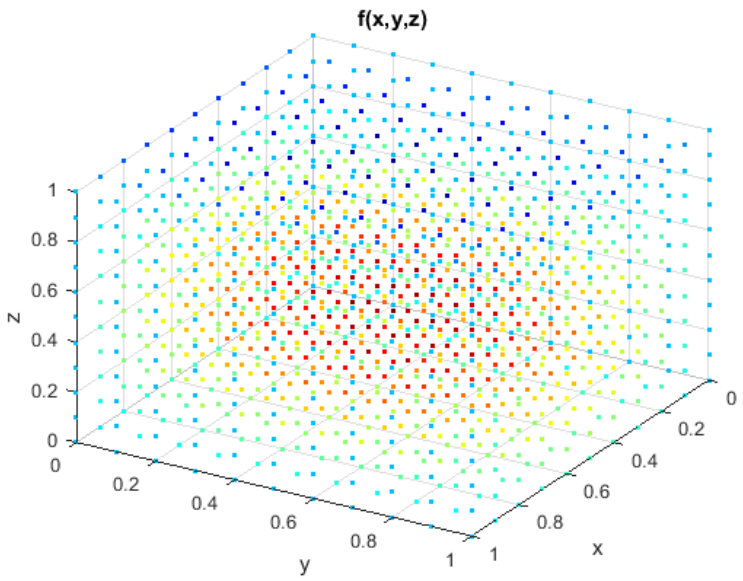
Multivariable functions – scalar function/field



$f(x) = x^3 + 2x^2 - x$



$f(x, y) = x^3y + 2x^2y^2 + 2$



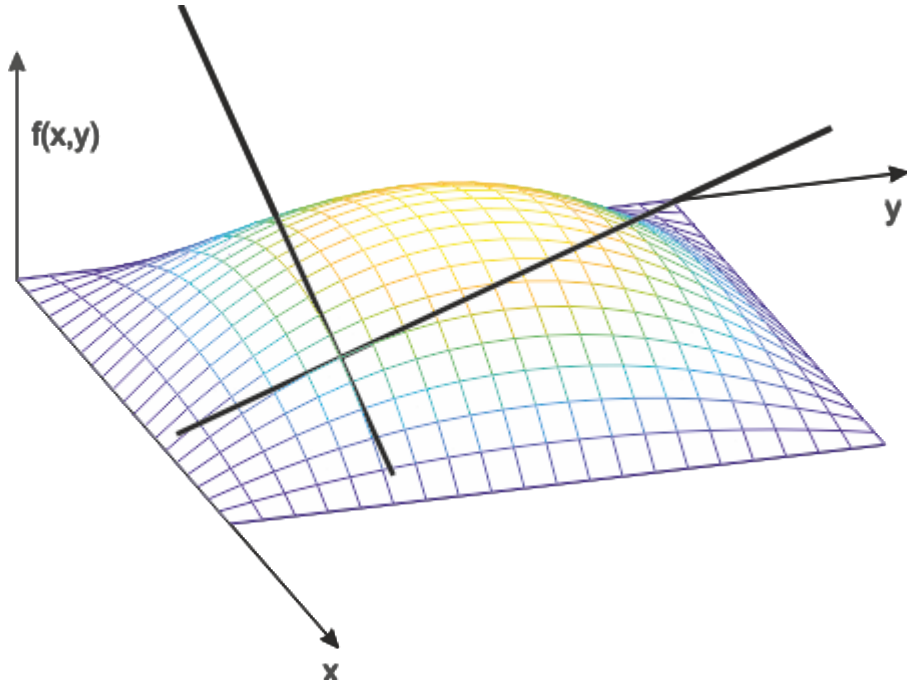
$f(x, y, z) = xyz + 2x^2y - \mathrm{z}$

For instance: temperature of bar, plate and block

Partial derivatives – interpretation

$$\frac{\partial f}{\partial x}(x_0, y_0) = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x, y_0) - f(x_0, y_0)}{\Delta x}$$

$$\frac{\partial f}{\partial y}(x_0, y_0) = \lim_{\Delta y \to 0} \frac{f(x_0, y_0 + \Delta y) - f(x_0, y_0)}{\Delta y}$$



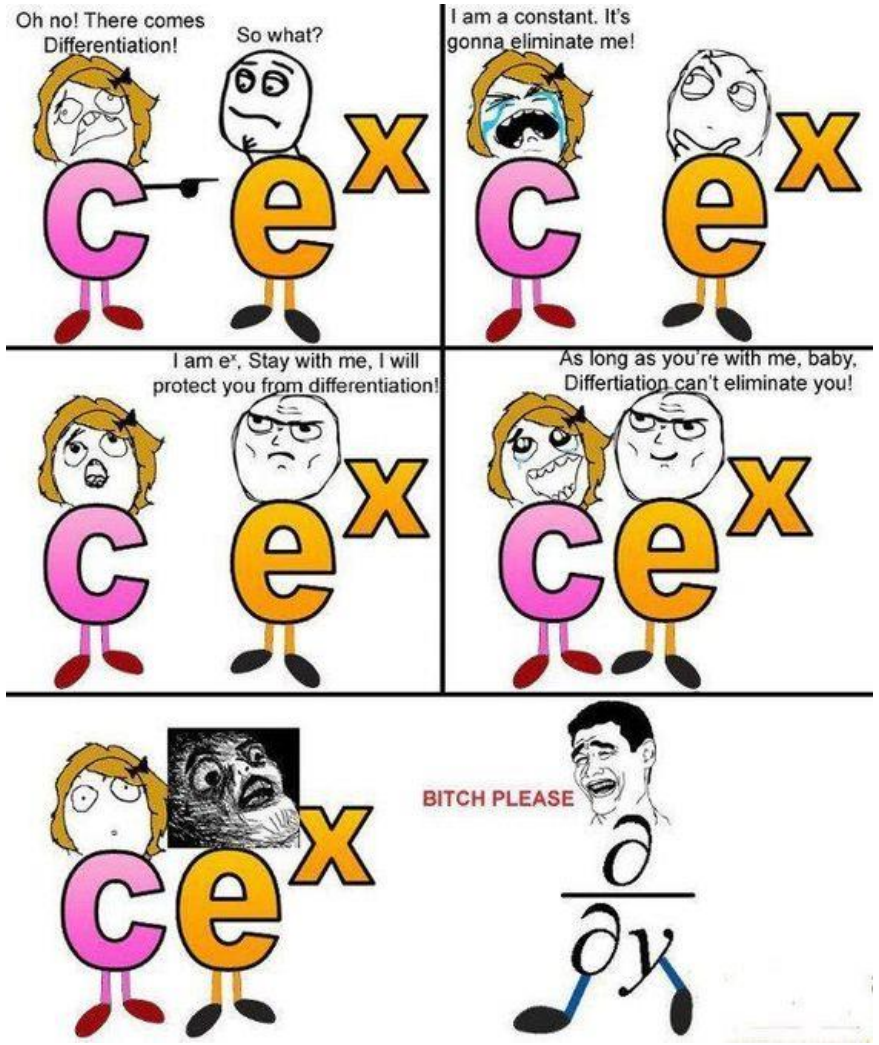
Partial derivatives – interpretation

Example: $f(x,y) = x^2 + xy - y^3$

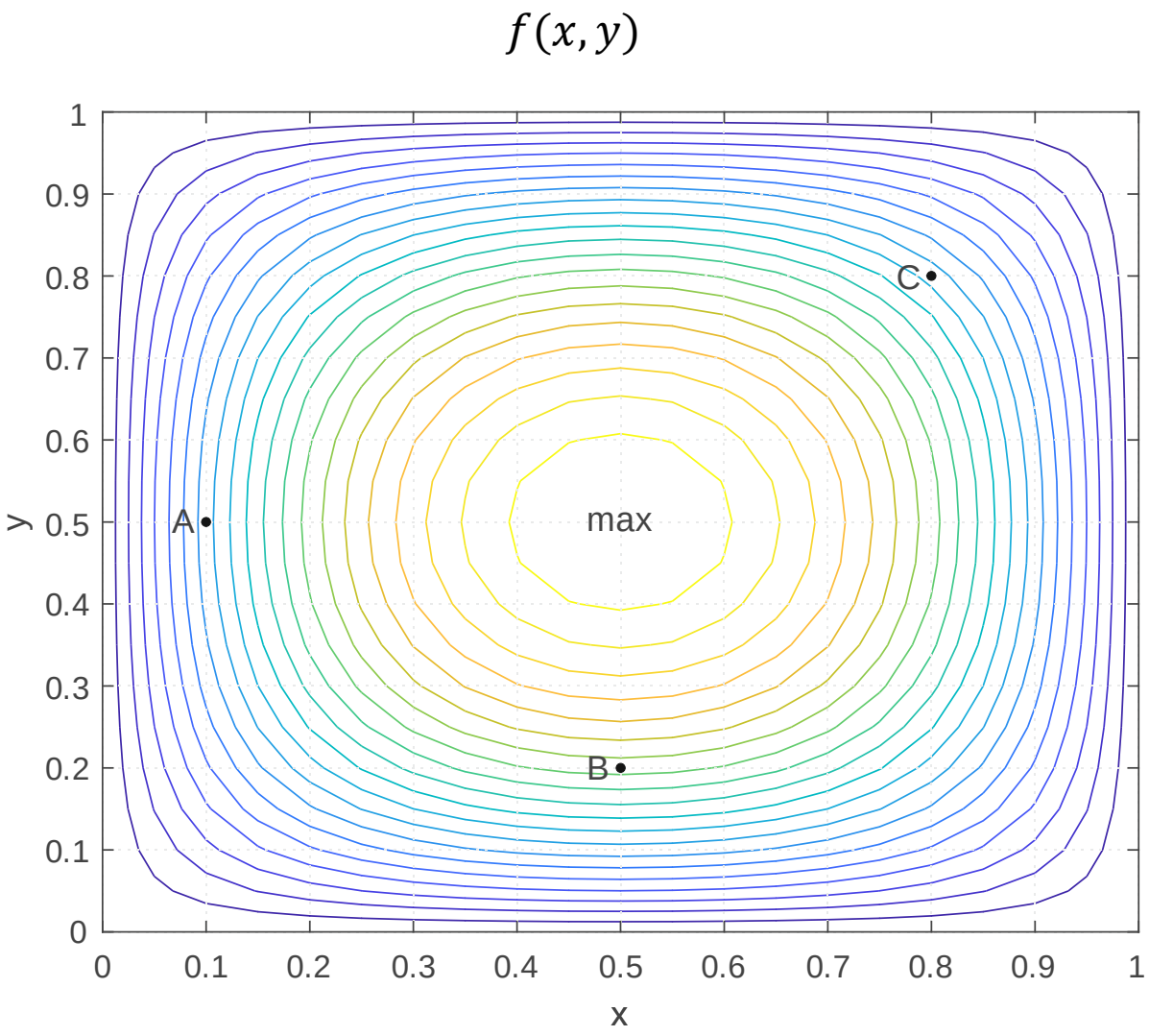
$$\frac{\partial f}{\partial x} = 2x + y - 0 \qquad \frac{\partial f}{\partial y} = 0 + x - 3y^2$$

Example: $f(x,y) = xsin(y) + 5e^x$

$$\frac{\partial f}{\partial x} = sin(y) + 5e^x \qquad \frac{\partial f}{\partial y} = xcos(y) + 0$$



Partial derivatives – interpretation



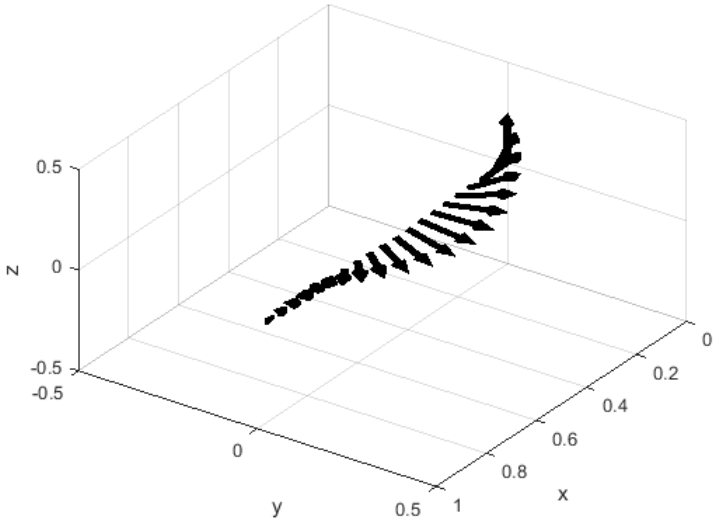
$>,\ \approx,\ <$???

$$\frac{\partial f}{\partial x}(A) \overset{>}{\ldots} 0 \qquad \frac{\partial f}{\partial y}(A) \overset{\approx}{\ldots} 0$$

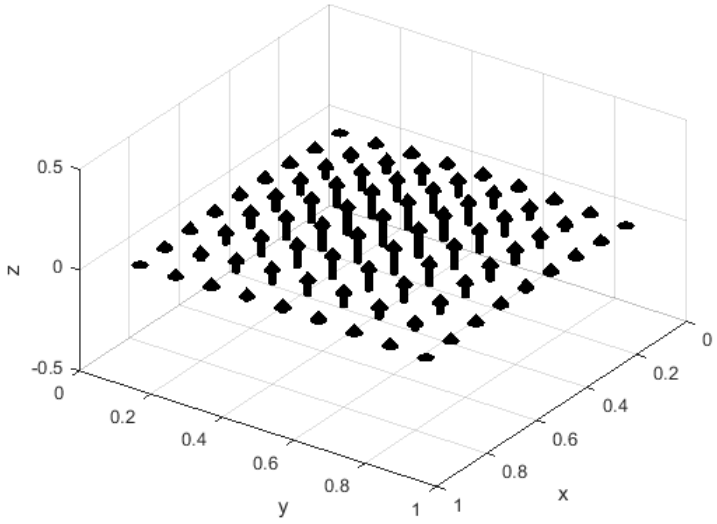
$$\frac{\partial f}{\partial x}(B) \overset{\approx}{\ldots} 0 \qquad \frac{\partial f}{\partial y}(B) \overset{>}{\ldots} 0$$

$$\frac{\partial f}{\partial x}(C) \overset{<}{\ldots} 0 \qquad \frac{\partial f}{\partial y}(C) \overset{<}{\ldots} 0$$

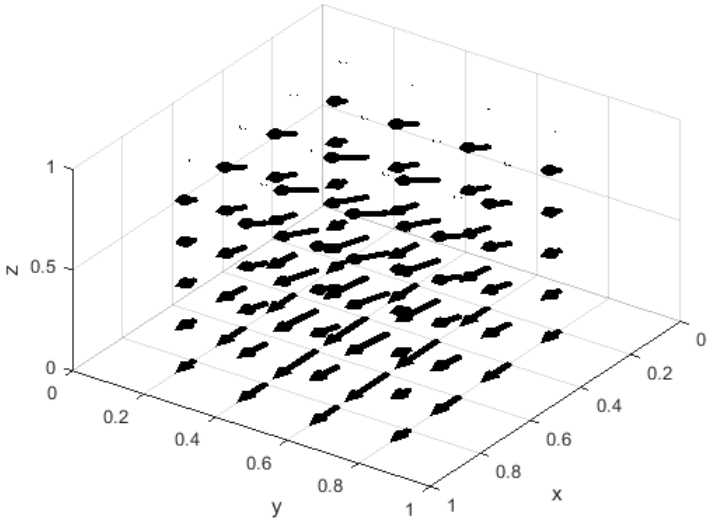
Multivariable functions – vector function/field



$\vec{F}(x) = [1 - \mathrm{x}, \sin x, \cos x]$



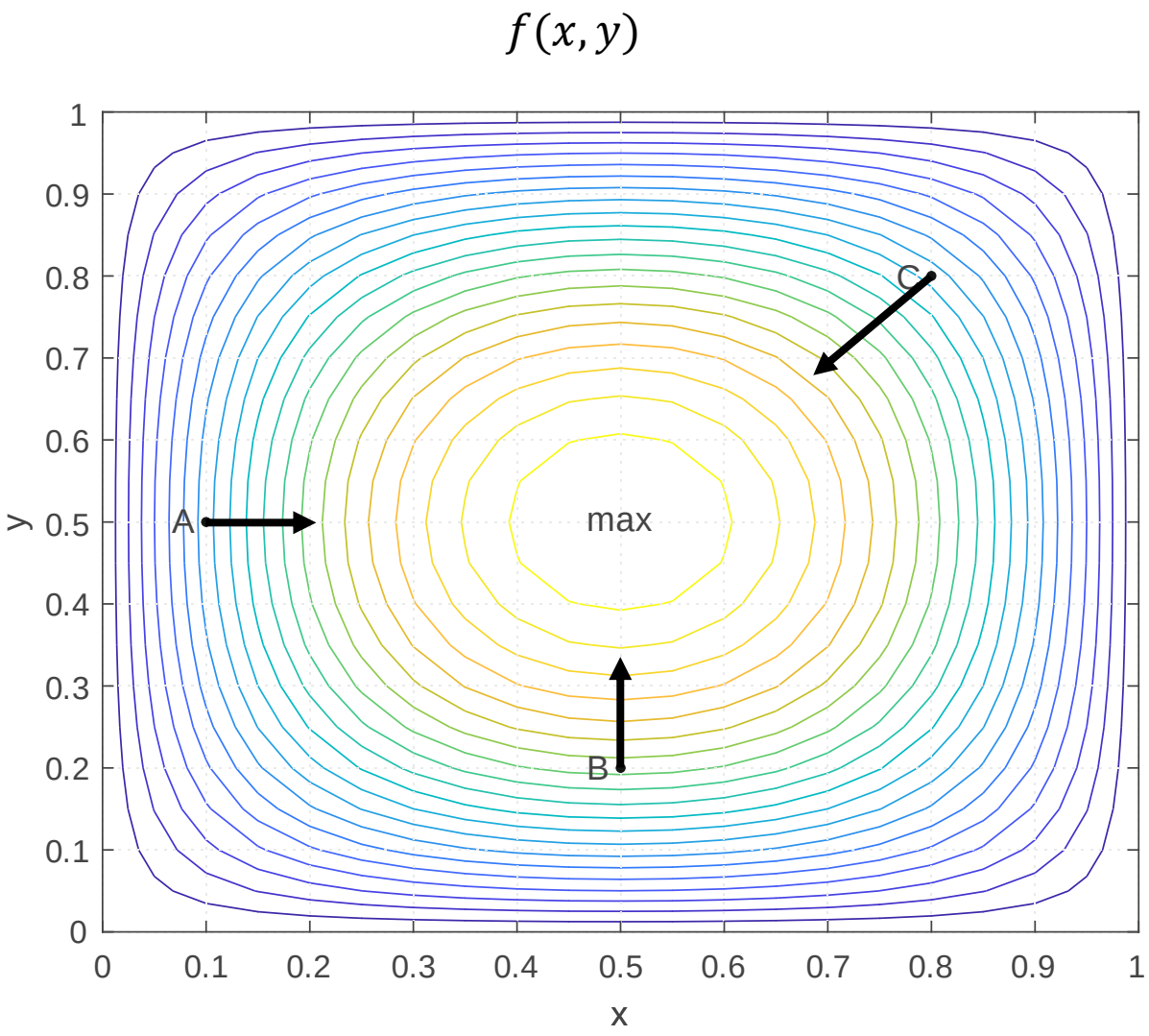
$\vec{F}(x, y) = [yx^2, y + x, (x^2 - x\,)(y^2 - y)]$



$\vec{F}(x, y, z) = [x^2 + y, y + z^2, xyz]$

For instance: forces acting on a bar, velocity of water particles in a square pipe, wind

Differential operators – gradient



$$\text{grad}\, f(x,y) = \frac{\partial f}{\partial x}\,\vec{e}_x + \frac{\partial f}{\partial y}\vec{e}_y$$

$$\frac{\partial f}{\partial x}(A) > 0 \qquad \frac{\partial f}{\partial y}(A) = 0$$

$$\frac{\partial f}{\partial x}(B) = 0 \qquad \frac{\partial f}{\partial y}(B) > 0$$

$$\frac{\partial f}{\partial x}(C) < 0 \qquad \frac{\partial f}{\partial y}(C) < 0$$

$$\text{grad}\, f(x,y) = \vec{F}(x,y)$$

$$\text{scalar function} \xrightarrow{\text{grad}} \text{vector field}$$

gradient points in the direction of greatest rate of increase of the function

Differential operators – gradient

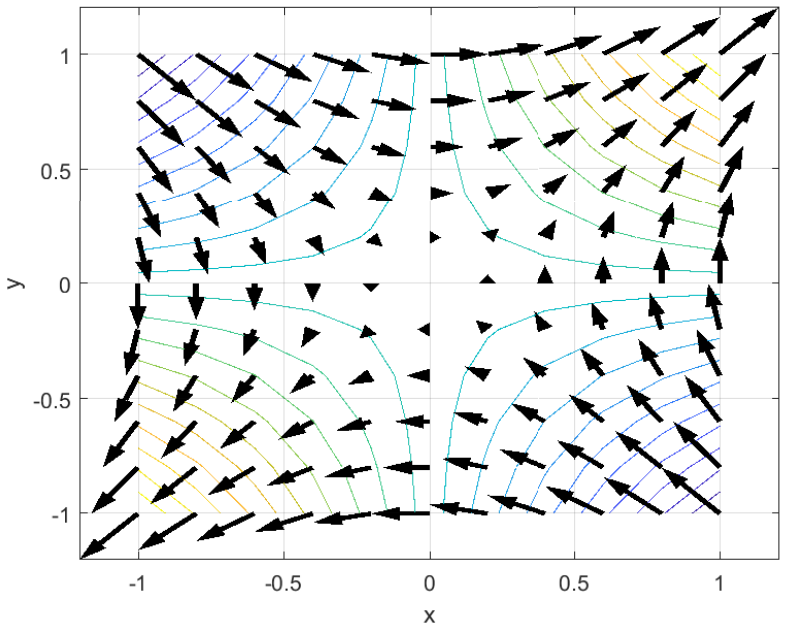
$$\text{grad}\, f(x,y,z) = \frac{\partial f}{\partial x}\,\vec{e}_x + \frac{\partial f}{\partial y}\,\vec{e}_y + \frac{\partial f}{\partial z}\,\vec{e}_z = \nabla f(x,y,z)$$

Del or **nabla**

$$\nabla = \frac{\partial}{\partial x}\,\vec{e}_x + \frac{\partial}{\partial y}\,\vec{e}_y + \frac{\partial}{\partial z}\,\vec{e}_z = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right]$$

Example: Find gradient of $f(x,y,z) = xy + z^2$

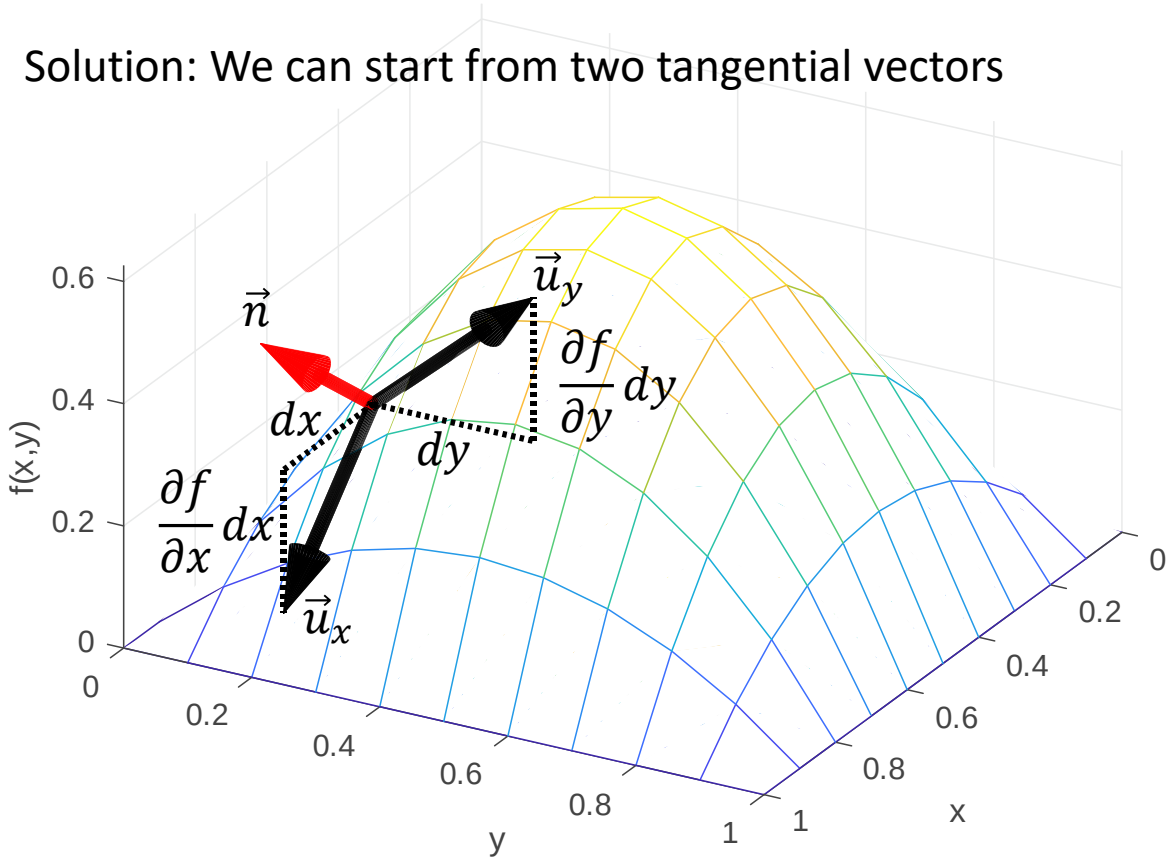
$\nabla f(x,y,z) = y\vec{e}_x + x\vec{e}_y + 2z\vec{e}_z = [y, x, 2z]$.



Partial derivatives – normal vector of a surface

Problem: Find the unit normal vector to the surface $f(x,y) = 1 - x^2 - y^2$

Solution: We can start from two tangential vectors



$$\vec{u}_x = \left[dx, 0, \frac{\partial f}{\partial x} dx\right]$$

$$\vec{u}_y = \left[0, dy, \frac{\partial f}{\partial y} dy\right]$$

Then directly from definition of cross product

$$\vec{N} = \vec{u}_x \times \vec{u}_y = \begin{vmatrix} \vec{e}_x & \vec{e}_y & \vec{e}_z \\ dx & 0 & \frac{\partial f}{\partial x} dx \\ 0 & dy & \frac{\partial f}{\partial y} dy \end{vmatrix} =$$

$$= \left[-\frac{\partial f}{\partial x}, -\frac{\partial f}{\partial y}, 1\right] dxdy$$

$$\vec{n} = \frac{\vec{N}}{\|\vec{N}\|}$$

Partial derivatives – normal vector of a surface

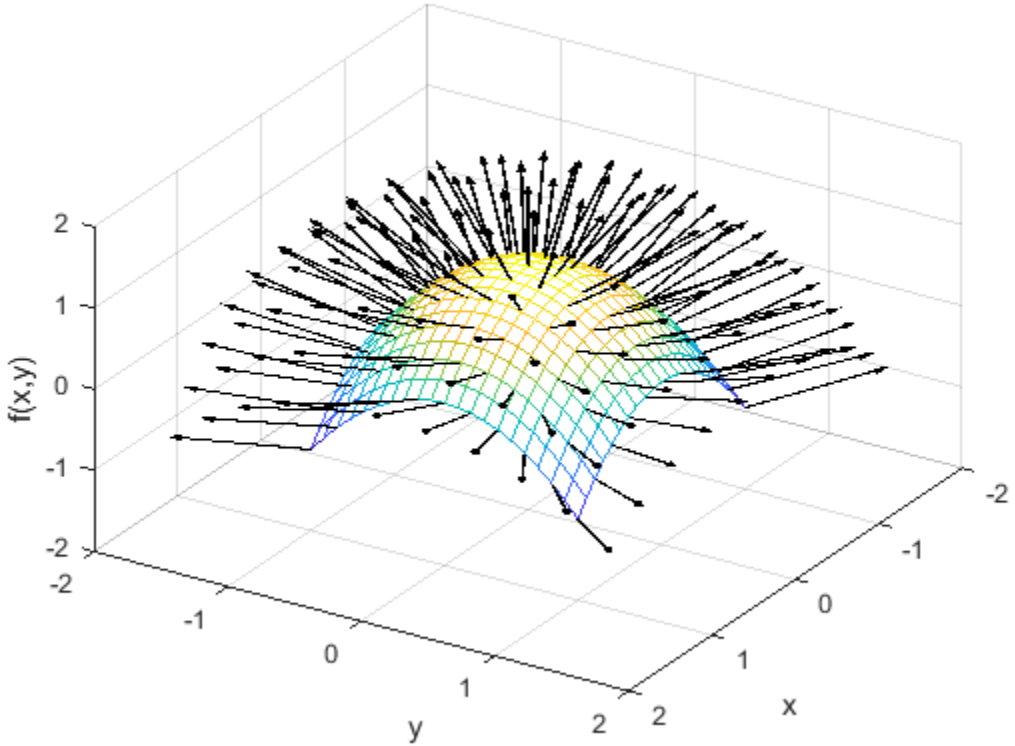
Problem: Find the unit normal vector to the surface $f(x, y) = 1 - x^2 - y^2$

Solution: Using the derived formula for function $f(x, y) = 1 - x^2 - y^2$, we get

$$\vec{N} = \left[-\frac{\partial f}{\partial x}, -\frac{\partial f}{\partial x}, 1\right] dxdy =$$

$$= [2x, 2y, 1]dxdy$$

$$\vec{n} = \frac{\vec{N}}{\|\vec{N}\|} = \frac{[2x, 2y, 1]}{\sqrt{(2x)^2 + (2y)^2 + 1}}$$



Partial derivatives – normal vector of a surface

Problem: The ball bounces off (perfectly elastic collision) the following surface $S: \{z = 1 - x^2 - y^2\}$ Find the velocity vector after the collision if its initial position and velocity are equal $\vec{r}_0 = [1,2,3]$ and $\vec{v}_0 = [0,0,-1]$, respectively. The ball dimension can be neglected.

Solution: The angle of incidence is equal to the angle of reflection. Since we have already derived the relation for the velocity after the impact

$$\vec{v}_2 = \vec{v}_1 - 2(\vec{v}_1 \circ \vec{n})\vec{n}$$

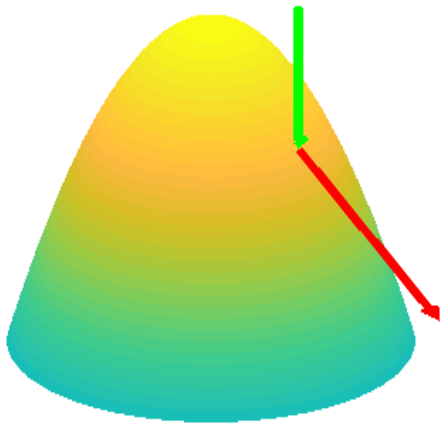
the problem boils down to determining the normal vector $\vec{n}$ at the point of contact. To find this point we have to analyze the motion of the ball before the incident

$$\vec{r}(t) = \vec{r}_0 + \vec{v}_0 t = [1,2,3 - t] = [x(t), y(t), z(t)]$$

and compare it with the surface equation

$$z(t) = 1 - x^2(t) - y^2(t) \Rightarrow 3 - t = 1 - 1 - 4$$

The ball reaches the surface after $t = 7$ s in point $P = (1,2,-4)$.



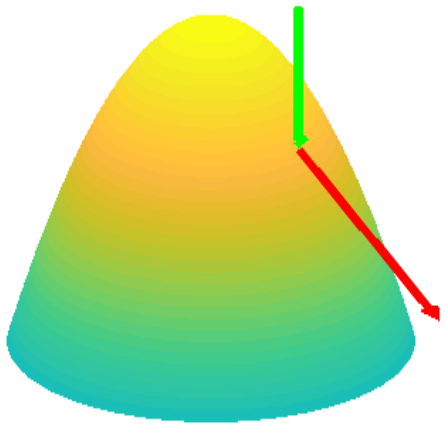
Partial derivatives – normal vector of a surface

Problem: The ball bounces off (perfectly elastic collision) the following surface $S: \{z = 1 - x^2 - y^2\}$ Find the velocity vector after the collision if its initial position and velocity are equal $\vec{r}_0 = [1,2,3]$ and $\vec{v}_0 = [0,0,-1]$, respectively. The ball dimension can be neglected.

Solution: The normal vector $\vec{n}$ at the point of contact $P = (1,2,-4)$ can be found from the formula $\vec{n} = \frac{\vec{N}}{\|\vec{N}\|}$ where $\vec{N} = \left[-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, 1\right] = [2x, 2y, 1] = [2,4,1]$. So, $\vec{n} = \frac{[2,4,1]}{\sqrt{21}}$

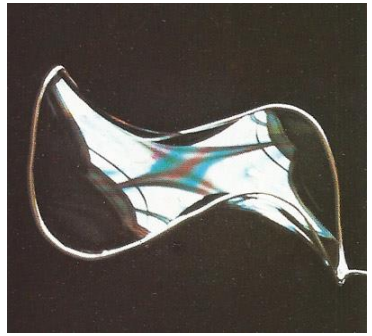
Then using relation $\vec{v}_2 = \vec{v}_1 - 2(\vec{v}_1 \circ \vec{n})\vec{n}$ where $\vec{v}_1 = \vec{v}_0$ we get

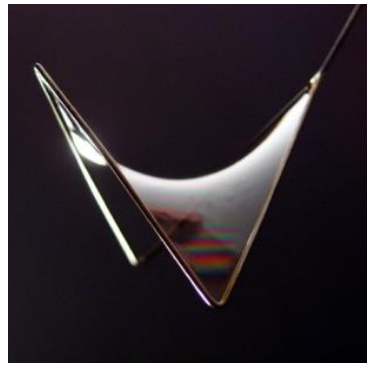
$$\vec{v}_2 = [0,0,-1] - 2\left(-\frac{1}{\sqrt{21}}\right)[2,4,1] = \left[\frac{4}{21}, \frac{8}{21}, -\frac{19}{21}\right]$$



Partial differential equation – introduction

Problem: Let us consider a soap film like in the pictures. Due to the surface tension, the geometry of the soap film is the minimal-area surface spanning the given loop. The equation governing such process (in the first approximation) is $\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$. It is named after Pierre-Simon Laplace. Find a shape of the film for a fixed boundary.





Solution: Let us assume the computational domain as a rectangle

$$\Omega: x \in [0,2], y \in [0,1]$$

with the following boundary conditions

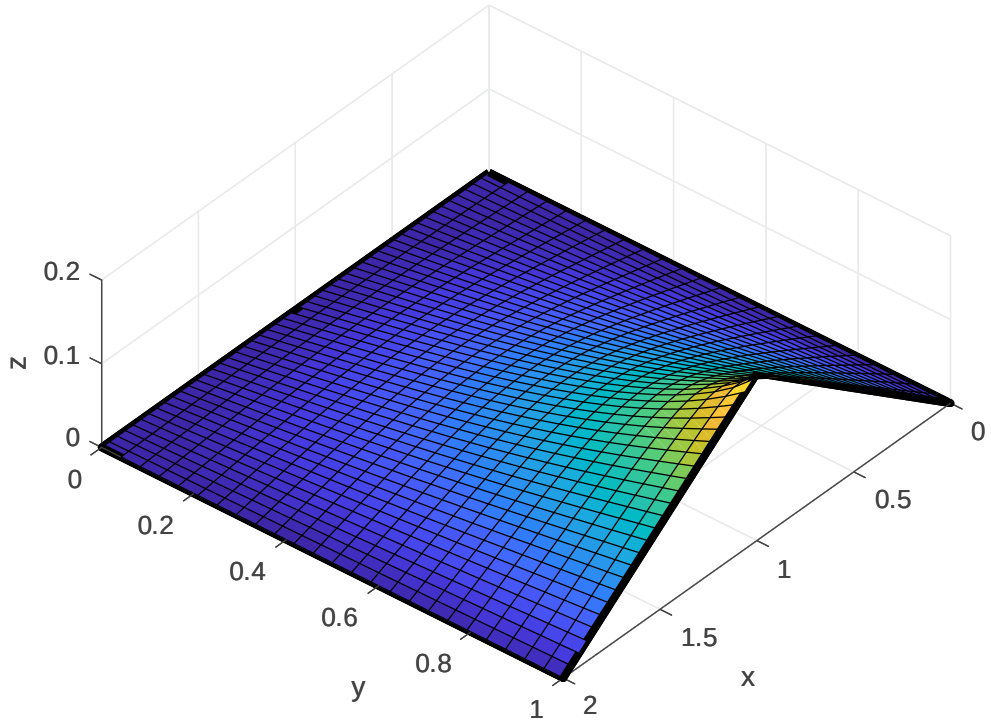
$$f(0,y) = f(2,y) = f(x,0) = 0,$$
$$f(x,1) = g(x) = 1 - |x-1|$$

Using the separation of variables method or https://www.wolframalpha.com we get

$$f(x,y) = \sum_{n=1}^{\infty} \mathrm{A_n} \sin\left(\frac{n\pi x}{2}\right) \sinh\left(\frac{n\pi y}{2}\right)$$

where

$$\mathrm{A_n} = \frac{2}{2\sinh\left(\frac{n\pi}{2}\right)} \int_0^2 g(x) \sin\left(\frac{n\pi x}{2}\right) dx$$



Partial differential equation – introduction

Problem: Let us consider a string of length $L = 10\ m$ that has been tightly stretched between two points at $x = 0$ and $x = L$. The equation governing such process (for small displacement) is called wave equation $\frac{\partial^2 f}{\partial x^2} = \frac{1}{v^2}\frac{\partial^2 f}{\partial t^2}$, where $v = 1000\ m/s$ represents the speed of propagation of the wave in the string (which depends on the mass of the string and its tension).

Solution: The vibration of the string depends on the initial conditions. Let $s(x)$ represents the shape of the string in $t = 0$ and $v(x)$ represents its initial speed . As in the previous example, the computational domain is a rectangle $\Omega: x \in [0, L], t \in [0, \infty)$. And once again we need four conditions to find unambiguous solution

$$f(0,t) = f(L,t) = 0, \qquad f(x,0) = s(x), \qquad \frac{\partial f}{\partial t}(x,0) = v(x)$$

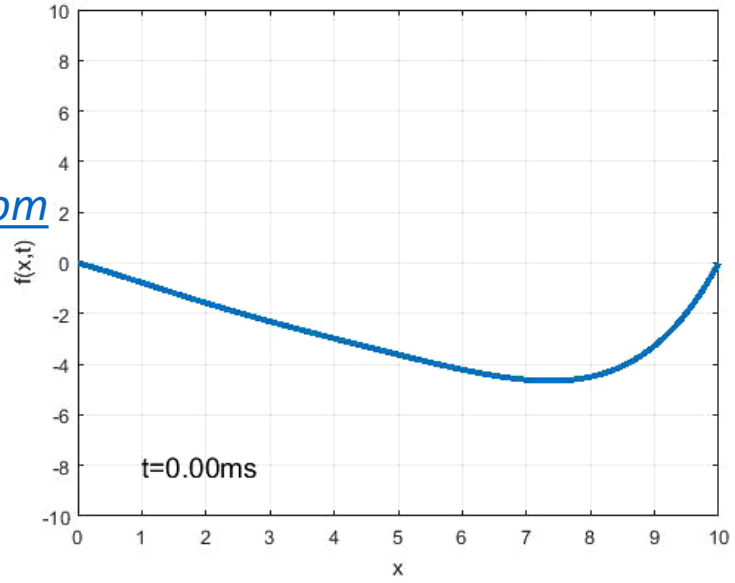
Let us take $s(x) = 0.0006x^5 - 0.01x^4 + 0.06x^3 - 0.13x^2 - 0.7x$ and $v(x) = 0$.

Using the separation of variables method or https://www.wolframalpha.com

$$f(x,t) = \sum_{n=1}^{\infty} \mathrm{A_n} \sin\left(\frac{n\pi x}{L}\right) \cos\left(\frac{n\pi v t}{L}\right)$$

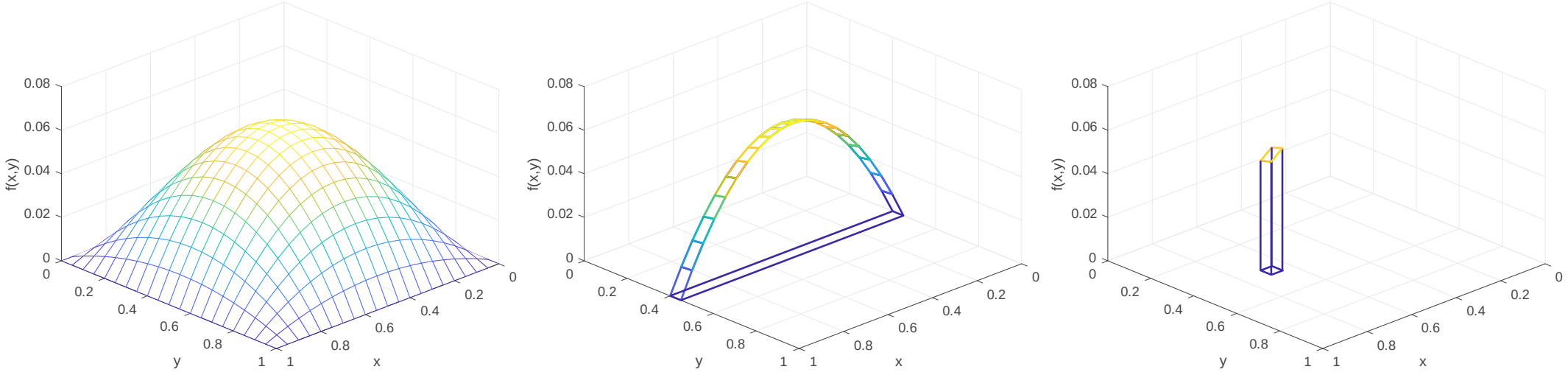
where

$$\mathrm{A_n} = \frac{2}{L}\int_0^2 s(x) \sin\left(\frac{n\pi x}{2}\right)\, dx$$



Multivariable functions – integration

Double integral as a volume



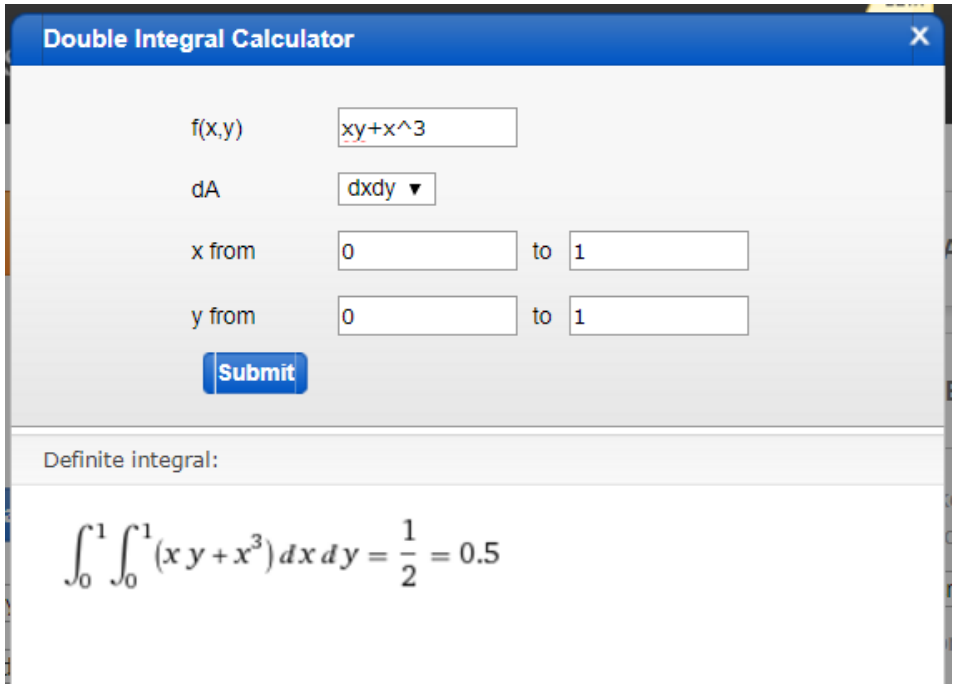
$$\iint_S f(x,y)dxdy = \int_0^1 \left[\int_0^1 f(x,y)\,dx\right] dy = \int_0^1 P(y)\,dy = V$$

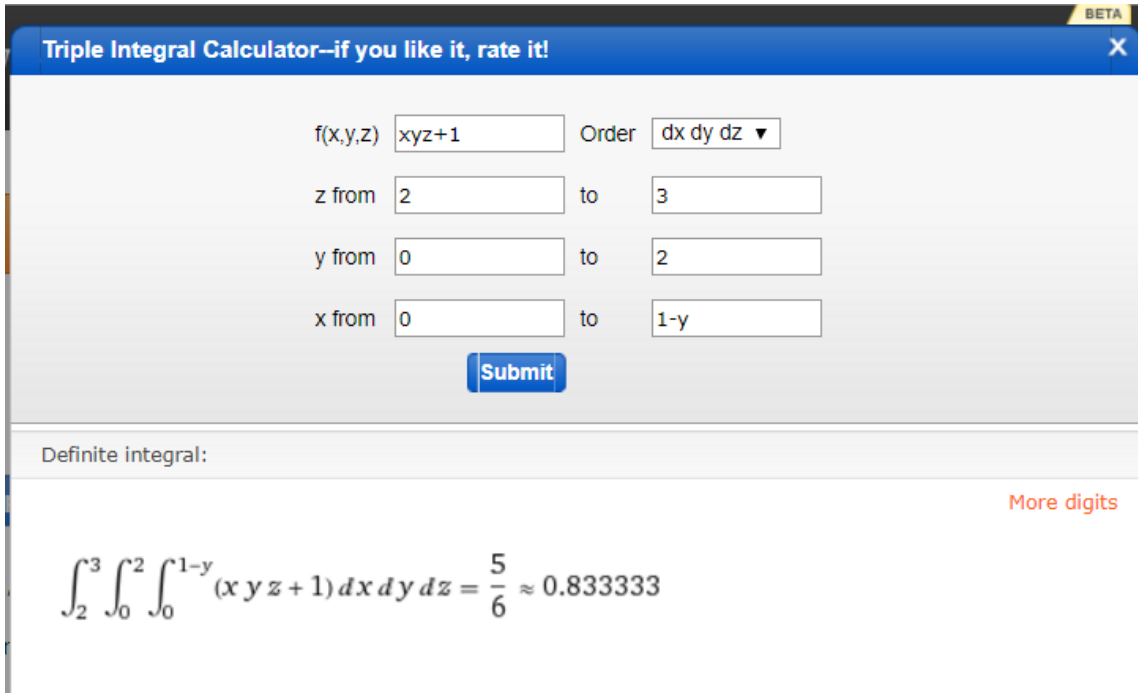
or

$$V = \iint_S f(x,y)dxdy = \iint_S f(x,y)ds$$

Multivariable functions – integration

Double/triple Integral Calculator http://www.wolframalpha.com/widgets/view.jsp?id=f5f3cbf14f4f5d6d2085bf2d0fb76e8a





Double integral – example

Problem: The surface electric charge density on a thin rectangular plate $S: \{x \in [0,1], y \in [0,1]\}$, is expressed by

$$f(x,y) = x^2 - xy - y^2 \; C/m^2.$$

What is the total charge of the plate?

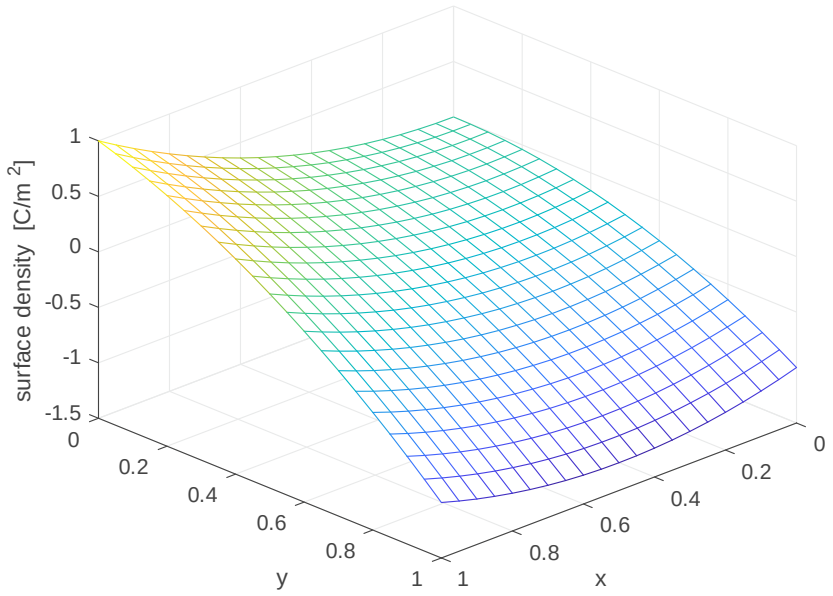
Solution: If the charge distribution is constant the total charge is just a product of this constant charge density and the area of the plate. Otherwise, the plate must be divided into infinitesimal parts of area ds. At each part the charge can be expressed by

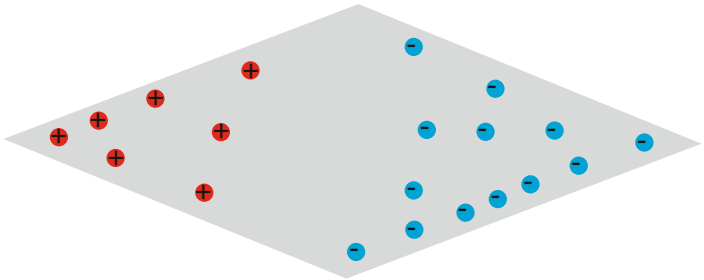
$$dQ = f(x,y)ds$$

Hence, the total charge is the sum of all these charges

$$Q = \iint_S f(x,y)ds =$$

$$= \iint_S (x^2 - xy - y^2)dxdy = \frac{5}{12} \; C$$

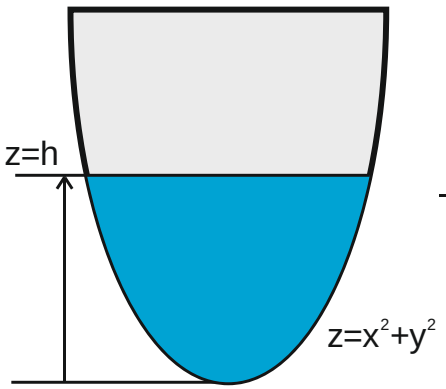


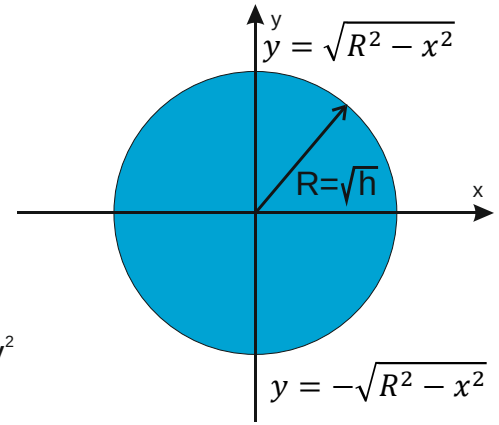


surface electric charge density
= charge/ infinitesimal area

Double integral – example

Problem: The surface $z = x^2 + y^2$ and plane $z = 10$ create the tank presented in the figure. Find the relation between the level of liquid $h \in (0,10)$ in the tank and the volume of this liquid.





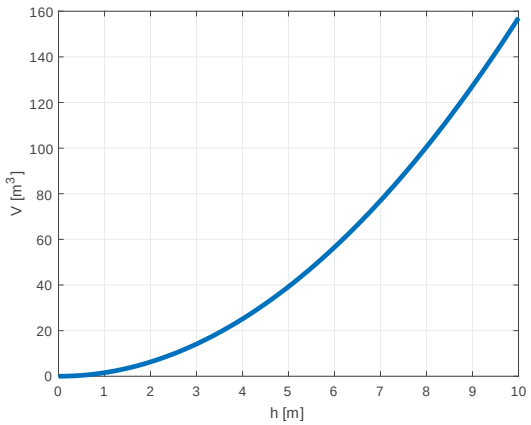
Solution: The volume can be evaluated with the use of double integral

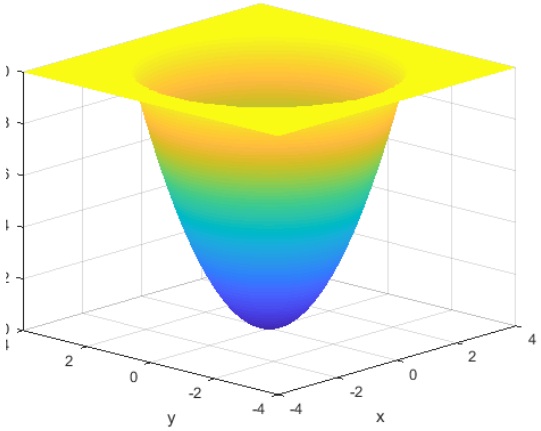
$$V(h) = \iint_S f(x,y)dxdy = \iint_S (h - x^2 - y^2)dxdy$$

where $S: \{x \in [-R, R], y \in [-\sqrt{R^2 - x^2}, \sqrt{R^2 - x^2}]\}$.

Hence,

$$V(h) = \int_{-R}^{R} \left[\int_{-\sqrt{R^2-x^2}}^{\sqrt{R^2-x^2}} (h - x^2 - y^2)dy \right] dx = \frac{\pi h^2}{2}$$





Triple integral – example

Problem: The volume electric charge density on a cuboid $V: \{x \in [0,1], y \in [0,1], z \in [0,1]\}$, is expressed by

$$f(x,y,z) = 2 - 64(x^2 - x)(y^2 - y)(z^2 - z)\ C/m^3.$$

What is the total charge of the cuboid?

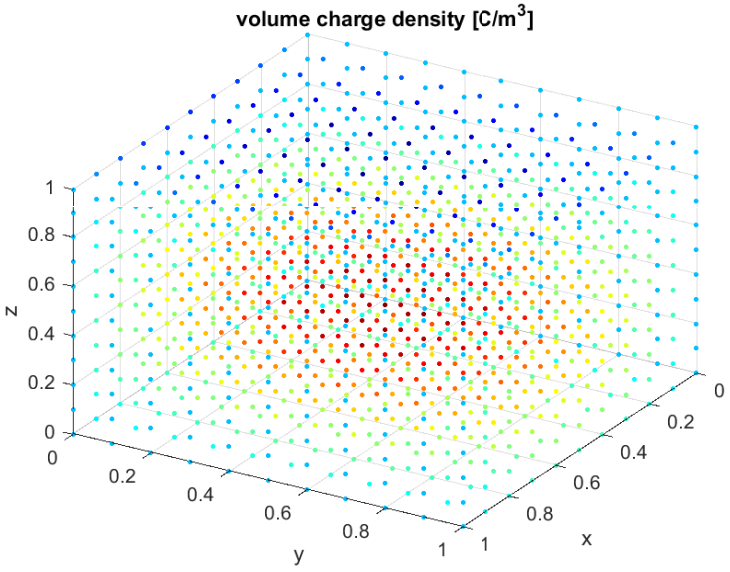
Solution: If the charge distribution is constant the total charge is just a product of this constant charge density and the volume of the cuboid. Otherwise, the cuboid must be divided into infinitesimal parts of volume dv. At each part the charge can be expressed by

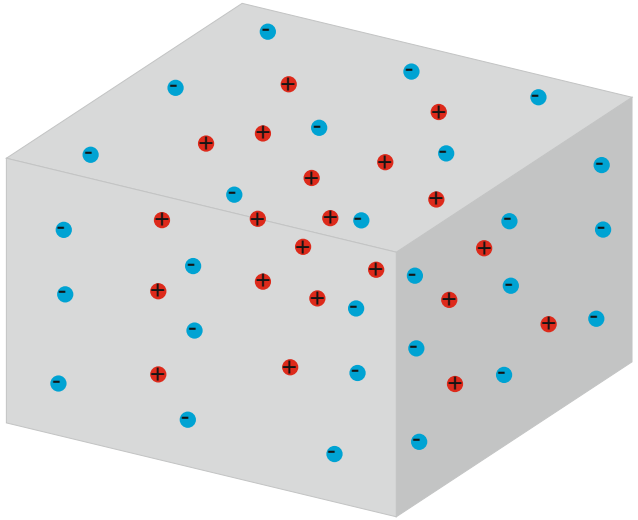
$$dQ = f(x,y,z)dv$$

Hence, the total charge is the sum of all these charges

$$Q = \iiint_V f(x,y,z)dv =$$

$$= \iint_S [2 - 64(x^2 - x)(y^2 - y)(z^2 - z)]dxdydz = \frac{62}{27}\ C$$





volume electric charge density
= charge/ infinitesimal volume

Extracurricular materials

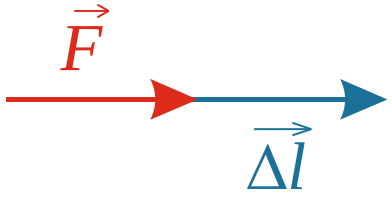
(beyond the scope of the tests)

Line integral of a vector field

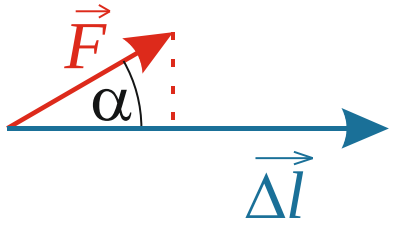
Problem: Let us assume that the vector function $\vec{F}(x, y, z) = [x + y, z, y]$ represents a force acting on the object moving along the curve defined by parametric equations $L: \{\, x(t) = 2t + 1\ ;\ y(t) = 3 - 2t\ ;\ z(t) = t^2\ ;\ t \in [0,1]\,\}$. Find the work corresponding to such a movement.

Solution: The work is defined as a scalar product between the force vector and the displacement vector

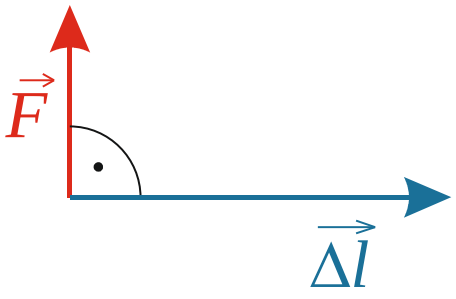
$$W = \vec{F} \circ \overrightarrow{\Delta l}$$



$$W = \|\vec{F}\|\|\overrightarrow{\Delta l}\|$$



$$W = \|\vec{F}\|\|\overrightarrow{\Delta l}\|\cos(\alpha)$$



$$W = 0$$

Line integral of a vector field

Problem: Let us assume that the vector function $\vec{F}(x,y,z) = [x+y, z, y]$ represents a force acting on the object moving along the curve defined by parametric equations $L: \{\, x(t) = 2t+1\,;\; y(t) = 3 - t\,;\; z(t) = t^2\,;\; t \in [0,1]\,\}$. Find the work corresponding to such a movement.

Solution: If the force changes during the movement the curve must be divided into infinitesimal segments $\overrightarrow{dl}$ an the infinitesimal works $dW = \vec{F} \circ \overrightarrow{dl}$ must be added

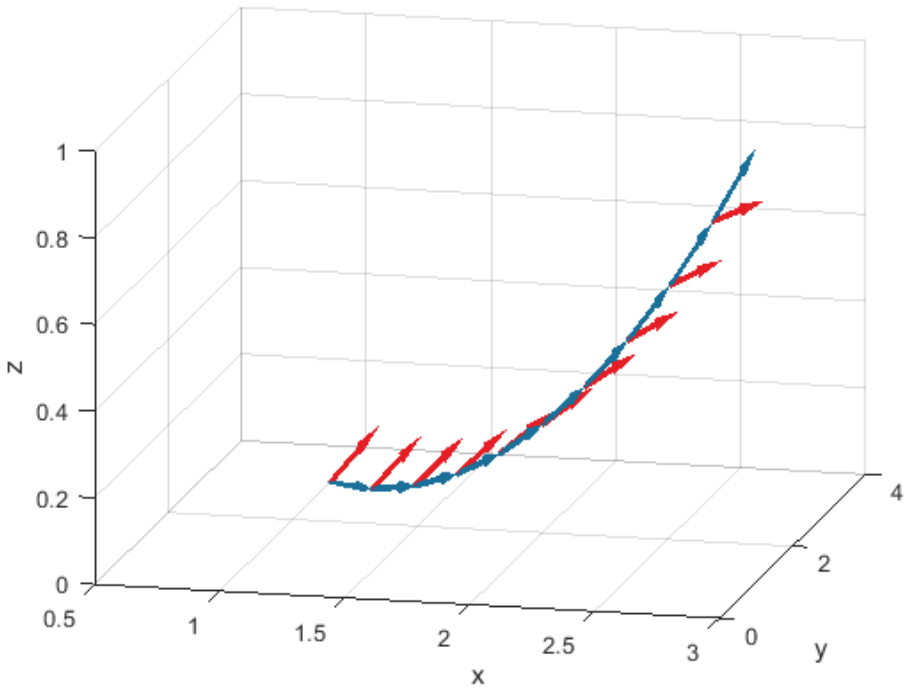
$$W = \int_L dW = \int_L \vec{F} \circ \overrightarrow{dl}$$

Since $\overrightarrow{dl} = [dx, dy, dz] = \left[\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt}\right] dt$, the line integral becomes regular integral with respect to t

$$W = \int_L [x+y, z, y] \circ \left[\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt}\right] dt$$

$$= \int_0^1 [t+4, t^2, 3-t] \circ [2, -1, 2t\,] dt =$$

$$= \int_0^1 [8 + 8t - 3t^2]\, dt = 11\, J$$

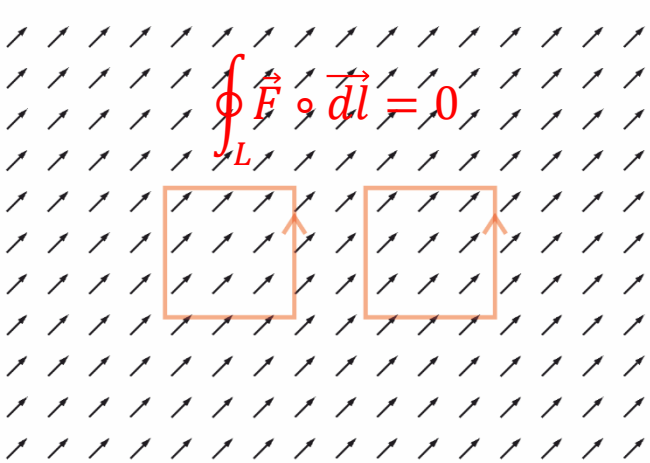


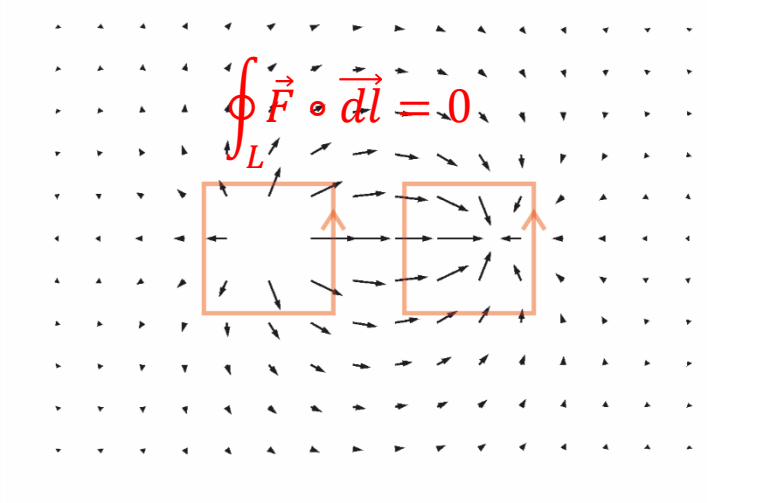
Line integral of a vector field

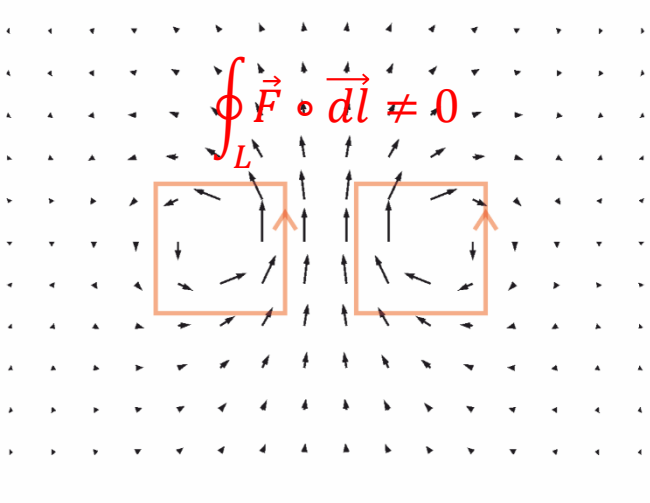
If the vector field $\vec{F}$ is irrotational (conservative), then

$$\oint_L \vec{F} \circ \overrightarrow{dl} = 0$$

for any closed curve L.





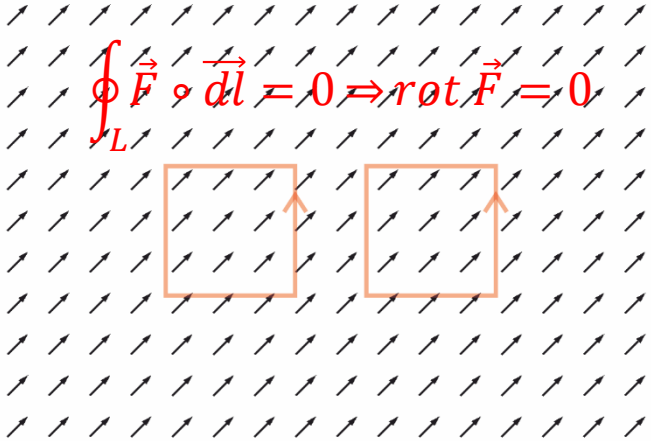


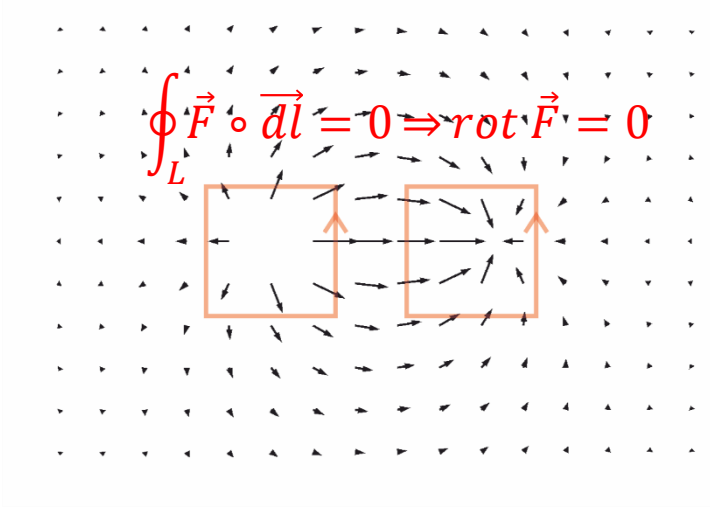
Differential operators – curl (rot)

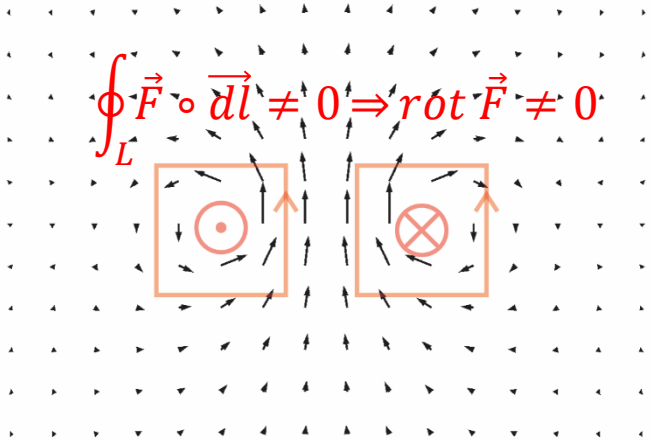
The curl/rot of vector field $\vec{F}$ represents the rotation of this field in each point (according to the right-hand rule)

$$\mathrm{rot}\,\vec{F} \circ \vec{n} = \lim_{S \to 0} \frac{\oint_L \vec{F} \circ \overrightarrow{dl}}{S}$$

where S is the area determined by curve L and $\vec{n}$ is a unit vector normal to this curve.



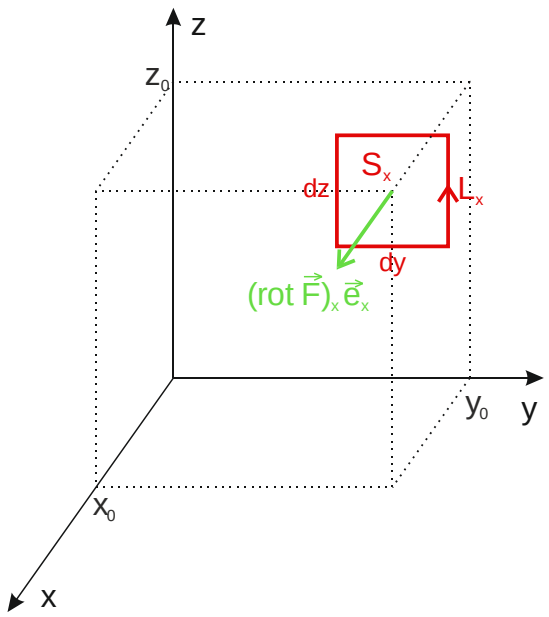




Differential operators – curl (rot)

$$\operatorname{rot}\vec{F} = \lim_{S_x\to 0}\frac{\oint_{L_x}\vec{F}\circ\overrightarrow{dl}}{S_x}\,\vec{e}_x + \lim_{S_y\to 0}\frac{\oint_{L_y}\vec{F}\circ\overrightarrow{dl}}{S_y}\,\vec{e}_y + \lim_{S_z\to 0}\frac{\oint_{L_z}\vec{F}\circ\overrightarrow{dl}}{S_z}\,\vec{e}_z$$

$$\lim_{S_x\to 0}\frac{\oint_{L_x}\vec{F}\circ\overrightarrow{dl}}{S_x} = \frac{\vec{F}\left(x_0, y_0+\frac{dy}{2}, z_0\right)\circ(dz\vec{e}_z) + \vec{F}\left(x_0, y_0, z_0+\frac{dz}{2}\right)\circ(-dy\vec{e}_y) + \vec{F}\left(x_0, y_0-\frac{dy}{2}, z_0\right)\circ(-dz\vec{e}_z) + \vec{F}\left(x_0, y_0, z_0-\frac{dz}{2}\right)\circ(dy\vec{e}_y)}{dydz}$$



$$= \frac{F_z\left(x_0, y_0+\frac{dy}{2}, z_0\right)dz - F_y\left(x_0, y_0, z_0+\frac{dz}{2}\right)dy - F_z\left(x_0, y_0-\frac{dy}{2}, z_0\right)dz + F_y\left(x_0, y_0, z_0-\frac{dz}{2}\right)dy}{dydz}$$

$$= \frac{F_z\left(x_0, y_0+\frac{dy}{2}, z_0\right) - F_z\left(x_0, y_0-\frac{dy}{2}, z_0\right)}{dy} - \frac{F_y\left(x_0, y_0, z_0+\frac{dz}{2}\right) - F_y\left(x_0, y_0, z_0-\frac{dz}{2}\right)}{dz}$$

$$= \frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}$$

$$\lim_{S_y\to 0}\frac{\oint_{L_y}\vec{F}\circ\overrightarrow{dl}}{S_y} = \frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}$$

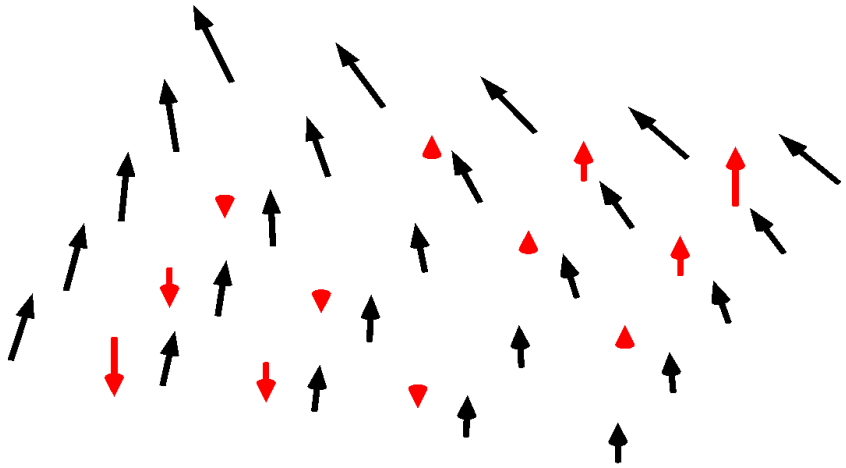
$$\lim_{S_z\to 0}\frac{\oint_{L_z}\vec{F}\circ\overrightarrow{dl}}{S_z} = \frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}$$

Differential operators – curl (rot)

$$\operatorname{rot}\vec{F} = \begin{vmatrix} \vec{e}_x & \vec{e}_y & \vec{e}_z \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ F_x & F_y & F_z \end{vmatrix} = \left[\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right]\vec{e}_x - \left[\frac{\partial F_z}{\partial x} - \frac{\partial F_x}{\partial z}\right]\vec{e}_y + \left[\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right]\vec{e}_z = \nabla \times \vec{F}$$

Example: Find the curl of $\vec{F}(x, y, z) = [y^2, x^2, z]$

$$\nabla \times \vec{F} = \begin{vmatrix} \vec{e}_x & \vec{e}_y & \vec{e}_z \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ y^2 & x^2 & z \end{vmatrix} = [0,0,2x - 2y]$$



Line integral of a scalar field

Problem: Let us assume that the function $f(x,y,z) = xy + z - z^2 - 3\ kg/m$ represents a linear mass distribution along the curve $L : \{\ x(t) = 2t + 1\ ;\ y(t) = 3 - t\ ;\ z(t) = t^2\ ;\ t \in [0,1]\ \}$. Find the total mass of the object.

Source: Lord of War (2005)

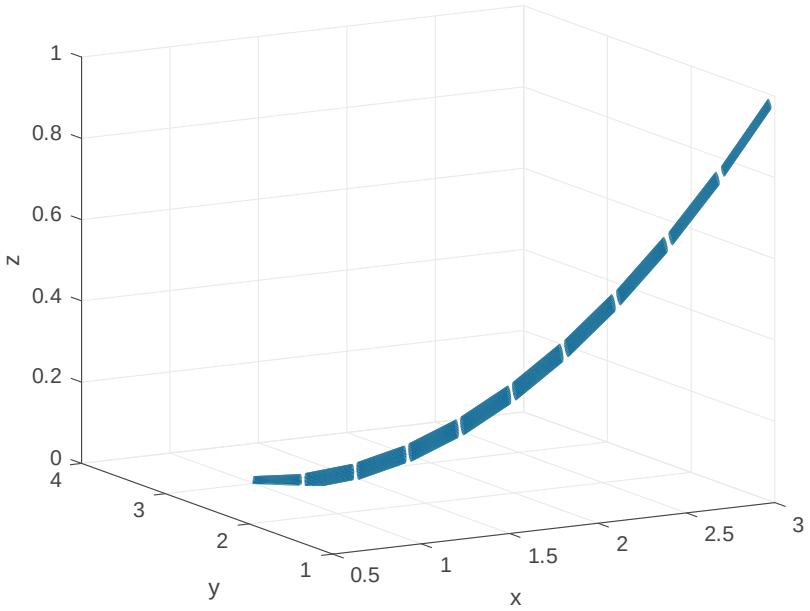
Solution: If the linear mass distribution is constant along the curve the total mass is just a product of this distribution and its length. Otherwise the curve must be divided into infinitesimal segments dl and the infinitesimal masses $dm = f(x,y,z)dl$ must be added

$$m = \int_L dm = \int_L f(x,y,z)dl$$

Since $dl = \|\vec{dl}\| = \sqrt{dx^2 + dy^2 + dz^2} = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2}\ dt,$ the line integral becomes regular integral with respect to t

$$m = \int_L (xy + z - z^2 - 3)\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2}\ dt$$

$$= \int_0^1 (5t - t^2 - t^4)\sqrt{5 + 4t^2}dt \approx 5.13\ kg$$

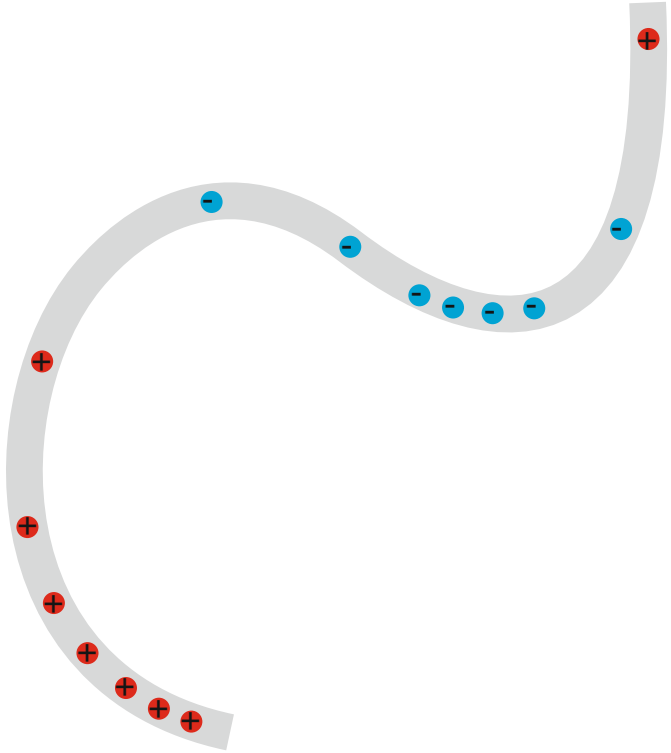


Line integral of a scalar field

Problem: Let us assume that the function $f(x,y,z) = xy + z - z^2 - 3\ C/m$ represents a linear charge density on a fiber described by $L : \{\, x(t) = 2t + 1\,;\ y(t) = 3 - t\,;\ z(t) = t^2\,;\ t \in [0,1]\,\}$. Find the total charge.

Solution: The problem is exactly the same as the previous one. The curve must be divided into infinitesimal segments dl and the infinitesimal charge on each part is $dQ = f(x,y,z)dl$. Then

$$Q = \int_L dQ = \int_L f(x,y,z)dl$$



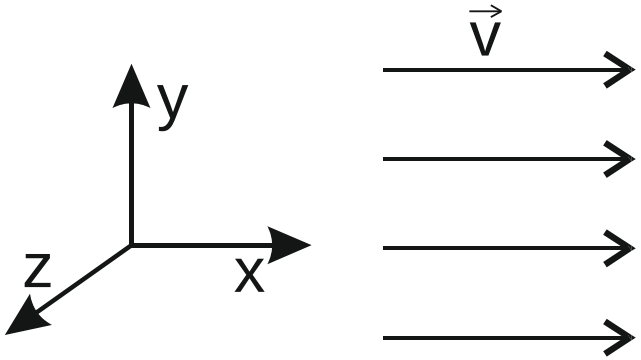
Once again $dl = \sqrt{dx^2 + dy^2 + dz^2} = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2}\, dt$, the line integral becomes regular integral with respect to t

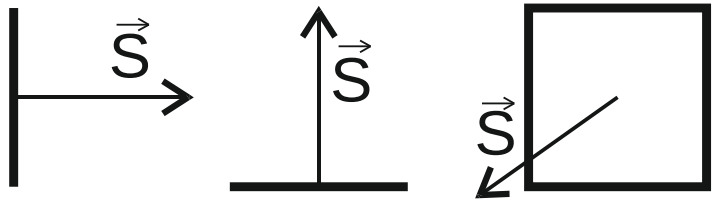
$$Q = \int_L (xy + z - z^2 - 3)\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2}\, dt$$

$$= \int_0^1 (5t - t^2 - t^4)\sqrt{5 + 4t^2}dt \approx 5.13\ C$$

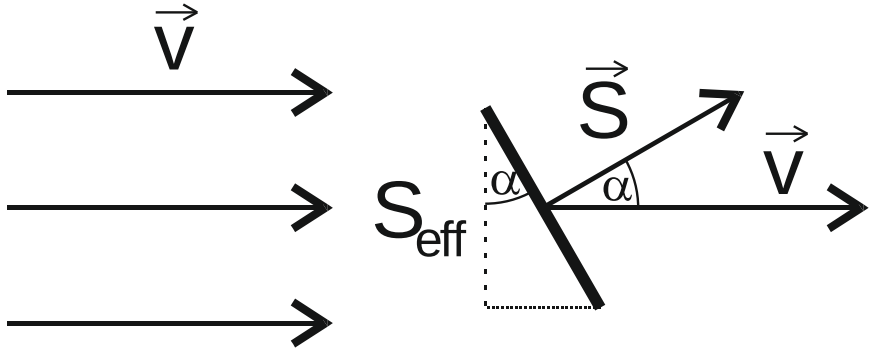
Flux as a scalar product

Flux as a scalar product of vector field and vector area





$\Phi = Sv$ $\Phi = 0$ $\Phi = 0$



$$S = a^2$$
$$S_{eff} = a^2 \cos\alpha$$
$$\Phi = vS\cos\alpha = \vec{v} \circ \vec{S}$$

At 439 feet in length, the five-masted, 42-sail **Royal Clipper** is the largest full-rigged sailing ship in the world

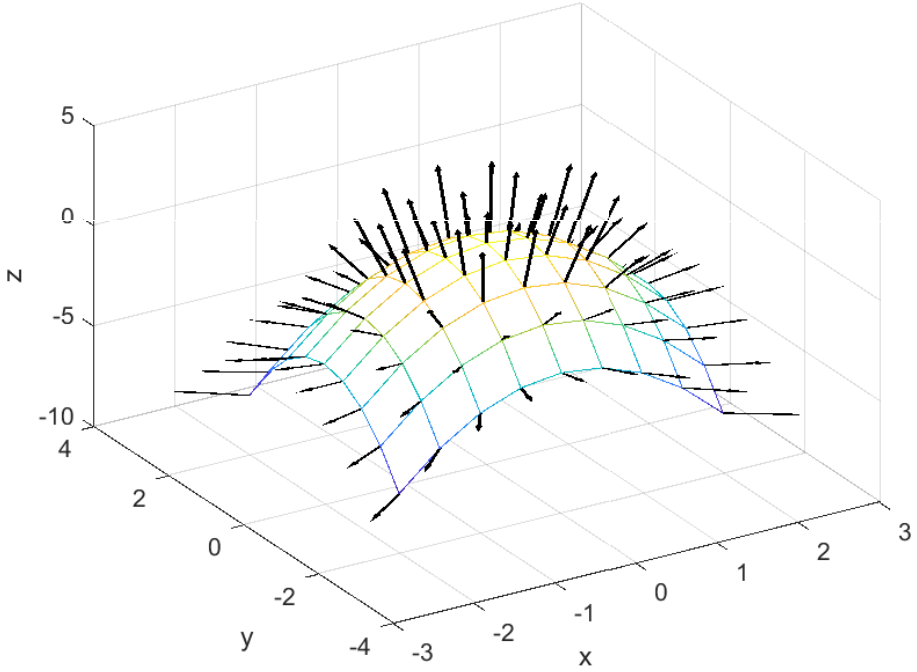
Surface integral of a vector field

Problem: Let us assume that the vector function $\vec{v}(x,y,z) = [-x, -y, \frac{1}{z+1}]$ represents a speed of water and $S: \{z = x^2 + y^2\ ; x \in [-2,2]; y \in [-2,2]\}$ represents a piece of net. Estimate the flux of water flowing through this net.

Solution: If the vector field changes and the surface is curved, the flux is expressed by the sum of infinitesimal fluxes $d\Phi = \vec{v} \circ \overrightarrow{ds}$

$$\Phi = \iint_S d\Phi = \iint_S \vec{v} \circ \overrightarrow{ds}$$

The problem is how to determine the $\overrightarrow{ds}$.



Surface integral of a vector field

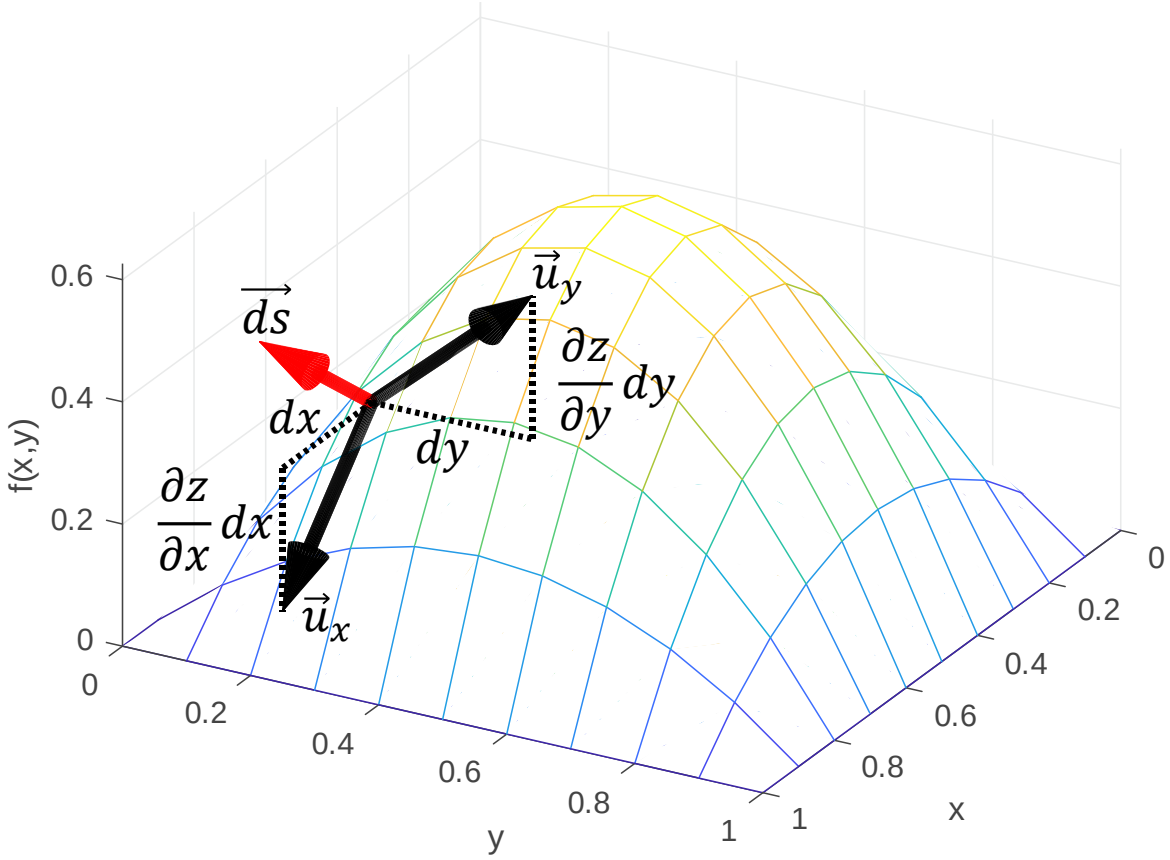
Problem: Let us assume that the vector function $\vec{v}(x,y,z) = [-x, -y, \frac{1}{z+1}]$ represents a speed of water and S: $\{z = x^2 + y^2\ ; x \in [-2,2]; y \in [-2,2]\}$ represents a piece of net. Estimate the flux of water flowing through this net.

Solution: To determine the $\overrightarrow{ds}$ we have to go back to the tangential vectors

$$\vec{u}_x = \left[1,0,\frac{\partial z}{\partial x}\right] dx, \qquad \vec{u}_y = \left[0,1,\frac{\partial z}{\partial y}\right] dy$$

The magnitude of a cross product represents the area of a parallelogram and its direction is orthogonal to the surface

$$\overrightarrow{ds} = \vec{u}_x \times \vec{u}_y = \left[-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, 1\right] dxdy$$



Surface integral of a vector field

Problem: Let us assume that the vector function $\vec{v}(x,y,z) = [-x, -y, \frac{1}{z+1}]$ represents a speed of water and $S: \{z = x^2 + y^2\ ; x \in [-2,2]; y \in [-2,2]\}$ represents a piece of net. Estimate the flux of water flowing through this net.

Solution: So, the flux can be expressed by simple double integral

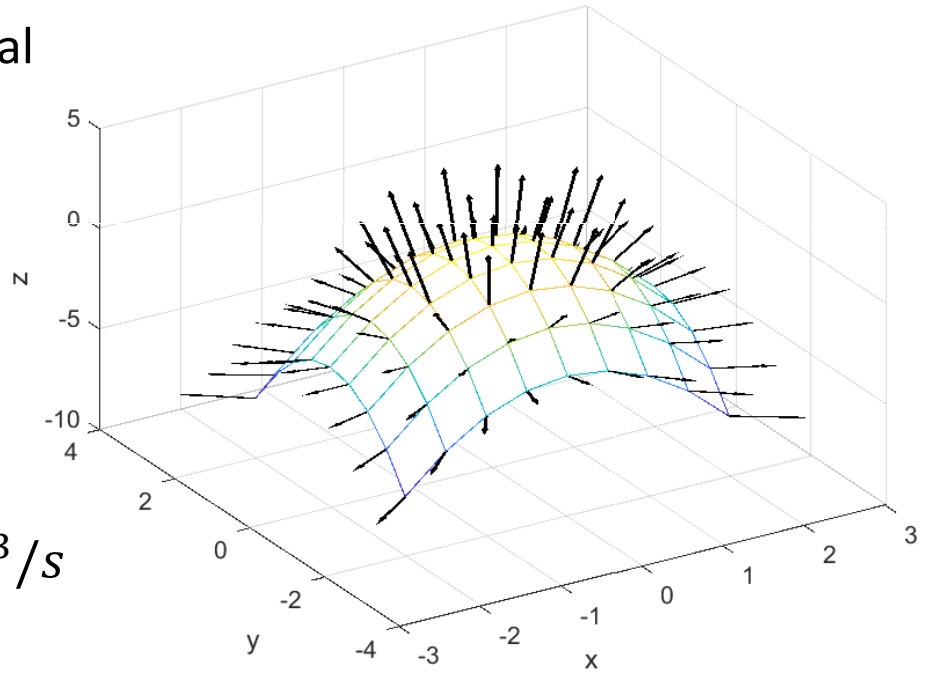
$$\Phi = \iint_S \vec{v} \circ \overrightarrow{ds} = \iint_S \vec{v} \circ \left[-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, 1\right] dxdy$$

Then

$$\Phi = \iint_S [-x, -y, \frac{1}{x^2 + y^2 + 1}] \circ [-2x, -2y, 1] dxdy$$

and finally

$$\Phi = \int_{-2}^{2} \int_{-2}^{2} \left[2x^2 + 2y^2 + \frac{1}{x^2 + y^2 + 1}\right] dxdy \approx 48.3\ m^3/s$$

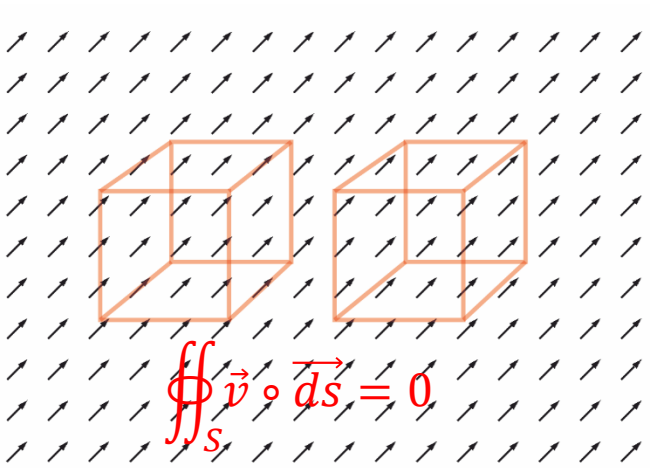


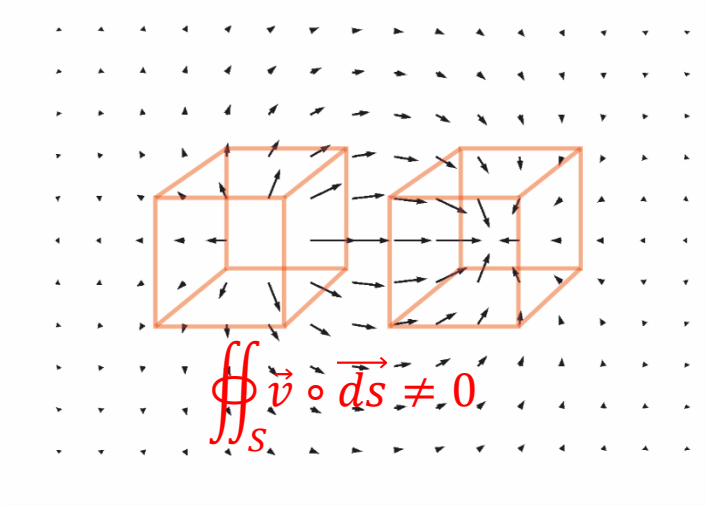
Surface integral of a vector field

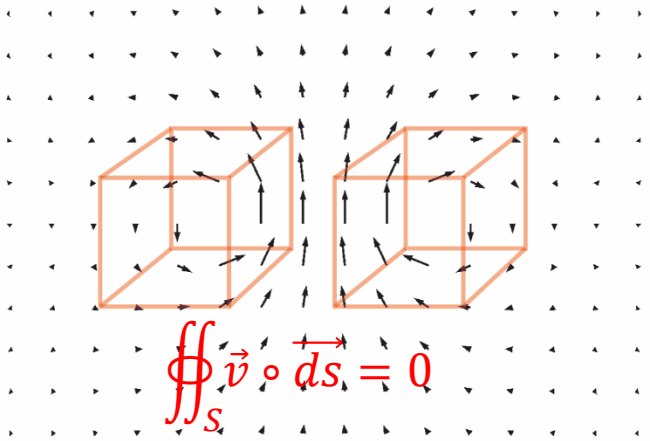
If the vector field $\vec{v}$ is solenoidal (free of sources), then

$$\oiint_S \vec{v} \circ \overrightarrow{ds} = 0$$

for any closed surface S.





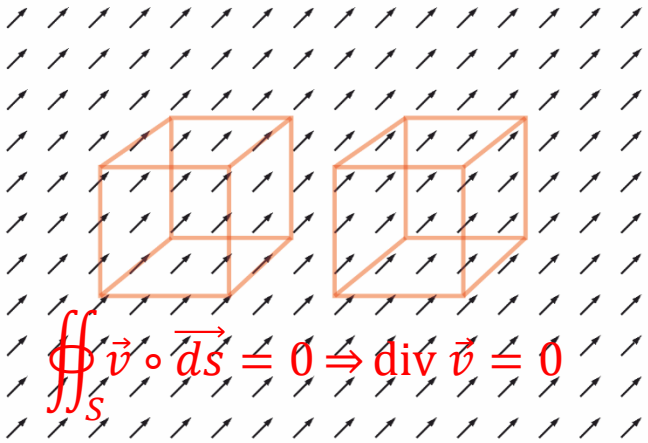


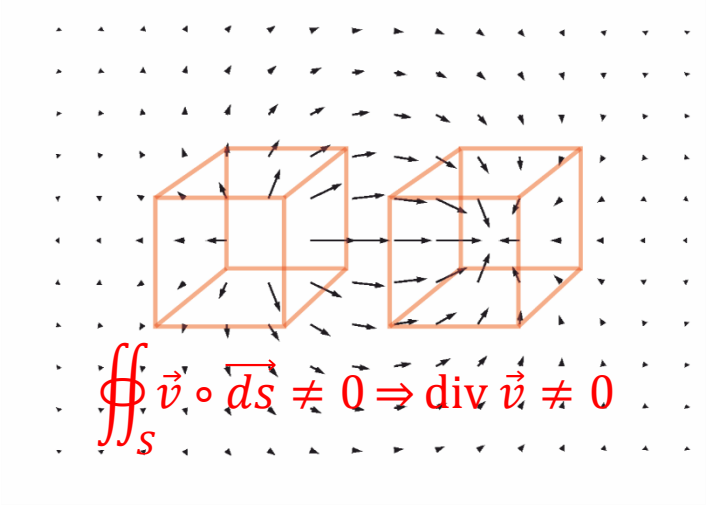
Differential operators – divergence

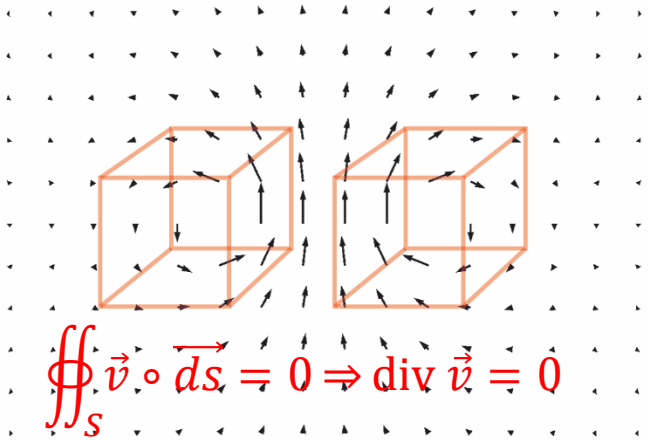
The divergence of vector field $\vec{v}$ produces a scalar field giving the quantity of the vector field's source at each point

$$\operatorname{div} \vec{v} = \lim_{V \to 0} \frac{\oiint_S \vec{v} \circ \overrightarrow{dl}}{V}$$

where V is the volume determined by surface S.







Differential operators – divergence

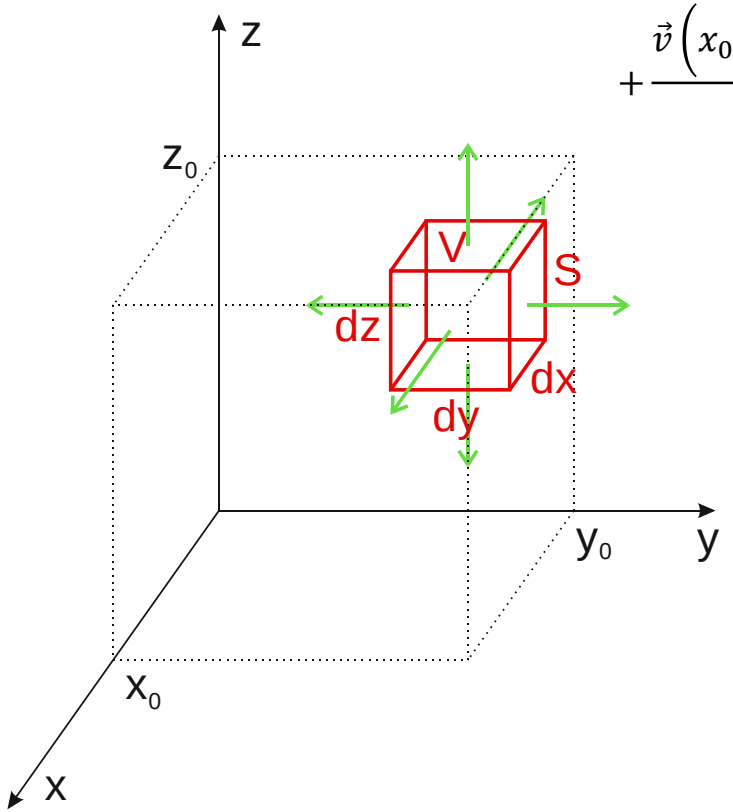
$$\operatorname{div}\vec{v} = \lim_{V\to 0}\frac{\oiint_S \vec{v}\circ\overrightarrow{ds}}{V} = \frac{\vec{v}\left(x_0+\frac{dx}{2},y_0,z_0\right)\circ(dydz\vec{e}_x)+\vec{v}\left(x_0-\frac{dx}{2},y_0,z_0\right)\circ(-dydz\vec{e}_x)+\vec{v}\left(x_0,y_0+\frac{dy}{2},z_0\right)\circ(dxdz\vec{e}_y)}{dxdydz}+$$

$$+\frac{\vec{v}\left(x_0,y_0-\frac{dy}{2},z_0\right)\circ(-dxdz\vec{e}_y)+\vec{v}\left(x_0,y_0,z_0+\frac{dz}{2}\right)\circ(dxdy\vec{e}_z)+\vec{v}\left(x_0,y_0,z_0-\frac{dz}{2}\right)\circ(-dxdy\vec{e}_z)}{dxdydz}=$$

$$=\frac{v_x\left(x_0+\frac{dx}{2},y_0,z_0\right)dydz-v_x\left(x_0-\frac{dx}{2},y_0,z_0\right)dydz+v_y\left(x_0,y_0+\frac{dy}{2},z_0\right)dxdz}{dxdydz}+$$

$$+\frac{-v_y\left(x_0,y_0+\frac{dy}{2},z_0\right)dxdz+v_z\left(x_0,y_0,z_0+\frac{dz}{2}\right)dxdy-v_z\left(x_0,y_0,z_0-\frac{dz}{2}\right)dxdy}{dxdydz}=$$

$$=\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}+\frac{\partial v_z}{\partial z}=\nabla\circ\vec{v}$$

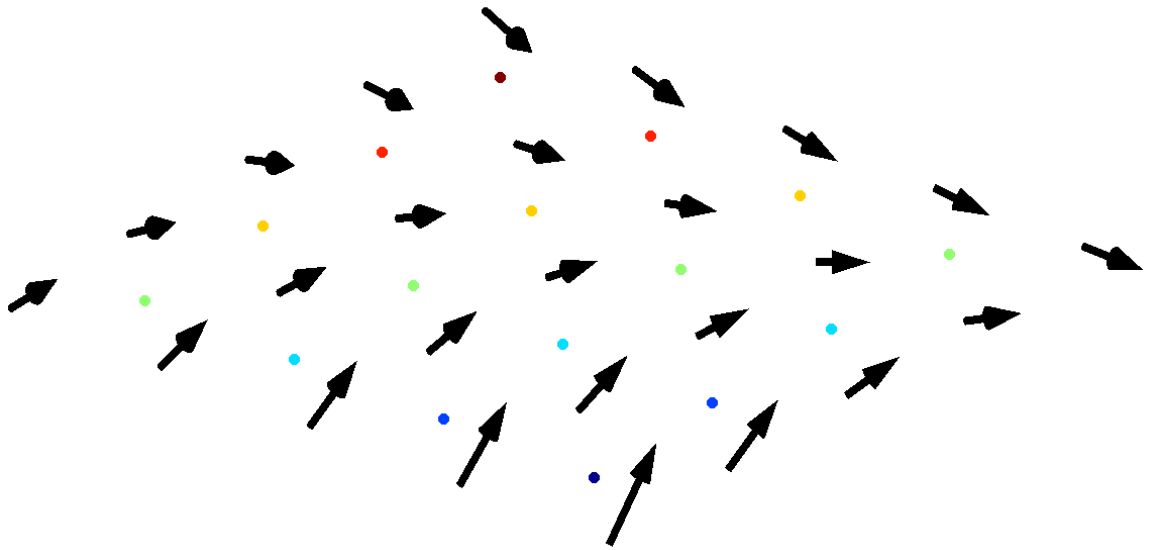


Differential operators – divergence

$$\operatorname{div} \vec{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} = \nabla \circ \vec{v}$$

Example: Find the divergence of $\vec{v}(x, y, z) = [xy, z - xy, x - y]$

$$\nabla \circ \vec{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$$
$$= \frac{\partial}{\partial x}(xy) + \frac{\partial}{\partial y}(z - xy) + \frac{\partial}{\partial z}(x - y) =$$
$$= y - x$$



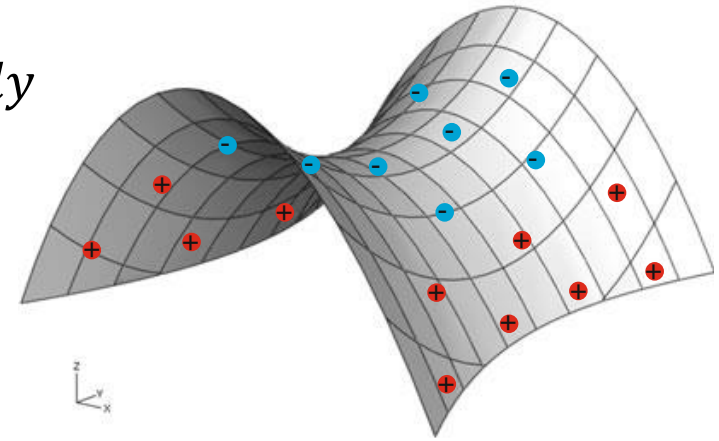
Surface integral of a scalar field

Problem: Let us assume that the scalar function $f(x,y,z) = xy - z\ C/m^2$ represents a surface electric charge density on a thin sheet S: $\{z = x^2 + y^2\ ; x \in [-2,2]; y \in [-2,2]\}$. Estimate the total charge of the sheet.

Solution: The surface must be divided into infinitesimal parts ds and the infinitesimal charges $dQ = f(x,y,z)ds$ must be added

$$\mathrm{Q} = \iint_S dQ = \iint_S f(x,y,z)ds$$

Since $ds = \|\overrightarrow{ds}\| = \left\|\left[-\frac{\partial z}{\partial x}, -\frac{\partial z}{\partial y}, 1\right]\right\| dxdy = \sqrt{\left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2 + 1}\ dxdy$



the total charge can be expressed by simple double integral

$$\mathrm{Q} = \iint_S (xy - x^2 - y^2)\sqrt{4x^2 + 4y^2 + 1}\ dxdy$$

So, finally

$$\mathrm{Q} = \int_{-2}^{2}\int_{-2}^{2} (xy - x^2 - y^2)\sqrt{4x^2 + 4y^2 + 1}\ dxdy = -166.506\ C$$

Modeling and Simulation of Systems

Lecture 4: Methods for the numerical solution of the ordinary differential equation

Piotr Kowalczyk

How Populations Grow: The Exponential and Logistic Equations

Why the simulation of population growth is so important?

It is used in:

- artificial neural networks,
- biology,
- biomathematics,
- chemistry,
- demography,
- economics,
- sociology,
- political science,
- …?

Exponential equation – the simplest model

Let us assume that population growth is proportional to current population size, time unit and the instantaneous **growth rate** r

$$\Delta N(t) = rN(t)\Delta t$$

Then

$$\frac{\Delta N(t)}{\Delta t} = rN(t)$$

or in differential form

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = rN(t)$$

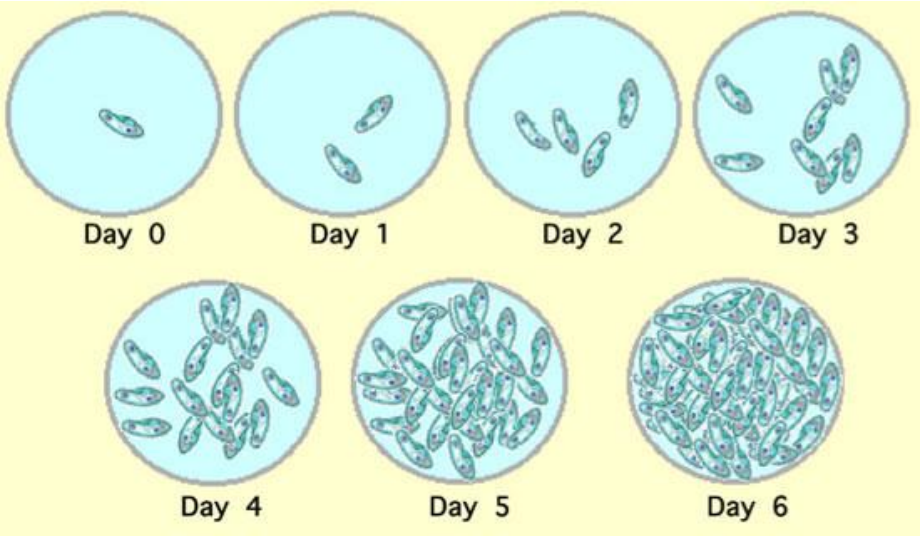
The solution is

$$N(t) = Ce^{rt}$$

If the initial size of the population is known $N(0) = N_0$, then

$$N(t) = N_0 e^{rt}$$

Exponential equation – the simplest model



Changes in a population of Paramecium over a six day period. Each individual in the population divides once per day.

Since each individual in the population divides once per day, then

$$N(t) = 2^t N_0$$

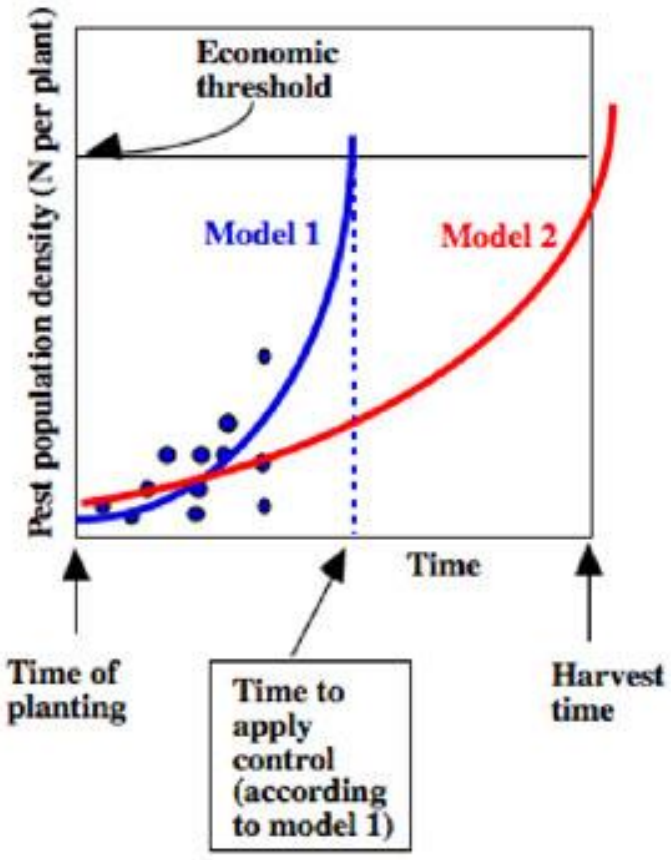
In this case the growth rate is equal

$$r = \ln 2 \Rightarrow N(t) = N_0 e^{\ln 2\, t}$$

Note, that unite of r is [1/day] and depends on the assumed time unit.

Source: Vandermeer, J. (2010) How Populations Grow: The Exponential and Logistic Equations. Nature Education Knowledge 3(10):15

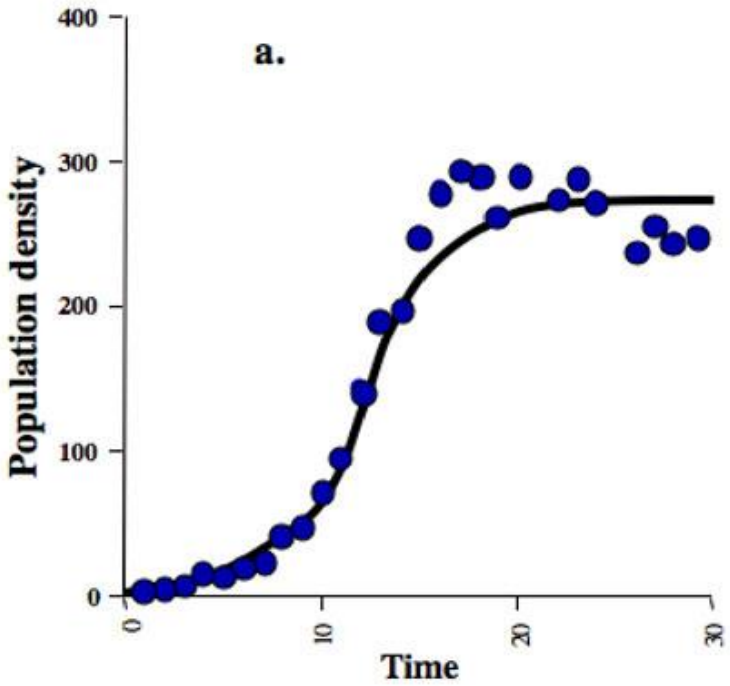
Exponential equation – the simplest model



According to model 1 (relatively large estimate of r), the farmer needs to think about applying a control procedure about half way through the season. According to model 2 (relatively small estimate of r), the farmer need not worry about controlling the pest at all, since its population exceeds the economic threshold only after the harvest. Clearly, it is important to know which model is correct. In this case, according to the available data (blue data points), either model 1 or 2 appears to provide a good fit, leaving the farmer still in limbo.

Source: Vandermeer, J. (2010) How Populations Grow: The Exponential and Logistic Equations. Nature Education Knowledge 3(10):15

Exponential equation – the simplest model



A laboratory population of Paramecium
Note how the density first looks exponential, but later, after the population gets to around 300 cells per cubic centimeter, it levels off. This is the classic logistic pattern.

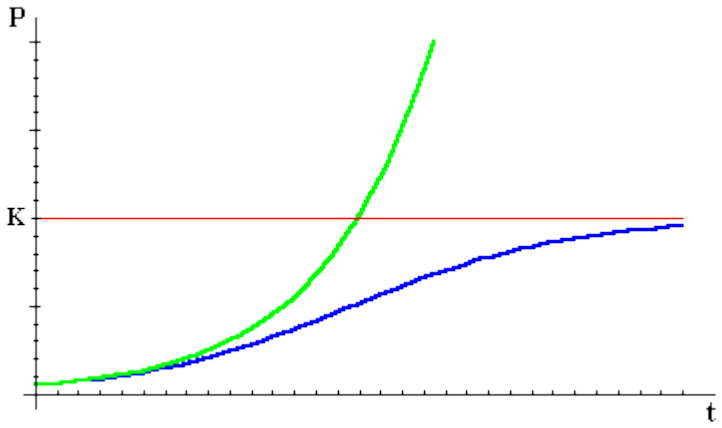
The simplest model is too simple!

Source: Vandermeer, J. (2010) How Populations Grow: The Exponential and Logistic Equations. Nature Education Knowledge 3(10):15

Logistic Equation

Most populations are constrained by limitations on resources. When the population is small relative to K, the two patterns (green and blue) are virtually identical – that is, the constraint does not make much difference. But, for the second population (blue), as N becomes a significant fraction of K, the curves begin to diverge, and as N gets close to K, the growth rate drops to 0.

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = rN(t)\left(1 - \frac{N(t)}{K}\right)$$



Source: https://www.maa.org/book/export/html/115630

Logistic Equation

The equation

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = rN(t)\left(1 - \frac{N(t)}{K}\right)$$

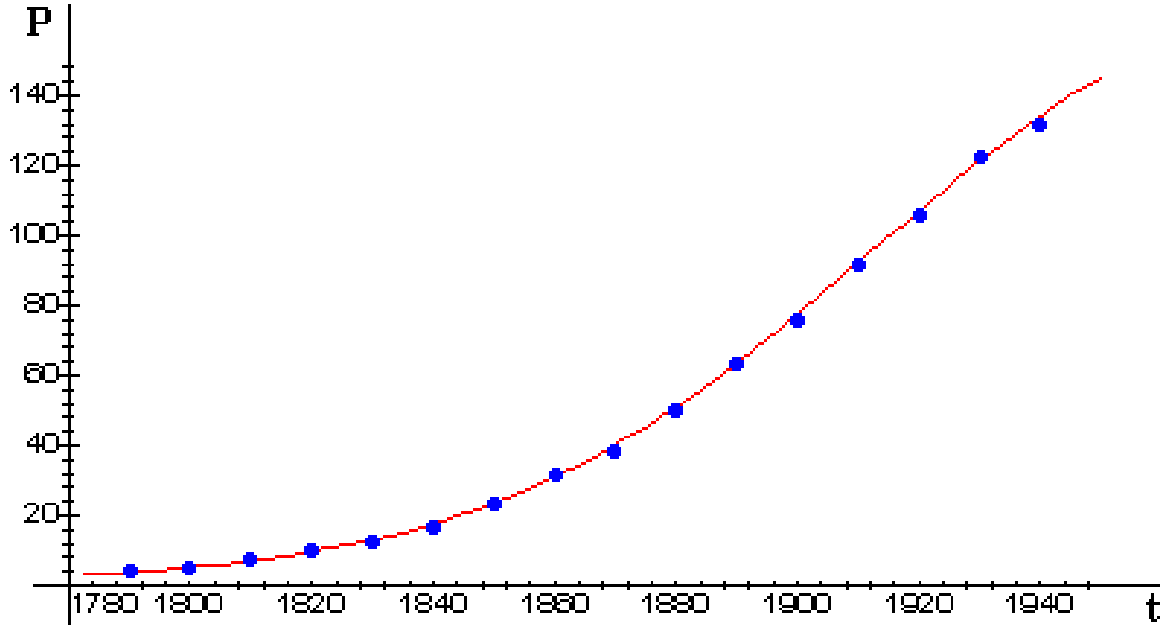
where K represents the **carrying capacity** is called the **logistic growth model** or the **Verhulst model.**

Pierre François Verhulst (1804 - 1849) was a mathematician and a doctor in number theory from the University of Ghent.

Source: https://www.maa.org/book/export/html/115630

Logistic Equation

Using data from the first five U.S. censuses (see the table) Verhulst made a prediction in 1840 of the U.S. population in 1940 and was off by less than 1%.



Source: https://www.maa.org/book/export/html/115630

Date (Years AD)	Population (millions)
1790	3.929
1800	5.308
1810	7.240
1820	9.638
1830	12.866
1840*	17.069

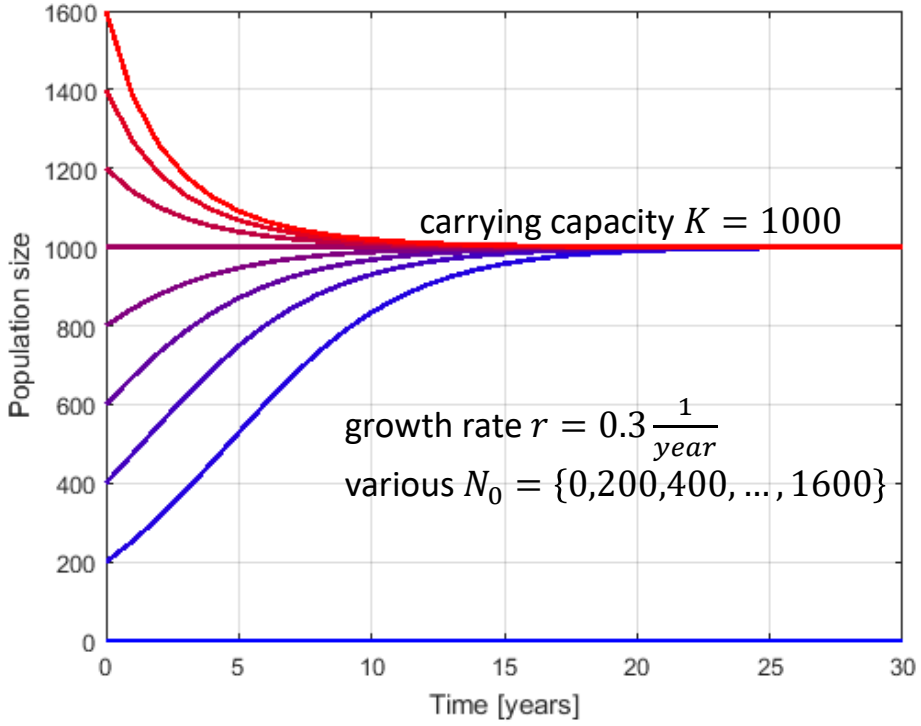
Logistic Equation

The solution of the logistic equation

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = rN(t)\left(1 - \frac{N(t)}{K}\right)$$

with the initial condition $N(t=0) = N_0$ can be found analytically (or in Maple, Matlab, **not** WolframAlpha free)

$$N(t) = \frac{KN_0}{N_0 + (K - N_0)e^{-rt}}$$



The equation has two equilibrium (constant) solutions. To find these solutions we assume $\frac{\mathrm{d}N(t)}{\mathrm{d}t} = 0$, then $rN(t)\left(1 - \frac{N(t)}{K}\right) = 0$. Hence, $N(t) = 0$ (unstable) or $N(t) = K$ (stable).

Logistic Equation – modifications

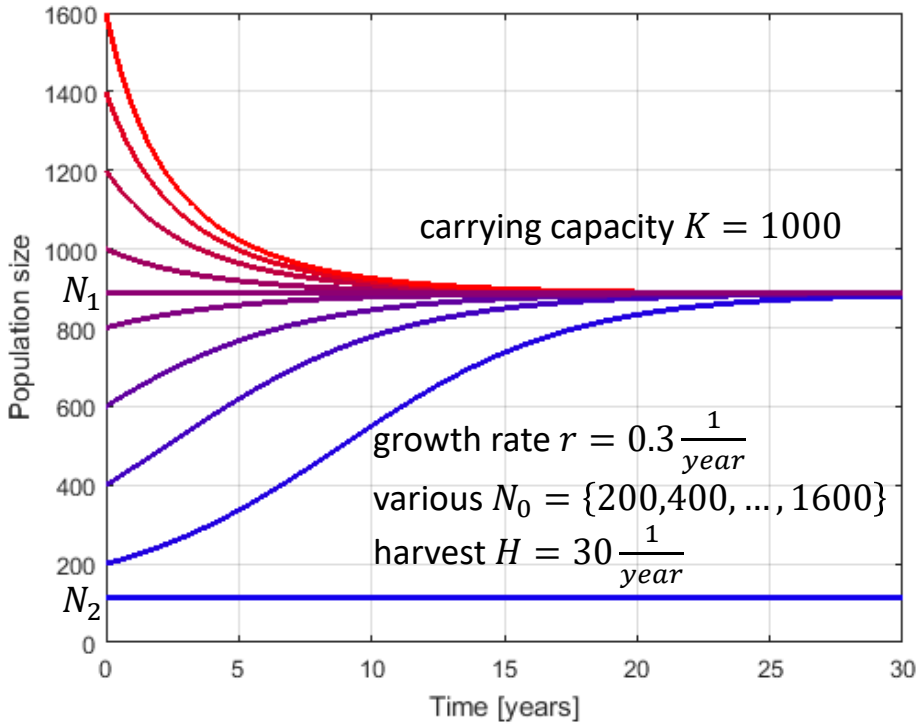
The solution of the logistic equation with constant-rate **harvesting** H

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = rN(t)\left(1 - \frac{N(t)}{K}\right) - H$$

and the initial condition $N(0) = N_0$ can also be found analytically

$$N(t) = \frac{K}{2} - \frac{A}{r}\tan\left(\tan^{-1}\left(\frac{Kr\ -\ 2N_0 r}{2A}\right) + \frac{A}{K}t\right)$$

where parameter $A = \sqrt{KHr\ - \frac{K^2r^2}{4}}$.



This equation has also two equilibrium (constant) solutions. Assuming $\frac{\mathrm{d}N(t)}{\mathrm{d}t} = 0$, one obtains the formula $rN(t)\left(1 - \frac{N(t)}{K}\right) - H = 0$. Hence, $N_1(t) = \frac{K}{2} + \sqrt{\frac{K^2}{4} - \frac{KH}{r}}$ (stable) or $N_2(t) = \frac{K}{2} - \sqrt{\frac{K^2}{4} - \frac{KH}{r}}$ (unstable).

Logistic Equation – modifications

Problem: Suppose a species of fish in a lake is modeled by a logistic population model with relative growth rate of $r = 0.3$ per year and carrying capacity of $K = 10000$. In addition, suppose $H = 400$ fish are harvested from the lake each year. Determine the equilibrium solutions for this model. Analyze the population size in the following 10 years, if the initial number of fish is: a) $N_0 = 1500$, b) $N_0 = 1700$, c) $N_0 = 3000$, d) $N_0 = 9000$. What is the greatest possible rate of harvesting in this lake (critical rate)? What happens if the harvesting rate is greater than the critical rate?

Solution: Since the expressions for the equilibrium solutions are derived,

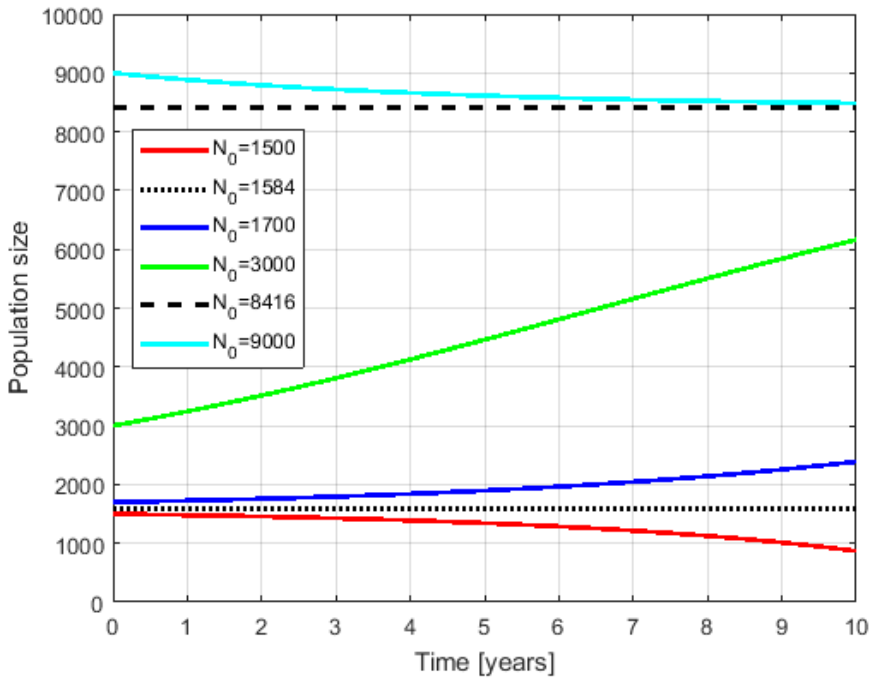
$$N_1(t) = \frac{K}{2} + \sqrt{\frac{K^2}{4} - \frac{KH}{r}} = 8416 \text{ (stable)},$$

$$N_2(t) = \frac{K}{2} - \sqrt{\frac{K^2}{4} - \frac{KH}{r}} = 1584 \text{ (unstable)}.$$

The analytical solution of the logistic equation with harvesting is also known

$$N(t) = \frac{K}{2} - \frac{A}{r}\tan\left(\tan^{-1}\left(\frac{Kr - 2N_0 r}{2A}\right) + \frac{A}{K}t\right)$$

so, the problem boils down to analysis of the above expression for various values of N_0.



Logistic Equation – modifications

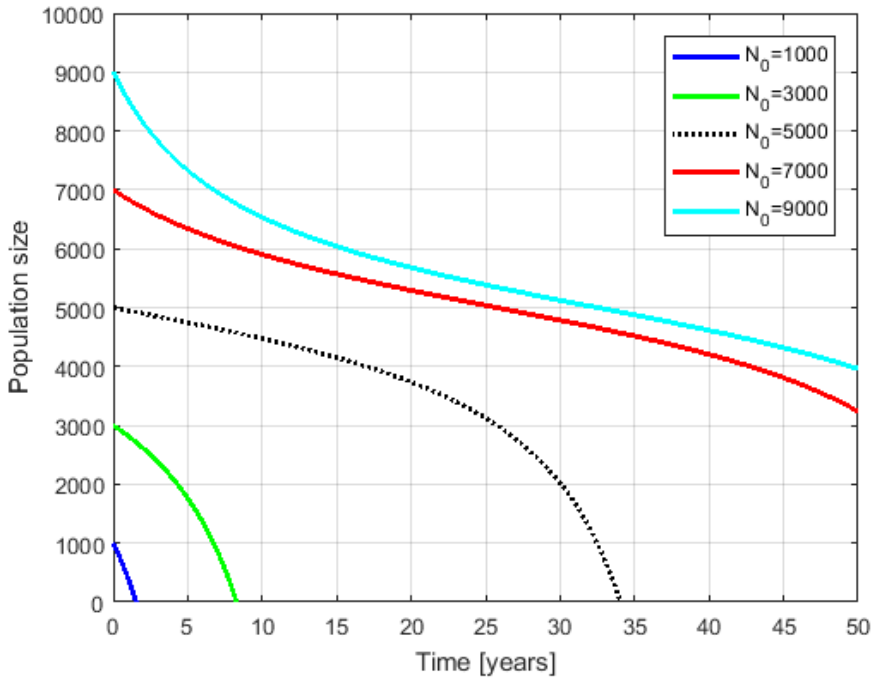
Problem: Suppose a species of fish in a lake is modeled by a logistic population model with relative growth rate of $r = 0.3$ per year and carrying capacity of $K = 10000$. In addition, suppose $H = 400$ fish are harvested from the lake each year. Determine the equilibrium solutions for this model. Analyze the population size in the following 10 years, if the initial number of fish is: a) $N_0 = 1500$, b) $N_0 = 1700$, c) $N_0 = 3000$, d) $N_0 = 9000$. What is the greatest possible rate of harvesting in this lake (critical rate)? What happens if the harvesting rate is greater than the critical rate?

Solution: The **critical harvesting rate** determined from the equilibrium solutions $N_{1,2}(t) = \frac{K}{2} \pm \sqrt{\frac{K^2}{4} - \frac{KH}{r}}$. The expression under the radical must be greater than zero

$$\frac{K^2}{4} - \frac{KH}{r} > 0$$

It results in the following formula $H_{critical} = \frac{rK}{4} = 750$.

Let us consider the situations for $H = 800$ and various N_0. In such a case the equilibrium solutions become complex numbers $N_{1,2}(t) = 5000 \pm 1291i$. The population decays for any initial value (time period is extended to confirm the tendency for higher N_0).

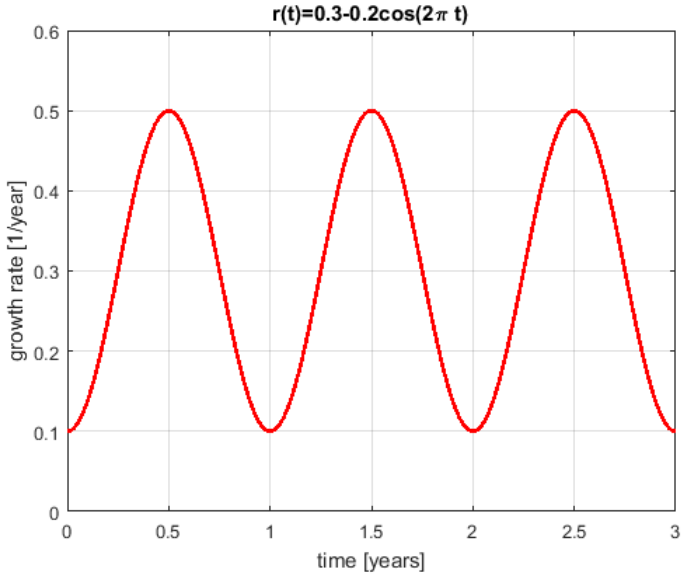


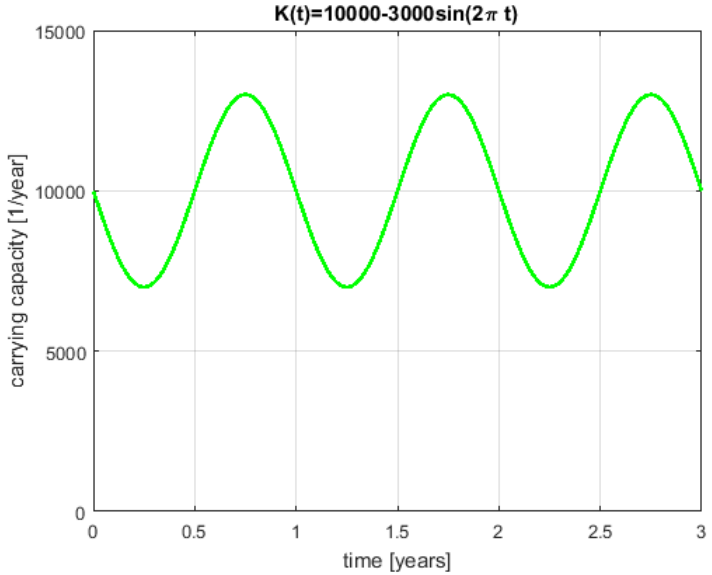
Logistic Equation – modifications

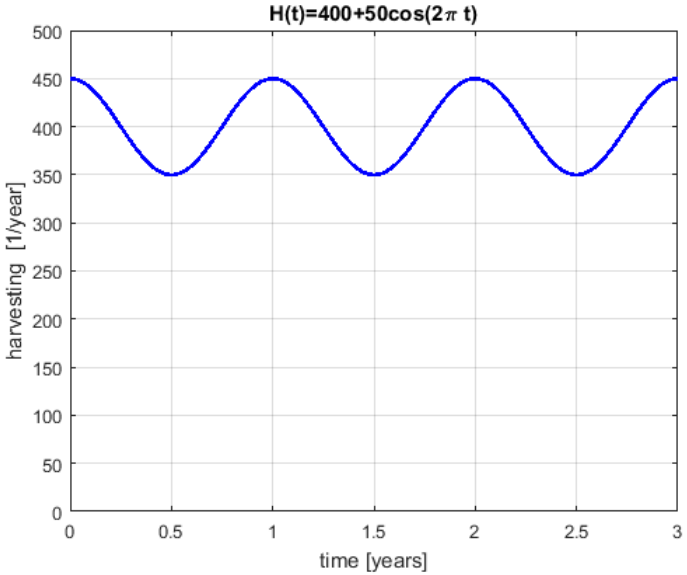
The logistic equation with **time dependent growth rate** $r(t)$, **time dependent carrying capacity** $K(t)$ and **time dependent harvesting** $H(t)$

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = r(t)N(t)\left(1 - \frac{N(t)}{K(t)}\right) - H(t)$$

with the initial condition $N(0) = N_0$, in general, can not be solved analytically.







Logistic Equation – modifications

Finally, the logistic equation has the following form

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = (0.3 - 0.2\cos(2\pi t))N(t)\left(1 - \frac{N(t)}{10000 - 3000\cos(2\pi t)}\right) - (400 + 50\cos(2\pi t))$$

and the initial condition $N(0) = N_0$.

It has to be solved with the use of numerical methods!

Euler method

In computational science, the Euler method (forward Euler method) is a first-order numerical procedure for solving ordinary differential equations (ODEs) with a given initial value. It is the most basic explicit method for numerical integration of ordinary differential equations and is the simplest Runge–Kutta method.

Leonhard Euler (1707 – 1783) was a Swiss mathematician, physicist, astronomer, logician and engineer who made important and influential discoveries in many branches of mathematics like infinitesimal calculus and graph theory while also making pioneering contributions to several branches such as topology and analytic number theory.

Euler method

Let us consider the following ODE of the first order

$$\frac{\mathrm{d}f}{\mathrm{d}t} = R[t, f(t)]$$

with the initial condition $f(0) = f_0$.

For instance, if $\frac{\mathrm{d}f}{\mathrm{d}t} = f(t) + t$ with $f(0) = 1$, then $R[t, f(t)] = f(t) + t$ and $f_0 = 1$.

Starting from the definition of the derivative

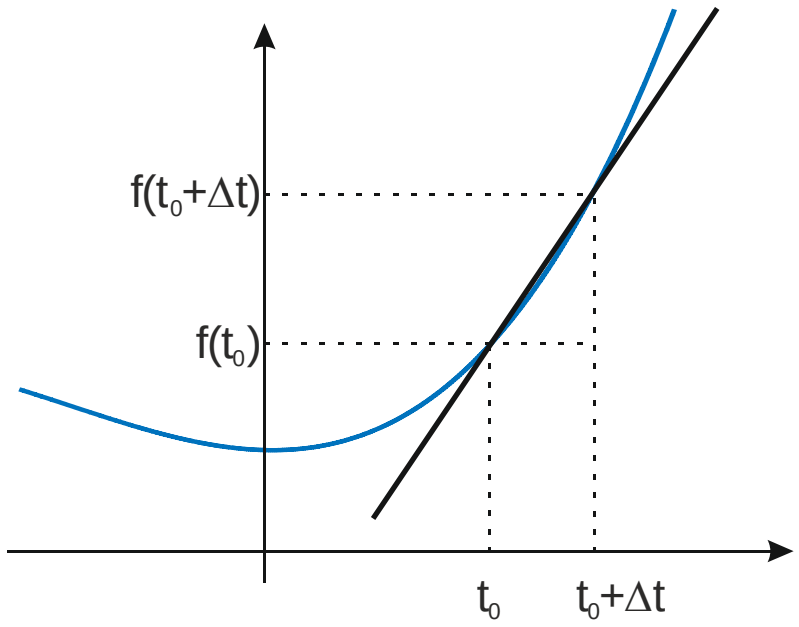
$$\frac{\mathrm{d}f}{\mathrm{d}t}(t_0) = \lim_{\Delta t \to 0} \frac{f(t_0 + \Delta t) - f(t_0)}{\Delta t}$$

one can write the following expression for a „very small Δt"

$$\frac{\mathrm{d}f}{\mathrm{d}t}(t_0) \approx \frac{f(t_0 + \Delta t) - f(t_0)}{\Delta t}$$

Then

$$\frac{f(t_0 + \Delta t) - f(t_0)}{\Delta t} \approx R[t_0, f(t_0)]$$



Euler method

The approximation can be applied for any time t_n

$$\frac{f(t_n + \Delta t) - f(t_n)}{\Delta t} \approx R[t_n, f(t_n)]$$

Assuming that $t_n = n\Delta t$ for the whole considered time domain, one obtains

$$\frac{f(t_{n+1}) - f(t_n)}{\Delta t} \approx R[t_n, f(t_n)]$$

Or in a more useful form

$$f(t_{n+1}) \approx f(t_n) + \Delta t R[t_n, f(t_n)]$$

It means that for a known $f(0)$, a sequence of $f(t_n)$ can be found iteratively!

Euler method – example

Problem: Find the solution of the equation $\frac{\mathrm{d}f}{\mathrm{d}t} = f(t) + t$ with $f(0) = 1$ using Euler method. Compare the numerical results with the analytical solution for $t \in [0,1]$.

Solution: Using the Euler's formula

$$f(t_{n+1}) \approx f(t_n) + \Delta t[f(t_n) + t_n]$$

with the initial value $f(t_0 = 0) = f_0 = 1$ and different values of Δt, we obtain the results collected in the next slide.

The analytical solution can be found using www.wolframalpha.com

$$f(t) = 2e^t - t - 1$$

```
MATLAB CODE EXAMPLE:

dt=0.1;
fn=1;
f=[fn]
for n=1:10
    tn=n*dt;
    fn1=fn+dt*(fn+tn);
    f=[f fn1];
    fn=fn1;
end
```

Euler method – example

Problem: Find the solution of the equation $\frac{\mathrm{d}f}{\mathrm{d}t} = f(t) + t$ with $f(0) = 1$ using Euler method. Compare the numerical results with the analytical solution for $t \in [0,1]$.

Solution: $\Delta t = 0.1$ (all steps)

t_n	Euler	Exact	Error [%]
0.0	1.0000	1.0000	0.0000
0.1	1.1100	1.1103	0.0308
0.2	1.2410	1.2428	0.1453
0.3	1.3951	1.3997	0.3299
0.4	1.5746	1.5836	0.5708
0.5	1.7821	1.7974	0.8552
0.6	2.0203	2.0442	1.1721
0.7	2.2923	2.3275	1.5123
0.8	2.6015	2.6511	1.8689
0.9	2.9517	3.0192	2.2362
1.0	3.3469	3.4366	2.6103

$\Delta t = 0.01$ (only 11 from 101 iterations)

t_n	Euler	Exact	Error [%]
0.0	1.0000	1.0000	0.0000
0.1	1.1103	1.1103	0.0046
0.2	1.2426	1.2428	0.0180
0.3	1.3992	1.3997	0.0387
0.4	1.5826	1.5836	0.0652
0.5	1.7957	1.7974	0.0964
0.6	2.0416	2.0442	0.1310
0.7	2.3236	2.3275	0.1680
0.8	2.6456	2.6511	0.2069
0.9	3.0118	3.0192	0.2469
1.0	3.4267	3.4366	0.2877

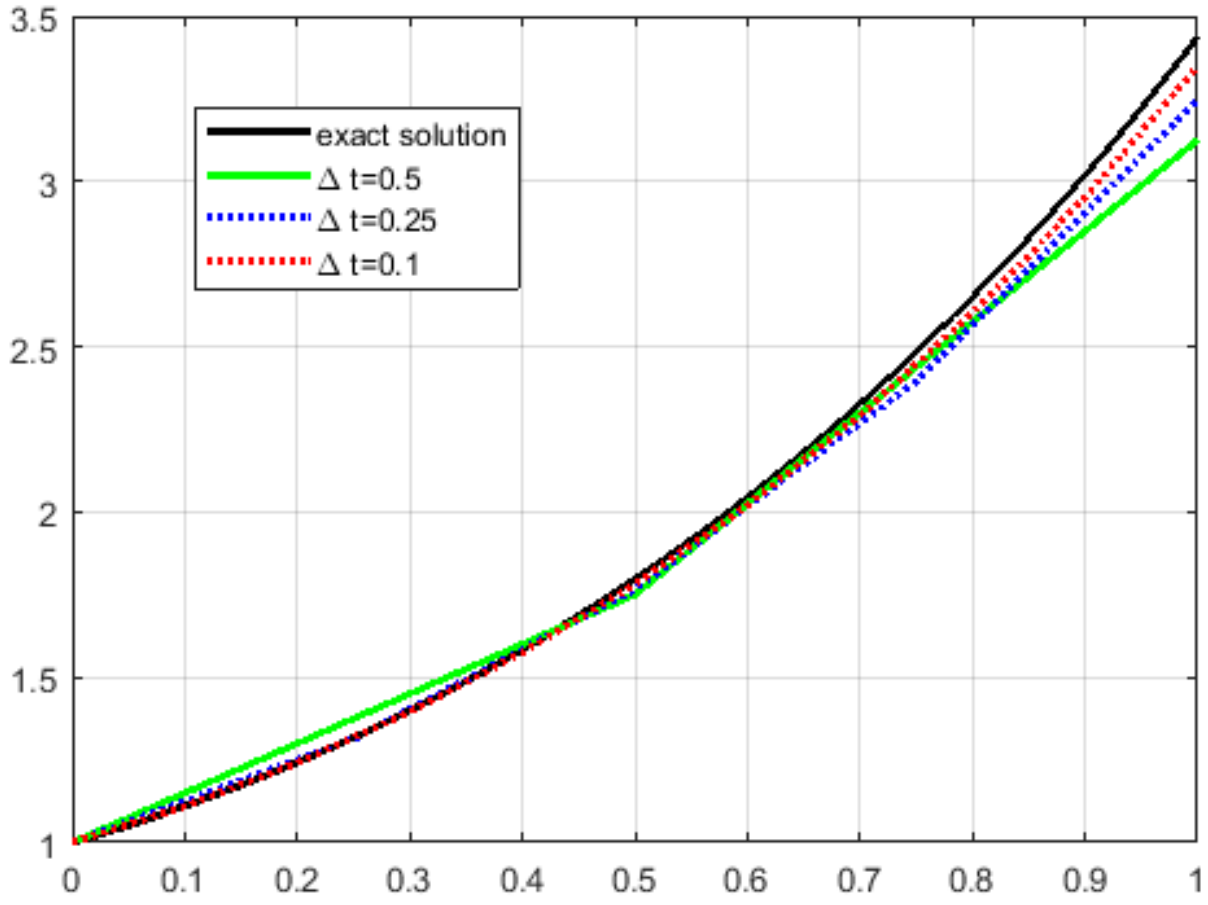
$\Delta t = 0.001$ (only 11 from 1001 iterations)

t_n	Euler	Exact	Error [%]
0.0	1.0000	1.0000	0.0000
0.1	1.1103	1.1103	0.0005
0.2	1.2428	1.2428	0.0018
0.3	1.3997	1.3997	0.0039
0.4	1.5835	1.5836	0.0066
0.5	1.7973	1.7974	0.0098
0.6	2.0440	2.0442	0.0132
0.7	2.3271	2.3275	0.0170
0.8	2.6505	2.6511	0.0209
0.9	3.0185	3.0192	0.0249
1.0	3.4356	3.4366	0.0291

Euler method – example

Problem: Find the solution of the equation $\frac{\mathrm{d}f}{\mathrm{d}t} = f(t) + t$ with $f(0) = 1$ using Euler method. Compare the numerical results with the analytical solution for $t \in [0,1]$.

Solution:



Euler method – example

Problem: Using Euler method find the solution of the modified logistic equation

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = r(t)N(t)\left(1 - \frac{N(t)}{\mathrm{K}(t)}\right) - H(t)$$

where $r(t) = 0.3 - 0.2\cos(2\pi t)$, $\mathrm{K}(t) = 10000 - 3000\cos(2\pi t)$, $H(t) = 400 + 50\cos(2\pi t)$ with a) $N(0) = 1500$, b) $N(0) = 1700$, c) $N(0) = 5000$, d) $N(0) = 10000$. The analysis period $t \in [0,50]$.

Solution: Using the Euler's formula

$$N(t_{n+1}) \approx N(t_n) + \Delta t\left[r(t_n)N(t_n)\left(1 - \frac{N(t_n)}{\mathrm{K}(t_n)}\right) - H(t_n)\right]$$

with the known initial value $N(0)$, we can perform the simulations for various Δt.

```
MATLAB CODE EXAMPLE:

dt=0.1;
fn=1500;
f=[fn]
for n=1:500
    tn=n*dt;
    r=0.3-0.2*cos(2*pi*tn);
    K=10000-3000*cos(2*pi*tn);
    H=400+50*cos(2*pi*tn);
    fn1=fn+dt*(r*fn*(1-fn/K)-H);
    f=[f fn1];
    fn=fn1;
end
```

Euler method – example

Problem: Using Euler method find the solution of the modified logistic equation

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = r(t)N(t)\left(1 - \frac{N(t)}{\mathrm{K}(t)}\right) - H(t)$$

where $r(t) = 0.3 - 0.2\cos(2\pi t)$, $\mathrm{K}(t) = 10000 - 3000\cos(2\pi t)$, $H(t) = 400 + 50\cos(2\pi t)$ with a) $N(0) = 1500$, b) $N(0) = 1700$, c) $N(0) = 5000$, d) $N(0) = 10000$. The analysis period $t \in [0,50]$.

Solution: The results for

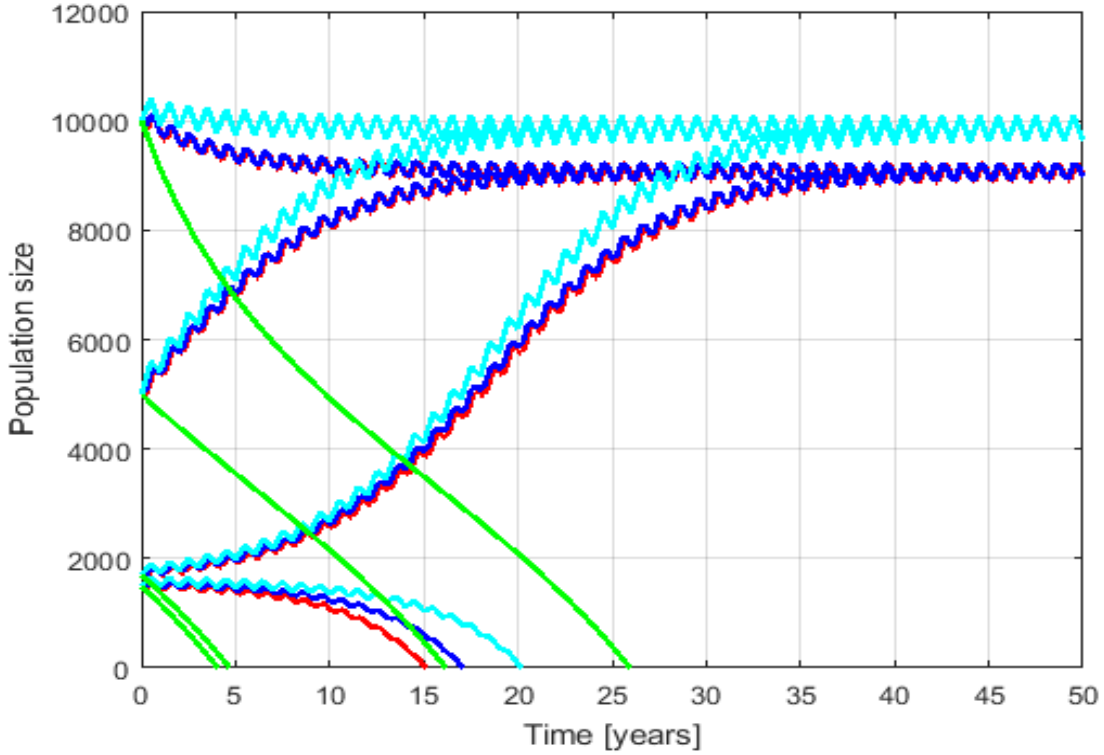
a) $N(0) = 1500$,

b) $N(0) = 1700$,

c) $N(0) = 5000$,

d) $N(0) = 10000$,

with $\Delta t = 1.0$ (green), $\Delta t = 0.5$ (cyan), $\Delta t = 0.25$ (blue), $\Delta t = 0.1$ (red).



Δt must be small!!!

Predator–prey equations

The predator–prey (Lotka–Volterra) equations, are first-order, nonlinear, differential equations which can be used to describe the interaction of two species, one as a predator and the other as prey

$$\begin{cases} \dfrac{\mathrm{d}M(t)}{\mathrm{d}t} = \alpha M(t) - \beta M(t)N(t) \\ \dfrac{\mathrm{d}N(t)}{\mathrm{d}t} = \delta M(t)N(t) - \gamma N(t) \end{cases}$$

where:

$M(t)$ number of prey (for example, rabbits)

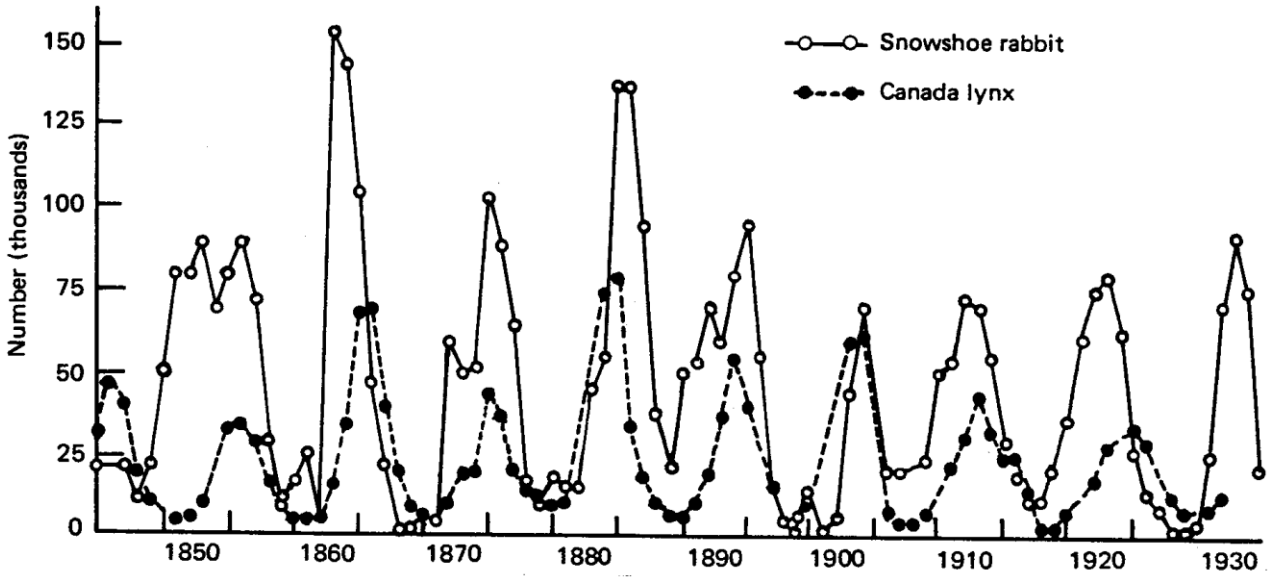
$N(t)$ number of predator (for example, foxes)

http://apbioecologyproject.weebly.com/uploads/4/9/7/2/49723057/1428082584.png

meaning:

- α growth rate of the prey
- β rate at which the predators and the prey meet
- δ growth rate of the predator
- γ loss rate of the predator (death or emigration)

Predator–prey equations



Source: https://globalchange.umich.edu/globalchange1/current/lecture

Predator–prey equations

Problem: Using Euler method find the solution of the predator–prey equations system

$$\begin{cases} \dfrac{\mathrm{d}M(t)}{\mathrm{d}t} = \alpha M(t) - \beta M(t)N(t) \\ \dfrac{\mathrm{d}N(t)}{\mathrm{d}t} = \delta M(t)N(t) - \gamma N(t) \end{cases}$$

where $\alpha = 0.2$, $\beta = 0.001$, $\delta = 0.001$ and $\gamma = 0.5$ for $t \in [0,100]$. The initial numbers of prey and predator are equal $\mathrm{M}_0 = 1200$ and $\mathrm{N}_0 = 300$.

Solution: The Euler method can by applied for both equations simultaneously

$$\begin{cases} M(t_{n+1}) \approx M(t_n) + \Delta t[\alpha M(t_n) - \beta M(t_n)N(t_n)] \\ N(t_{n+1}) \approx N(t_n) + \Delta t[\delta M(t_n)N(t_n) - \gamma N(t_n)] \end{cases}$$

If the both initial values are known: $M(0) = 1200$ and $N(0) = 300$.

```
MATLAB CODE EXAMPLE:

dt=0.01;
a=0.2;
b=0.001;
d=0.001;
g=0.5;
Mn=1200;
Nn=300;
M=[Mn];
N=[Nn];
for n=1:10000
    tn=n*dt
    Mn1=Mn+dt*(a*Mn-b*Mn*Nn);
    Nn1=Nn+dt*(d*Mn*Nn-g*Nn);
    M=[M Mn1];
    N=[N Nn1];
    Mn=Mn1;
    Nn=Nn1;
end
```

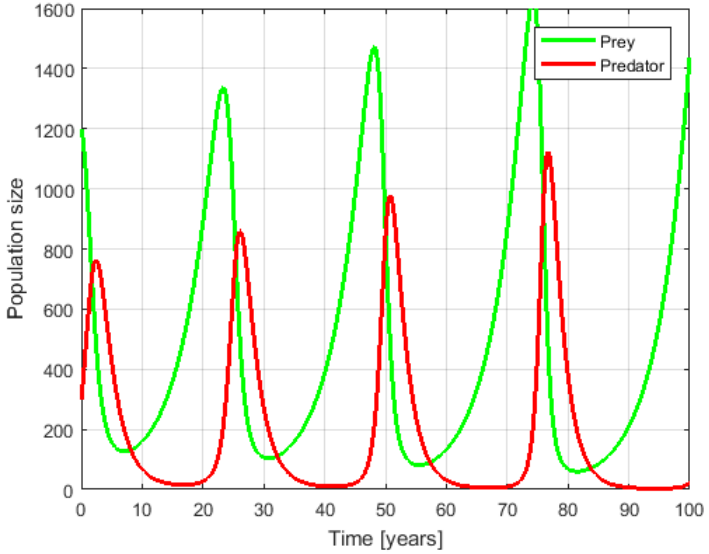
Predator–prey equations

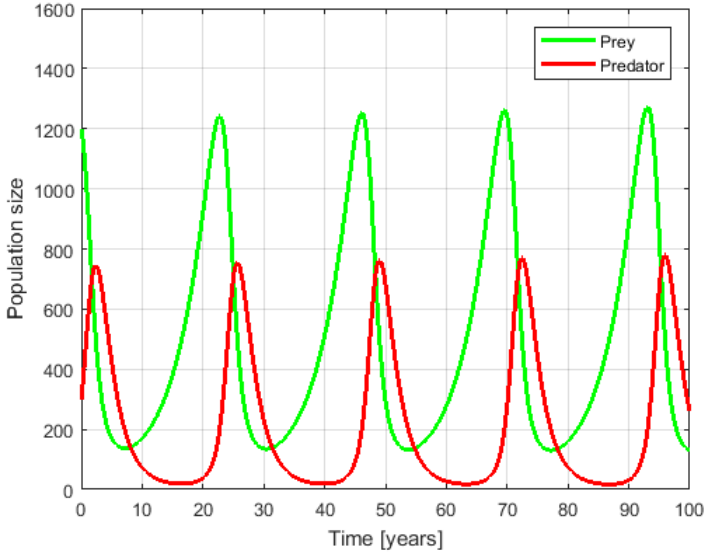
Problem: Using Euler method find the solution of the predator–prey equations system

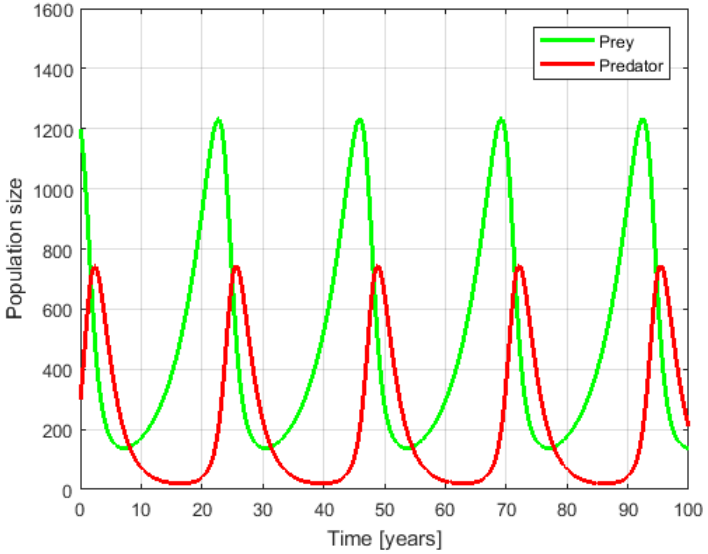
$$\frac{\mathrm{d}M(t)}{\mathrm{d}t} = \alpha M(t) - \beta M(t)N(t) \qquad \frac{\mathrm{d}N(t)}{\mathrm{d}t} = \delta M(t)N(t) - \gamma N(t)$$

where $\alpha = 0.2$, $\beta = 0.001$, $\delta = 0.001$ and $\gamma = 0.5$ for $t \in [0,100]$. The initial numbers of prey and predator are equal $\mathrm{M}_0 = 1200$ and $\mathrm{N}_0 = 300$.

Solution: $\Delta t = 0.1$ $\Delta t = 0.01$ $\Delta t = 0.001$







The accuracy and stability of the Euler method is poor!

Midpoint method

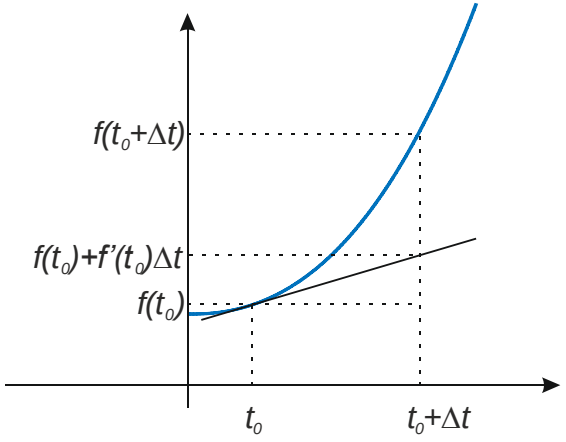
Let us once again consider the following ODE of the first order

$$\frac{\mathrm{d}f}{\mathrm{d}t} = R[t, f(t)]$$

with the initial condition $f(0) = f_0$.

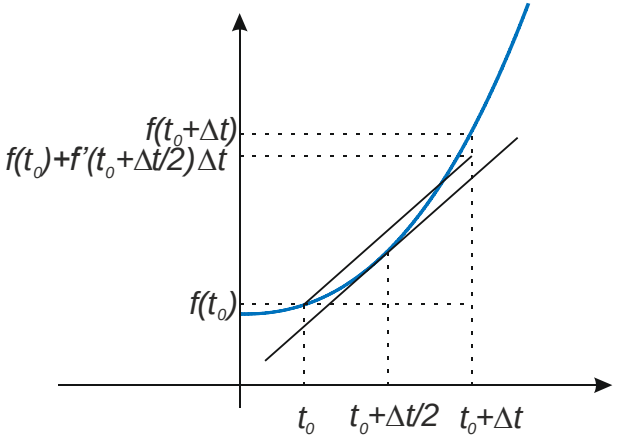
The definition of the derivative $\frac{\mathrm{d}f}{\mathrm{d}t}(t_0) = \lim_{\Delta t \to 0} \frac{f(t_0+\Delta t)-f(t_0)}{\Delta t}$ concerns exactly one point t_0, since $\Delta t \to 0$. In numerical approaches Δt is finite, so the relations between $f(t_0 + \Delta t)$ and $f(t_0)$ is inaccurate.

$$\frac{f(t_0 + \Delta t) - f(t_0)}{\Delta t} \approx \frac{\mathrm{d}f}{\mathrm{d}t}(t_0)$$



OR

$$\frac{f(t_0 + \Delta t) - f(t_0)}{\Delta t} \approx \frac{\mathrm{d}f}{\mathrm{d}t}\left(t_0 + \frac{\Delta t}{2}\right)$$



Midpoint method

Hence, the equation

$$\frac{\mathrm{d}f}{\mathrm{d}t} = R[t, f(t)] \tag{1}$$

can be approximated with the formula (more accurate than in Euler method)

$$\frac{f(t_0+\Delta t)-f(t_0)}{\Delta t} = R\left[t_0 + \frac{\Delta t}{2}, f\left(t_0 + \frac{\Delta t}{2}\right)\right] \tag{2}$$

However, since $f\left(t_0 + \frac{\Delta t}{2}\right)$ is unknown, the formula can not be applied directly. But using simple approximation

$$f\left(t_0 + \frac{\Delta t}{2}\right) \approx f(t_0) + \frac{\mathrm{d}f}{\mathrm{d}t}(t_0)\frac{\Delta t}{2} = f(t_0) + R[t_0, f(t_0)]\frac{\Delta t}{2} \tag{3}$$

the relations (2) can be modified

$$\frac{f(t_0+\Delta t)-f(t_0)}{\Delta t} = R\left[t_0 + \frac{\Delta t}{2}, f(t_0) + R[t_0, f(t_0)]\frac{\Delta t}{2}\right] \tag{4}$$

Midpoint method

Finally, the midpoint formula has the following form

$$f(t_0 + \Delta t) = f(t_0) + \Delta t R\left[t_0 + \frac{\Delta t}{2}, f(t_0) + R[t_0, f(t_0)]\frac{\Delta t}{2}\right]$$

Assuming that $t_n = n\Delta t$ for the whole considered time domain, one obtains

$$f(t_{n+1}) = f(t_n) + \Delta t R\left[t_n + \frac{\Delta t}{2}, f(t_n) + R[t_n, f(t_n)]\frac{\Delta t}{2}\right]$$

The accuracy and stability of the midpoint algorithm is much higher than Euler method!

Midpoint method– example

Problem: Using the midpoint method find the solution of the equation

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = f(t)\sin(t)$$

Compare the results with the ones obtained from Euler method for $t \in [0,20]$ and two time steps $\Delta t = 0.1$ and $\Delta t = 0.01$.

$$\boxed{f(t_{n+1}) = f(t_n) + \Delta t R\left[t_n + \frac{\Delta t}{2}, f(t_n) + R[t_n, f(t_n)]\frac{\Delta t}{2}\right]}$$

Solution: Using the midpoint method where

$$R[t_n, f(t_n)] = f(t_n)\sin(t_n)$$

$$f(t_{n+1}) \approx f(t_n) + \Delta t\left[\left(f(t_n) + R[t_n, f(t_n)]\frac{\Delta t}{2}\right)\sin\left(t_n + \frac{\Delta t}{2}\right)\right]$$

Midpoint method– example

Problem: Using the midpoint method find the solution of the equation

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = f(t)\sin(t)$$

Compare the results with the ones obtained from Euler method for $t \in [0,20]$ and two time steps $\Delta t = 0.1$ and $\Delta t = 0.01$.

```
MATLAB CODE EXAMPLE:

dt=0.1;
fn=1;
f=[fn];
N=20/dt;
tt=[0];
for n=1:N
    tn=n*dt
    R=fn*sin(tn);
    fn1=fn+dt*((fn+R*dt/2)*sin(tn+dt/2));
    f=[f fn1];
    fn=fn1;
end
```

Solution: The implementation is quite easy

$$R[t_n, f(t_n)] = f(t_n)\sin(t_n)$$

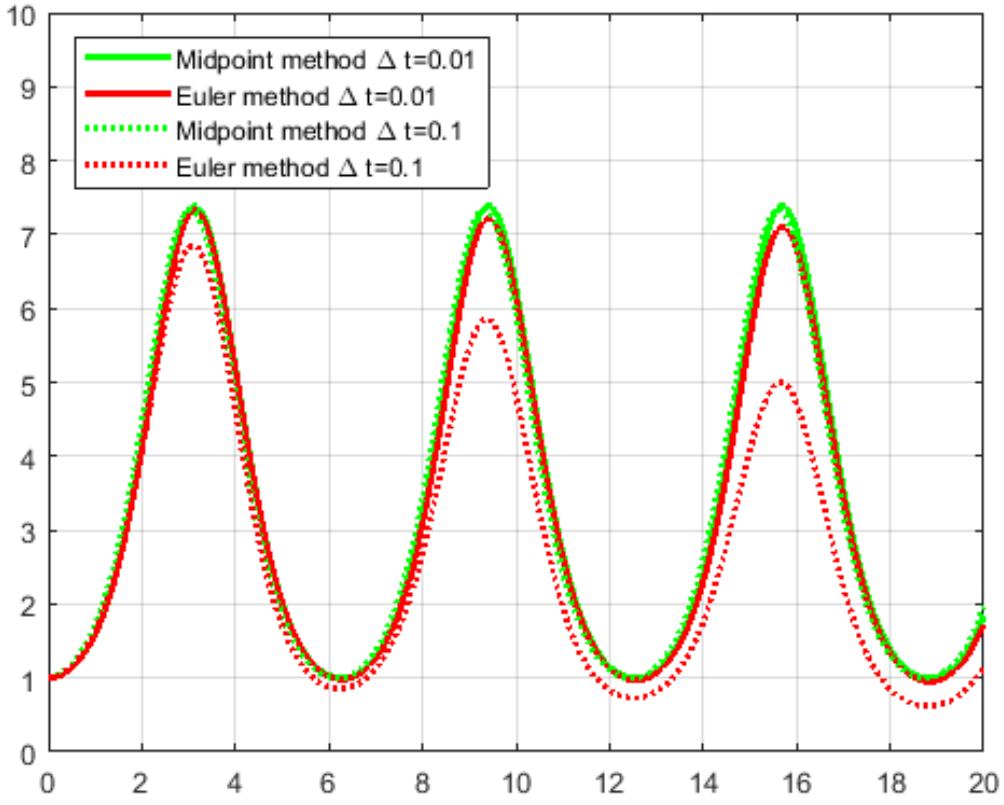
$$f(t_{n+1}) \approx f(t_n) + \Delta t\left[\left(f(t_n) + R[t_n, f(t_n)]\frac{\Delta t}{2}\right)\sin\left(t_n + \frac{\Delta t}{2}\right)\right]$$

Midpoint method– example

Problem: Using the midpoint method find the solution of the equation

$$\frac{\mathrm{d}f(t)}{\mathrm{d}t} = f(t)\sin(t)$$

Compare the results with the ones obtained from Euler method for $t \in [0,20]$ and two time steps $\Delta t = 0.1$ and $\Delta t = 0.01$.



Solution: The implementation is quite easy

$$R[t_n, f(t_n)] = f(t_n)\sin(t_n)$$

$$f(t_{n+1}) \approx f(t_n) + \Delta t\left[\left(f(t_n) + R[t_n, f(t_n)]\frac{\Delta t}{2}\right)\sin\left(t_n + \frac{\Delta t}{2}\right)\right]$$

Modeling and Simulation of Systems

Lecture 5: Methods for the numerical solution of the second-order differential equation

Piotr Kowalczyk

Gravity pendulum

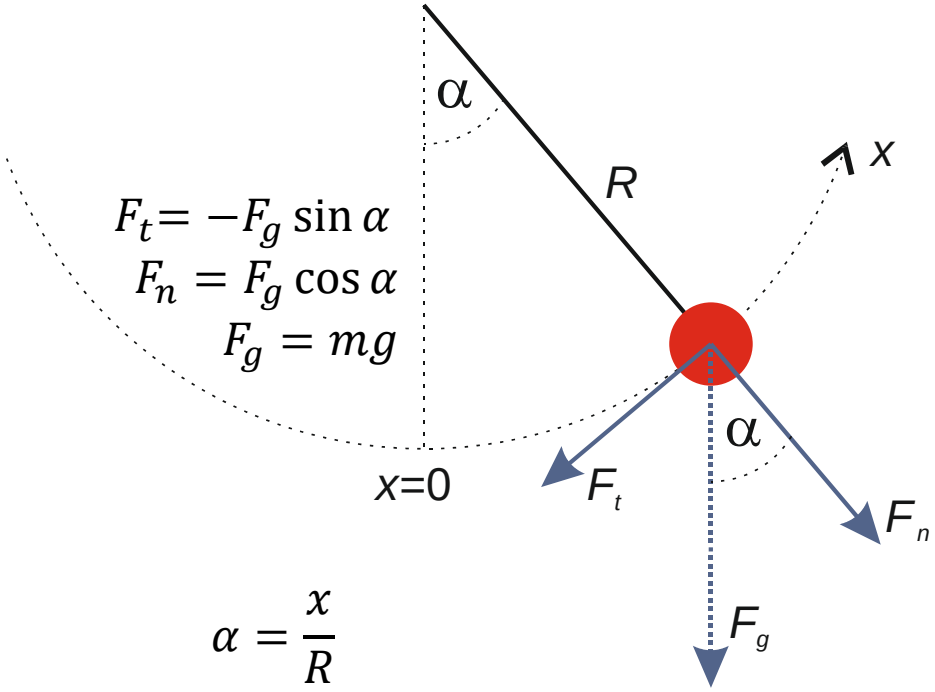
Problem 5.1: Use the Euler method to simulate a simple damped pendulum of length $R = 10$ m ended with a point mass $m = 1$ kg. Assume, that the frictional force can be modeled as being proportional to the velocity of the object $F_f = -bv$, where $b = 0.1$ Ns/m is called the viscous damping coefficient. Consider the amplitudes beyond the small angle approximation.

Solution: According to the Newton's second law, we have

$$ma(t) = -bv(t) - mg\sin\alpha(t)$$

or in the direct form (nonlinear second-order ordinary differential equation)

$$m\frac{d^2x}{dt^2}(t) = -b\frac{dx}{dt}(t) - mg\sin\left(\frac{x(t)}{R}\right)$$



$$\alpha = \frac{x}{R}$$

Gravity pendulum

Problem 5.1: Use the Euler method to simulate a simple damped pendulum of length $R = 10$ m ended with a point mass $m = 1$ kg. Assume, that the frictional force can be modeled as being proportional to the velocity of the object $F_f = -bv$, where $b = 0.1$ Ns/m is called the viscous damping coefficient. Consider the amplitudes beyond the small angle approximation.

Solution: The Euler method is dedicated for first–order differential equation and we have to separate the problem into the system of two equations

$$m\frac{d^2x}{dt^2}(t) = -b\frac{dx}{dt}(t) - mg\sin\left(\frac{x(t)}{R}\right) \Rightarrow \begin{cases} \frac{dv}{dt}(t) = -\frac{b}{m}v(t) - g\sin\left(\frac{x(t)}{R}\right) \\ \frac{dx}{dt}(t) = v(t) \end{cases}$$

Then, we can apply the Euler's formula

$$\begin{cases} v(t_{n+1}) = v(t_n) + \Delta t\left[-\frac{b}{m}v(t_n) - g\sin\left(\frac{x(t_n)}{R}\right)\right] \\ x(t_{n+1}) = x(t_n) + \Delta t v(t_n) \end{cases}$$

$$\frac{\mathrm{d}f}{\mathrm{d}t} = R[t, f(t)]$$

$$f(t_{n+1}) \approx f(t_n) + \Delta t R[t_n, f(t_n)]$$

Gravity pendulum

Problem 5.1: Use the Euler method to simulate a simple damped pendulum of length $R = 10$ m ended with a point mass $m = 1$ kg. Assume, that the frictional force can be modeled as being proportional to the velocity of the object $F_f = -bv$, where $b = 0.1$ Ns/m is called the viscous damping coefficient. Consider the amplitudes beyond the small angle approximation.

Solution: The implementation of the problem is quite simple

$$\begin{cases} v(t_{n+1}) = v(t_n) + \Delta t \left[-\frac{b}{m} v(t_n) - g \sin\left(\frac{x(t_n)}{R}\right) \right] \\ x(t_{n+1}) = x(t_n) + \Delta t v(t_n) \end{cases}$$

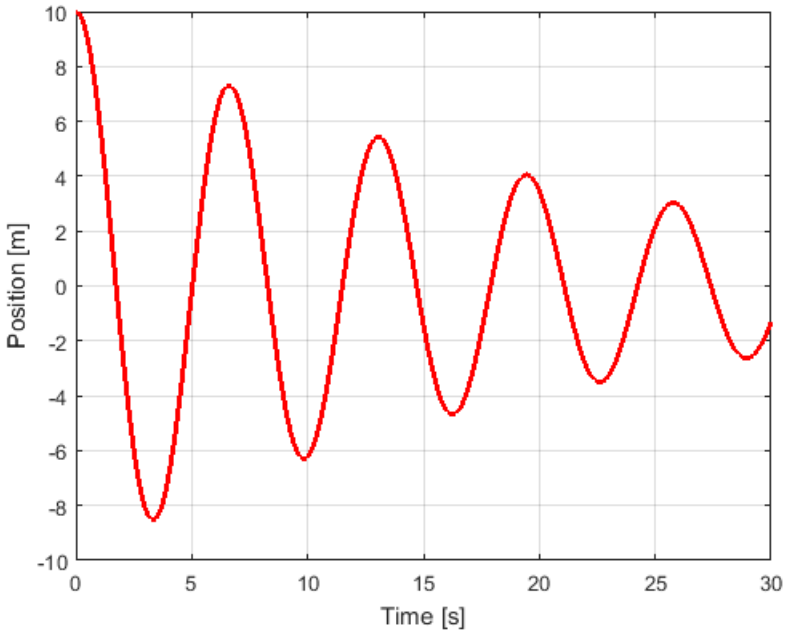
If the initial values of position and velocity are known $x(t_1) = x_0$ and $v(t_1) = v_0$.

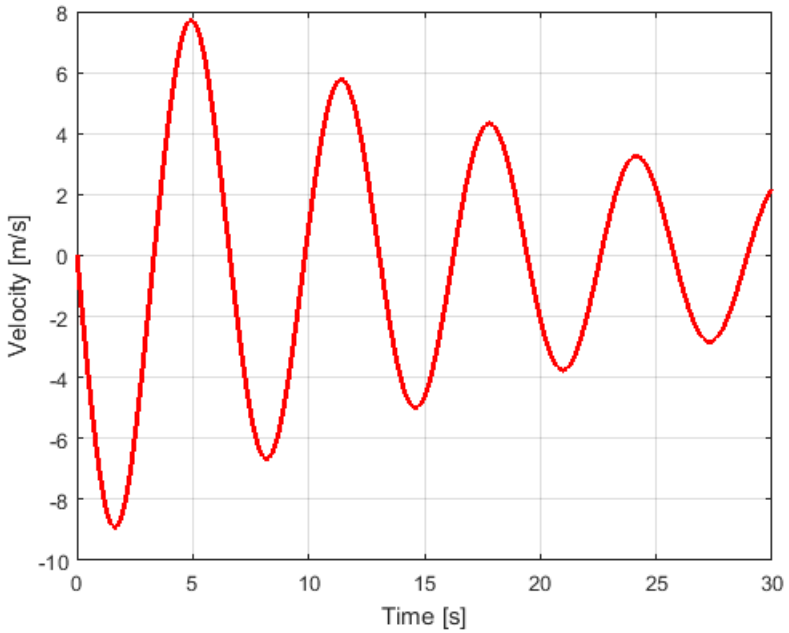
```
MATLAB CODE EXAMPLE:

g=10; m=1; R=10; b=0.1;
vn=0;
xn=10;
dt=0.01;
T=30;
v=[vn];
x=[xn];
for t=dt:dt:T
    vn1=vn+dt*(-b/m*vn-g*sin(xn/R));
    xn1=xn+dt*vn;
    v=[v vn1];
    x=[x xn1];
    vn=vn1;
    xn=xn1;
end
```

Gravity pendulum

Problem 5.1: Use the Euler method to simulate a simple damped pendulum of length $R = 10$ m ended with a point mass $m = 1$ kg. Assume, that the frictional force can be modeled as being proportional to the velocity of the object $F_f = -bv$, where $b = 0.1$ Ns/m is called the viscous damping coefficient. Consider the amplitudes beyond the small angle approximation.

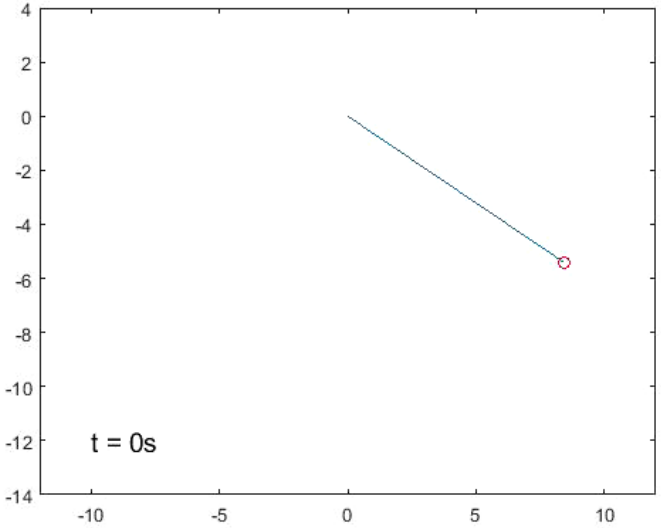
Solution: The results of the simulation for: $\Delta t = 0.01$ s, the initial position $x_0 = 10$ m and the initial velocity $v_0 = 0$ m/s:





Gravity pendulum

Problem 5.1: Use the Euler method to simulate a simple damped pendulum of length $R = 10$ m ended with a point mass $m = 1$ kg. Assume, that the frictional force can be modeled as being proportional to the velocity of the object $F_f = -bv$, where $b = 0.1$ Ns/m is called the viscous damping coefficient. Consider the amplitudes beyond the small angle approximation.



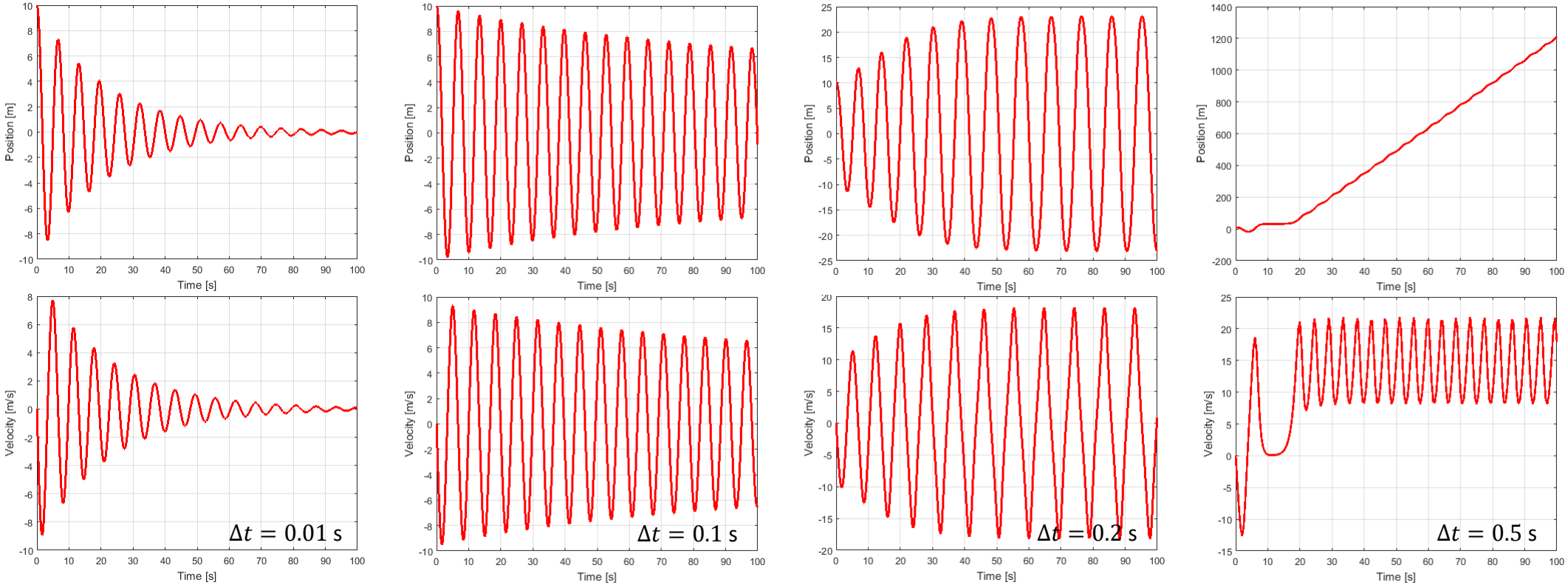
Solution: Also the visualization is quite simple

```
MATLAB CODE EXAMPLE:

n=0;
for t=0:dt:T
    n=n+1;
    xn=x(n);
    alpha=xn/R;
    xx=R*cos(-pi/2+alpha);
    yy=R*sin(-pi/2+alpha);
    plot([0 xx], [0 yy]);
    hold on
    plot(xx, yy,'ro')
    hold off
    xlim([-1.2*R 1.2*R])
    ylim([-1.4*R 0.4*R])
    drawnow;
end
```

Gravity pendulum

Problem 5.1: Use the Euler method to simulate a simple damped pendulum of length $R = 10$ m ended with a point mass $m = 1$ kg. Assume, that the frictional force can be modeled as being proportional to the velocity of the object $F_f = -bv$, where $b = 0.1$ Ns/m is called the viscous damping coefficient. Consider the amplitudes beyond the small angle approximation.

Solution: The results for: the initial position $x_0 = 10$ m, the initial velocity $v_0 = 0$ m/s



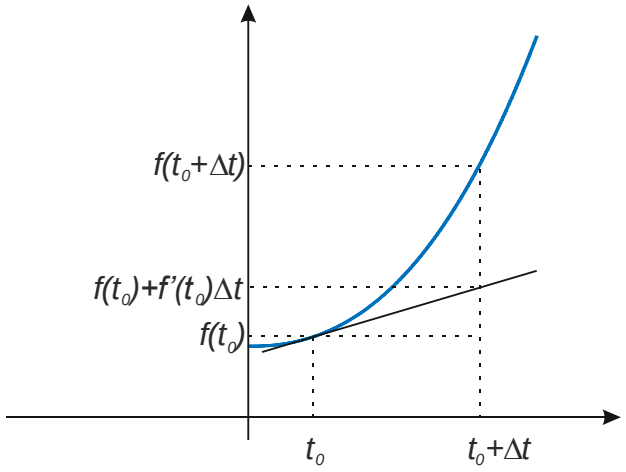
Introduction to finite difference methods

In Euler, midpoint, Runge-Kutta and finite difference method the problem boils down to determining the most suitable relations (involving derivatives) between the function values in some finite distance

$$f(t_0) \xrightarrow[\text{of the function and the step } \Delta t]{\text{relation involving derivatives}} f(t_0 + \Delta t)$$

$$\frac{f(t_0 + \Delta t) - f(t_0)}{\Delta t} \approx \frac{\mathrm{d}f}{\mathrm{d}t}(t_0)$$

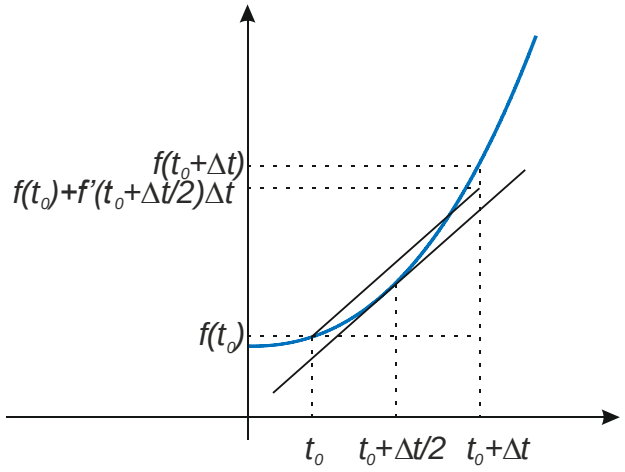
$$f(t_0 + \Delta t) \approx f(t_0) + \frac{\mathrm{d}f}{\mathrm{d}t}(t_0)\Delta t$$



forward difference

$$\frac{f(t_0 + \Delta t) - f(t_0)}{\Delta t} \approx \frac{\mathrm{d}f}{\mathrm{d}t}\left(t_0 + \frac{\Delta t}{2}\right)$$

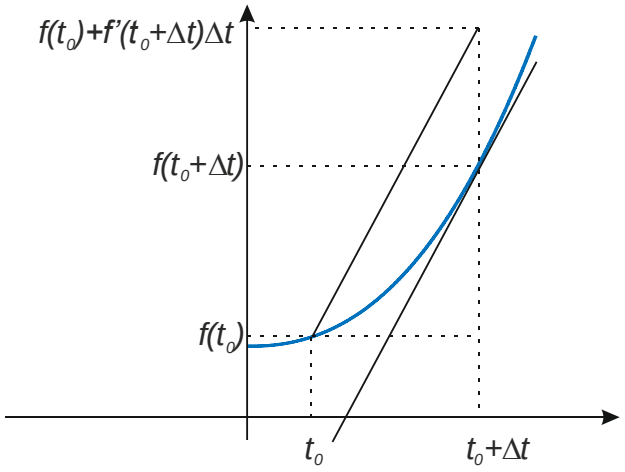
$$f(t_0 + \Delta t) \approx f(t_0) + \frac{\mathrm{d}f}{\mathrm{d}t}\left(t_0 + \frac{\Delta t}{2}\right)\Delta t$$



central difference

$$\frac{f(t_0) - f(t_0 - \Delta t)}{\Delta t} \approx \frac{\mathrm{d}f}{\mathrm{d}t}(t_0 + \Delta t)$$

$$f(t_0) \approx f(t_0 - \Delta t) + \frac{\mathrm{d}f}{\mathrm{d}t}(t_0 + \Delta t)\Delta t$$



backward difference

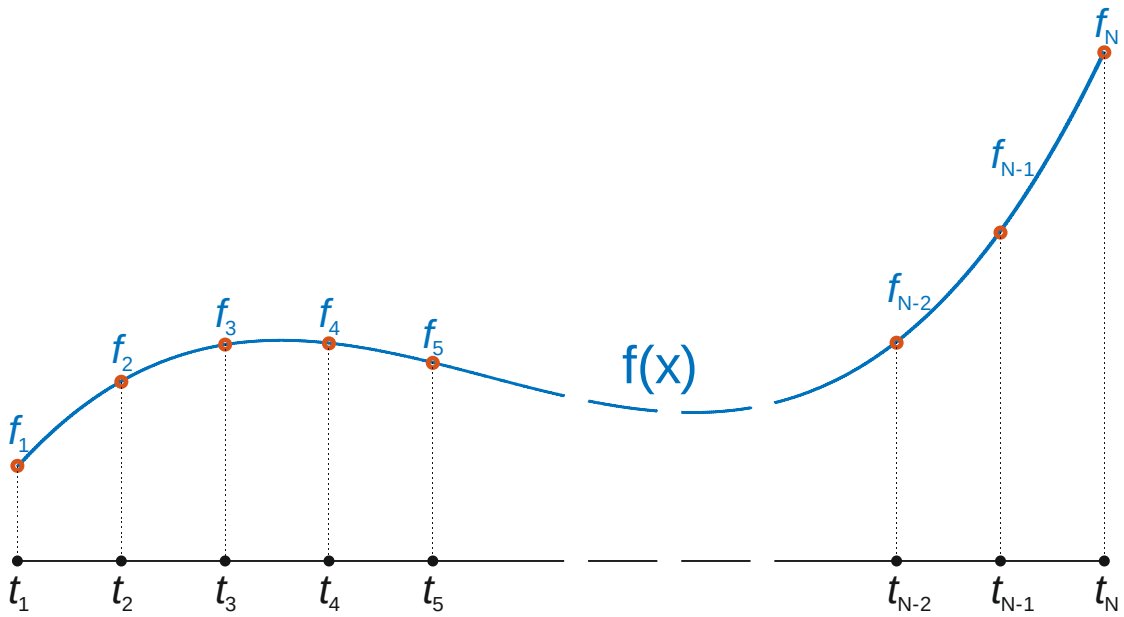
Introduction to finite difference methods

At first, the domain must be discretized with the use of a fixed step Δt

$$t_n = t_1 + (n-1)\Delta t, \qquad n = 1, \dots, N,$$

where $\Delta t = (t_N - t_1)/(N-1)$. Then the function will be also discretized

$$f_n = f(t_n), \qquad n = 1, \dots, N.$$



$$\boxed{\frac{\mathrm{d}f}{\mathrm{d}t}(t_n) \approx \frac{f_{n+1} - f_{n-1}}{2\Delta t}}$$

$$\boxed{\frac{\mathrm{d}^2 f}{\mathrm{d}t^2}(t_n) \approx \frac{\frac{\mathrm{d}f}{\mathrm{d}t}\left(t_{n+\frac{1}{2}}\right) - \frac{\mathrm{d}f}{\mathrm{d}t}\left(t_{n-\frac{1}{2}}\right)}{\Delta t} \approx \frac{\frac{f_{n+1} - f_n}{\Delta t} - \frac{f_n - f_{n-1}}{\Delta t}}{\Delta t} \approx \frac{f_{n+1} - 2f_n + f_{n-1}}{\Delta t^2}}$$

Taylor series – approximations

Taylor series is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point

$$f(x_0+\Delta x)=\sum_{n=0}^{\infty} f^{(n)}(x_0)\frac{\Delta x^n}{n!}=f(x_0)+f'(x_0)\Delta x+f''(x_0)\frac{\Delta x^2}{2}+f'''(x_0)\frac{\Delta x^3}{6}+f''''(x_0)\frac{\Delta x^4}{24}+\cdots$$

Let us consider the function $f(x)=\sin(x)$ and expansion point $x_0=\frac{\pi}{3}$

$f(x)=\sin(x)\ \ \rightarrow f(x_0)=\sin(x_0)=\frac{\sqrt{3}}{2}$

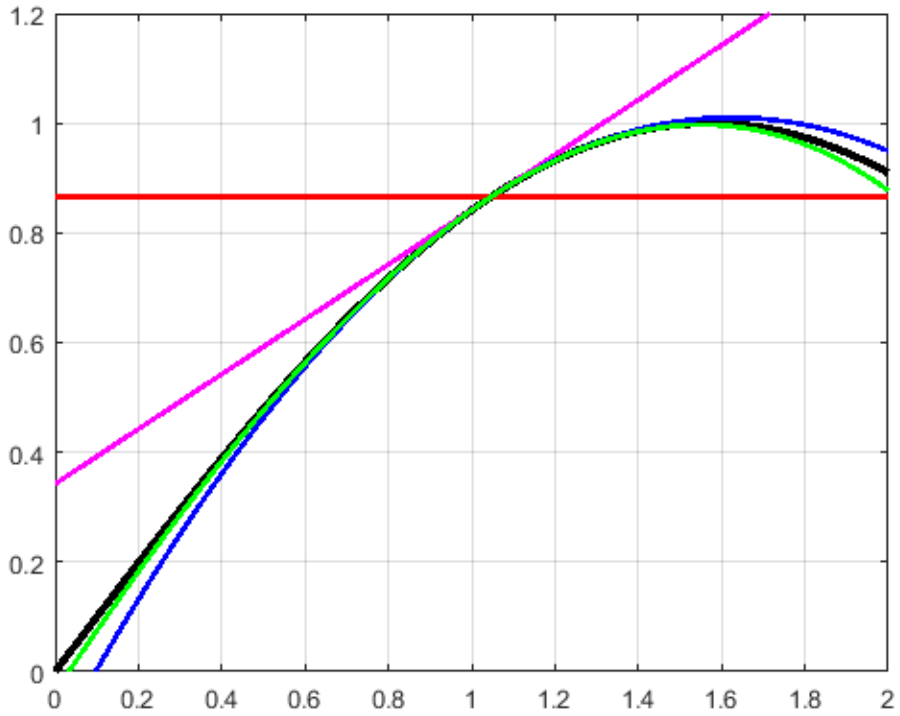
$f'(x)=\cos(x)\ \ \rightarrow f'(x_0)=\cos(x_0)=\frac{1}{2}$

$f''^{(x)}=-\sin(x)\ \ \rightarrow f''^{(x_0)}=-\sin(x_0)=-\frac{\sqrt{3}}{2}$

$f'''^{(x)}=-\cos(x)\ \ \rightarrow f'''^{(x_0)}=-\cos(x_0)=-\frac{1}{2}$

Then, in close vicinity of $x_0=\frac{\pi}{3}$ the function can be approximated by

$$f(x_0+\Delta x)\approx\boxed{\boxed{\boxed{\frac{\sqrt{3}}{2}}+\frac{1}{2}\Delta x}-\frac{\sqrt{3}}{4}\Delta x^2}-\frac{1}{12}\Delta x^3$$



Introduction to finite difference methods

The both formulas can be confirmed with Taylor series

$$f(t_0 + \Delta t) \approx f(t_0) + f'(t_0)\Delta t$$

$$f(t_0 - \Delta t) \approx f(t_0) - f'(t_0)\Delta t$$

$$f(t_0 + \Delta t) - f(t_0 - \Delta t) \approx 2f'(t_0)\Delta t$$

$$f'(t_0) \approx \frac{f(t_0 + \Delta t) - f(t_0 - \Delta t)}{2\Delta t}$$

$$f(t_0 + \Delta t) \approx f(t_0) + f'(t_0)\Delta t + f''(t_0)\frac{\Delta t^2}{2}$$

$$f(t_0 - \Delta t) \approx f(t_0) - f'(t_0)\Delta t + f''(t_0)\frac{\Delta t^2}{2}$$

$$f(t_0 + \Delta t) + f(t_0 - \Delta t) \approx 2f(t_0) + f''(t_0)\Delta t^2$$

$$f''(t_0) \approx \frac{f(t_0 + \Delta t) - 2f(t_0) + f(t_0 - \Delta t)}{\Delta t^2}$$

$$\boxed{\frac{\mathrm{d}f}{\mathrm{d}t}(t_n) \approx \frac{f_{n+1} - f_{n-1}}{2\Delta t}}$$

$$\boxed{\frac{\mathrm{d}^2 f}{\mathrm{d}t^2}(t_n) \approx \frac{\frac{\mathrm{d}f}{\mathrm{d}t}\left(t_{n+\frac{1}{2}}\right) - \frac{\mathrm{d}f}{\mathrm{d}t}\left(t_{n-\frac{1}{2}}\right)}{\Delta t} \approx \frac{\frac{f_{n+1} - f_n}{\Delta t} - \frac{f_n - f_{n-1}}{\Delta t}}{\Delta t} \approx \frac{f_{n+1} - 2f_n + f_{n-1}}{\Delta t^2}}$$

Gravity pendulum – finite difference method

Problem 5.2: Use the finite difference method to simulate a simple damped pendulum of length $R = 1$ m ended with a point mass $m = 1$ kg. Assume, that the frictional force can be modeled as being proportional to the velocity of the object $F_f = -bv$, where $b = 0.1$ Ns/m is called the viscous damping coefficient. Consider the amplitudes beyond the small angle approximation.

Solution: Appling the finite difference formulas to equation

$$m\frac{d^2x}{dt^2}(t) = -b\frac{dx}{dt}(t) - mg\sin\left(\frac{x(t)}{R}\right)$$

one gets (for $n = 2,3,4, \ldots$)

$$m\frac{x_{n+1} - 2x_n + x_{n-1}}{\Delta t^2} = -b\frac{x_{n+1} - x_{n-1}}{2\Delta t} - mg\sin\left(\frac{x_n}{R}\right)$$

or, in the more useful form

$$x_{n+1} = \frac{\frac{2m}{\Delta t^2}x_n + \left(\frac{b}{2\Delta t} - \frac{m}{\Delta t^2}\right)x_{n-1} - mg\sin\left(\frac{x_n}{R}\right)}{\left(\frac{m}{\Delta t^2} + \frac{b}{2\Delta t}\right)}$$

The initial conditions must be used to determine x_1 and x_2

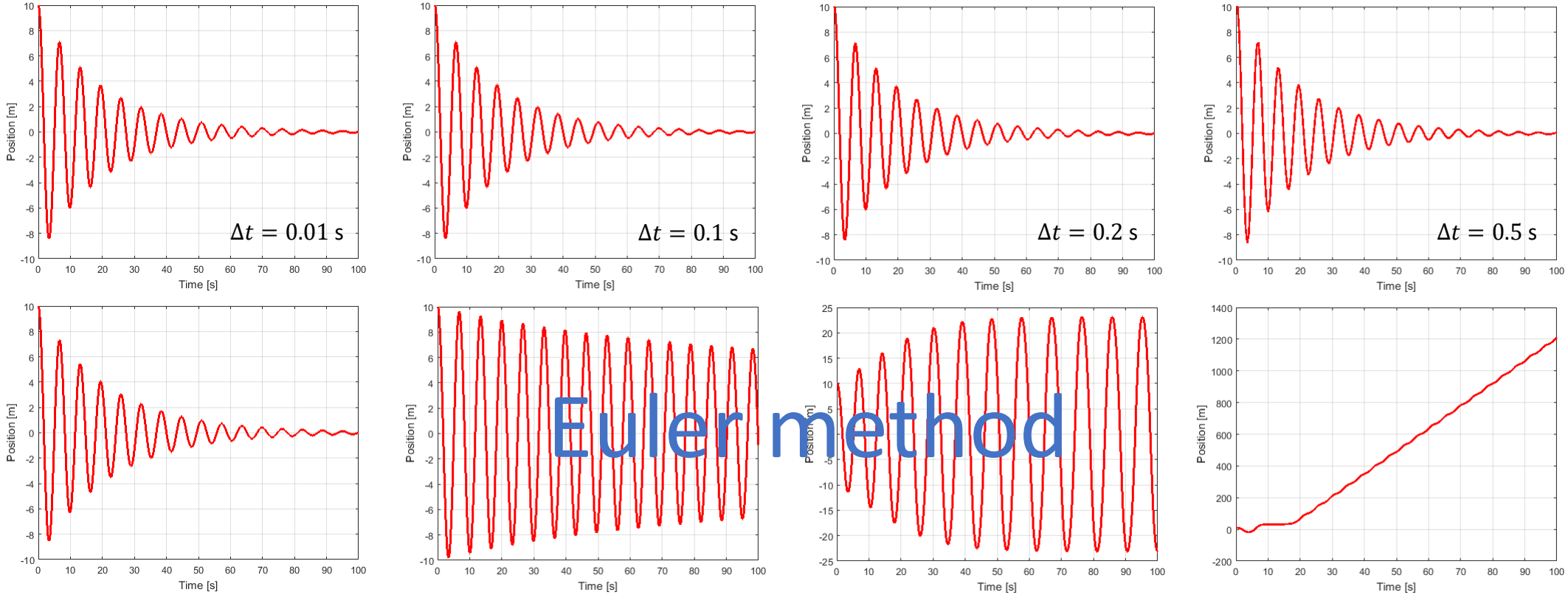
$$x_1 = x(t_1) \qquad x_2 = x_1 + v(t_1)\Delta t$$

```
MATLAB CODE EXAMPLE:

b=0.1; g=10; m=1; R=10; v0=0; xn=10;
dt=0.1;
T=30;
x=[xn xn+v0*dt];
N=round(T/dt);
A=(m/dt^2+b/dt/2)^(-1);
B=A*2*m/dt^2;
C=A*b/2/dt-m/dt^2;
D=A*m*g;
for n=2:N
    x(n+1)=B*x(n)+C*x(n-1)-D*sin(x(n)/R);
end
```

Gravity pendulum – finite difference method

Problem 5.2: Use the finite difference method to simulate a simple damped pendulum of length $R = 1$ m ended with a point mass $m = 1$ kg. Assume, that the frictional force can be modeled as being proportional to the velocity of the object $F_f = -bv$, where $b = 0.1$ Ns/m is called the viscous damping coefficient. Consider the amplitudes beyond the small angle approximation.

Solution: The results for: the initial position $x_0 = 10$ m, the initial velocity $v_0 = 0$ m/s



Wave equation

The wave equation is one of the most important **partial differential equations**. It is the fundamental equation of:

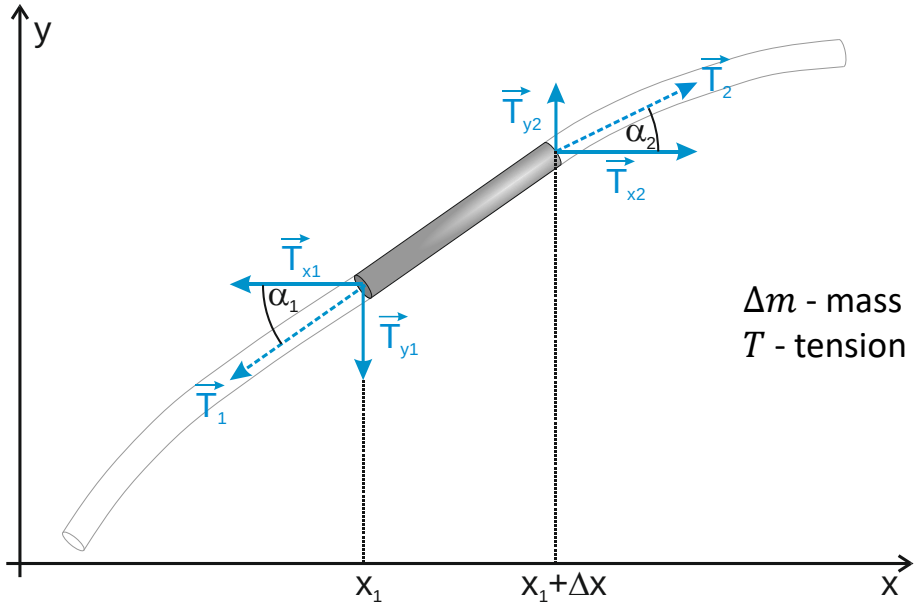
- acoustics (sound waves)
- electrodynamics (electromagnetic waves)
- photonics (light waves)
- fluid dynamics (water waves)
- quantum mechanics (wave functions)
- gravity?
- …

In 1746, d'Alembert formulated the one-dimensional wave equation, and within ten years Euler generalized it to the three-dimensional wave equation.

Jean-Baptiste le Rond d'Alembert (1717 – 1783) was a French mathematician, physicist and music theorist.

Source: https://en.wikipedia.org/wiki/Jean_le_Rond_d%27Alembert

Wave equation - derivation

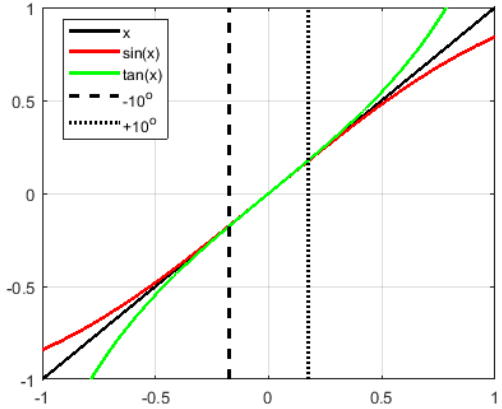


Δm - mass
T - tension

$$T_1 \approx T_2 \approx T$$

$$T_y(x,t) \approx T \sin\alpha_1 \approx T \tan\alpha_1 \approx T\frac{\partial y}{\partial x}(x,t)$$

$$T_y(x+\Delta x,t) \approx T \sin\alpha_2 \approx T \tan\alpha_2 \approx T\frac{\partial y}{\partial x}(x+\Delta x,t)$$



According to the Newton's second law

$$\Delta m \frac{\partial^2 y}{\partial t^2}(x,t) = T_y(x+\Delta x,t) - T_y(x,t)$$

Then, for small angles

$$\Delta m \frac{\partial^2 y}{\partial t^2}(x,t) = T\frac{\partial y}{\partial x}(x+\Delta x,t) - T\frac{\partial y}{\partial x}(x,t)$$

Next, dividing the both sides by Δx

$$\frac{\Delta m}{\Delta x}\frac{\partial^2 y}{\partial t^2}(x,t) = \frac{T\frac{\partial y}{\partial x}(x+\Delta x,t) - T\frac{\partial y}{\partial x}(x,t)}{\Delta x} = T\frac{\partial^2 y}{\partial x^2}(x,t)$$

Wave equation – derivation

Finally, introducing linear mass density $\mu = \frac{\Delta m}{\Delta x}$ [kg/m]

$$\mu \frac{\partial^2 y}{\partial t^2}(x,t) = T \frac{\partial^2 y}{\partial x^2}(x,t)$$

or simply

$$\frac{\partial^2 y}{\partial t^2}(x,t) = v^2 \frac{\partial^2 y}{\partial x^2}(x,t)$$

where $v = \sqrt{\frac{T}{\mu}}$ [m/s] represents velocity of the wave.

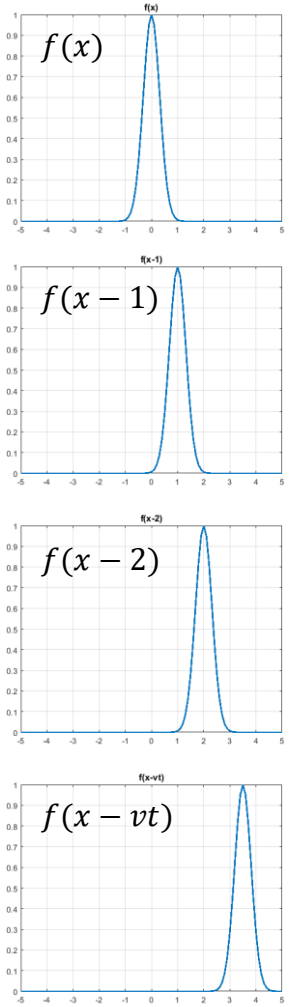
Any function $y(x,t) = f(x - vt)$ is a solution of the wave equation!

Proof:

Since $\frac{\partial y}{\partial t}(x,t) = -vf'(x - vt)$, $\frac{\partial^2 y}{\partial t^2}(x,t) = v^2 f''(x - vt)$,

$\frac{\partial y}{\partial x}(x,t) = f'(x - vt)$ and $\frac{\partial^2 y}{\partial x^2}(x,t) = f''(x - vt)$,

then the equation is fulfilled for any function of argument $x - vt$



Wave equation – finite difference method

Problem 5.3: Use the finite difference method to simulate a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m.

Solution: The finite difference formula can be applied for time

$$\frac{\partial^2 y}{\partial t^2}(x,t) \approx \frac{y(x,t+\Delta t) - 2y(x,t) + y(x,t-\Delta t)}{\Delta t^2}$$

as well as for space

$$\frac{\partial^2 y}{\partial x^2}(x,t) \approx \frac{y(x+\Delta x,t) - 2y(x,t) + y(x-\Delta x,t)}{\Delta x^2}$$

But first, the computational domain must be discretized. Let us assume, that

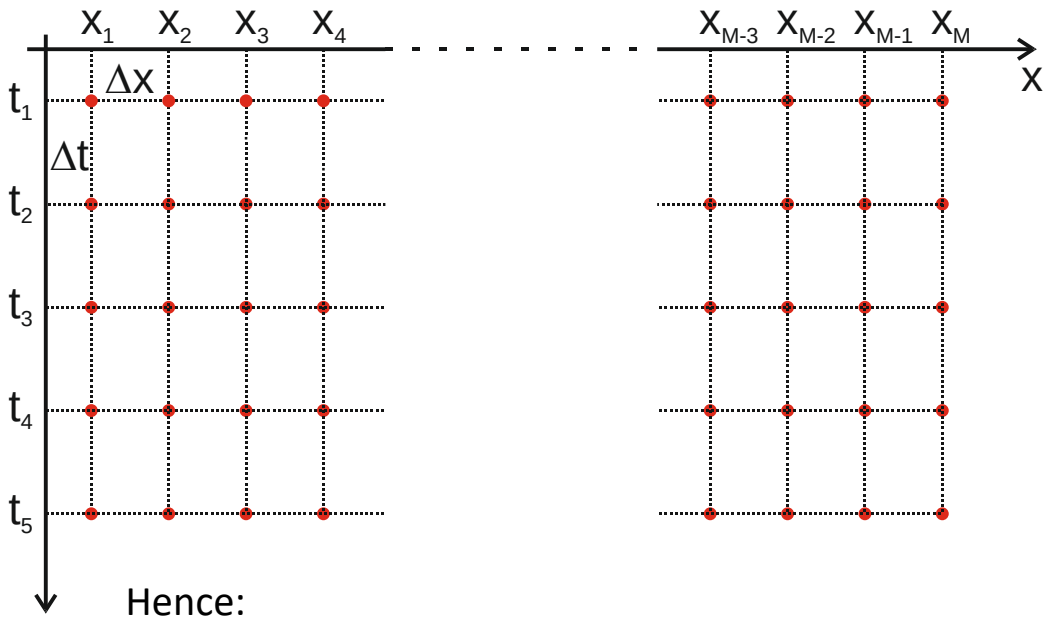
$$y_{m,n} = y(x_m, t_n)$$

where

$$x_m = x_1 + (m-1)\Delta x$$

and

$$t_n = t_1 + (n-1)\Delta t$$



Hence:

$$\frac{\partial^2 y}{\partial t^2}(x_m, t_n) \approx \frac{y_{m,n+1} - 2y_{m,n} + y_{m,n-1}}{\Delta t^2}$$

and

$$\frac{\partial^2 y}{\partial x^2}(x_m, t_n) \approx \frac{y_{m+1,n} - 2y_{m,n} + y_{m-1,n}}{\Delta x^2}$$

Wave equation – finite difference method

Problem 5.3: Use the finite difference method to simulate a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m.

Solution: Using these simple formulas

$$\frac{\partial^2 y}{\partial t^2}(x_m, t_n) \approx \frac{y_{m,n+1} - 2y_{m,n} + y_{m,n-1}}{\Delta t^2}$$

and

$$\frac{\partial^2 y}{\partial x^2}(x_m, t_n) \approx \frac{y_{m+1,n} - 2y_{m,n} + y_{m-1,n}}{\Delta x^2}$$

the wave equation

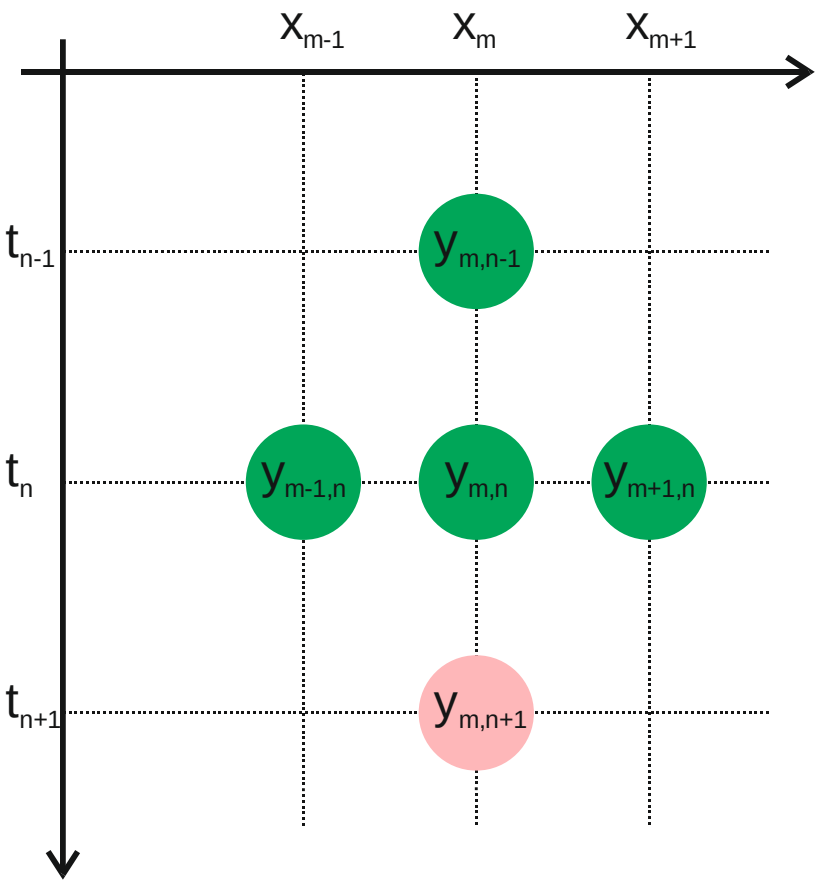
$$\frac{\partial^2 y}{\partial t^2}(x, t) = v^2 \frac{\partial^2 y}{\partial x^2}(x, t)$$

transforms to algebraic form

$$\frac{y_{m,n+1} - 2y_{m,n} + y_{m,n-1}}{\Delta t^2} = v^2 \frac{y_{m+1,n} - 2y_{m,n} + y_{m-1,n}}{\Delta x^2}$$

or

$$y_{m,n+1} = \left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{m+1,n} - 2y_{m,n} + y_{m-1,n}\right) + 2y_{m,n} - y_{m,n-1}$$



Wave equation – finite difference method

Problem 5.3: Use the finite difference method to simulate a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m.

Solution: From the initial conditions we get first „two rows". The initial shape of the string determines $y_{m,1}$

$$y(x, t_1) = s(x) \;\rightarrow\; y_{m,1} = s(x_m) = s_m$$

However, the initial velocity does not determines $y_{m,2}$ directly. The velocity provides the relation

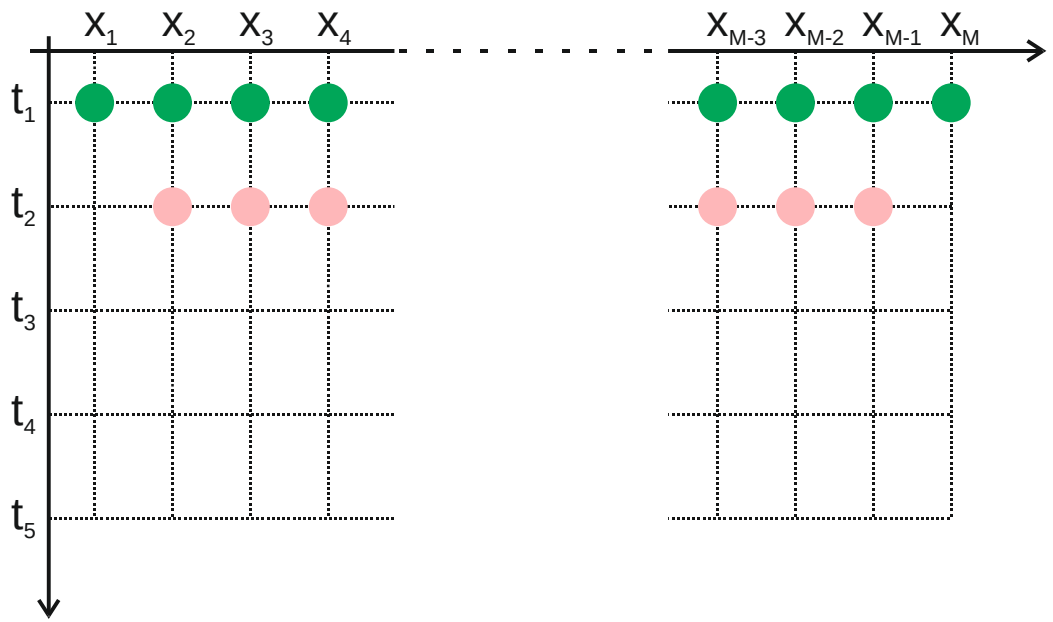
$$\frac{\partial y}{\partial \mathrm{t}}(x, t_1) = g(x) \rightarrow \frac{y_{m,2} - y_{m,0}}{2\Delta t} = g(x_m) = g_m$$

which must be combined with the general formula for $n = 1$

$$y_{m,2} = \left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{m+1,1} - 2y_{m,1} + y_{m-1,1}\right) + 2y_{m,1} - y_{m,0}$$

Finally, we get

$$y_{m,2} = \frac{1}{2}\left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{m+1,1} - 2y_{m,1} + y_{m-1,1}\right) + y_{m,1} + \Delta t g_m$$



Wave equation – finite difference method

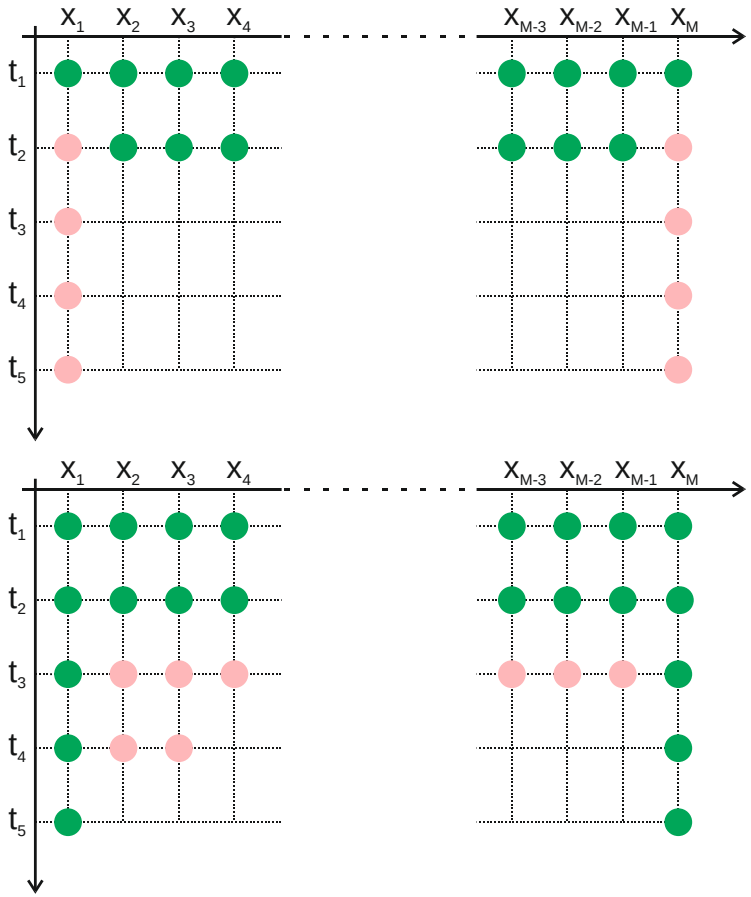
Problem 5.3: Use the finite difference method to simulate a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m.

Solution: If the value of the function is known at the boundary (Dirichlet boundary conditions), then

$$y(x_1, t\,) = l(t) \;\rightarrow\; y_{1,n} = l(t_n) = l_n$$

and

$$y(x_M, t\,) = r(t) \;\rightarrow\; y_{M,n} = r(t_n) = r_n$$



Finally, the general formula can be applied

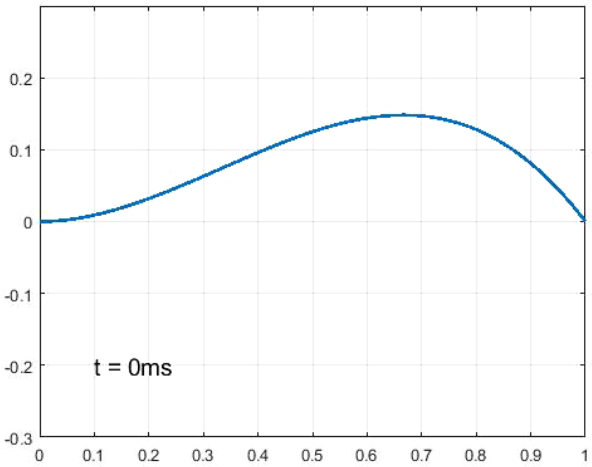
$$y_{m,n+1} = \left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{m+1,n} - 2y_{m,n} + y_{m-1,n}\right) + 2y_{m,n} - y_{m,n-1}$$

Wave equation – finite difference method

Problem 5.3: Use the finite difference method to simulate a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m.

Solution: The implementation is much easier than the theory. To obtain a stable solution the time step must be sufficiently small $\Delta t \leq \frac{\Delta x}{v}$ (Courant–Friedrichs–Lewy condition)! The assumed initial conditions are $\mathrm{y}(x,0) = x^2(L-x)$ and $\frac{\partial y}{\partial \mathrm{t}}(x,0) = 0$. The boundary conditions are fixed to zero.

```
MATLAB CODE EXAMPLE:

mu=0.01; tau=100; L=1; M=100; N=1000; x=linspace(0,L,M)';
v=sqrt(tau/mu); dx=x(2)-x(1); dt=dx/v; p=(v*dt/dx)^2;
s=x.^2.*(L-x); g=zeros(M,1); l=zeros(1,N); r=zeros(1,N);

f(:,1)=s;
f(2:M-1,2)=p/2*(f(3:M,1)-2*f(2:M-1,1)+f(1:M-2,1))+f(2:M-1,1)+dt*g(2:M-1);
f([1 M],2)=[l(2) r(2)];
for n=2:N-1
   f(2:M-1,n+1)=p*(f(3:M,n)-2*f(2:M-1,n)+f(1:M-2,n))+2*f(2:M-1,n)-f(2:M-1,n-1);
   f([1 M],n+1)=[l(n+1) r(n+1)];
end
```

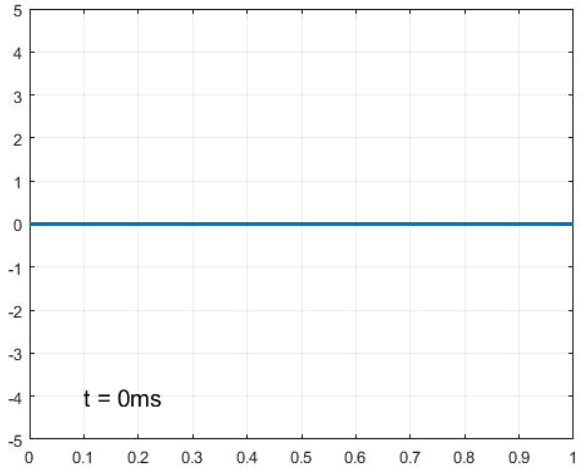


Wave equation – finite difference method

Problem 5.3: Use the finite difference method to simulate a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m.

Solution: Let us consider the situation without initial displacement $y(x, 0) = 0$ and without initial velocity $\frac{\partial y}{\partial t}(x, 0) = 0$, but with time dependent boundary conditions $l(0, t) = \sin(1000t)$ and $r(0, t) = 0$.

```
MATLAB CODE EXAMPLE:

mu=0.01; tau=100; L=1; M=100; N=1000; x=linspace(0,L,M)';
v=sqrt(tau/mu); dx=x(2)-x(1); dt=dx/v; p=(v*dt/dx)^2;
s=zeros(M,1); g=zeros(M,1); l=sin((0:N)'*dt*1000); r=zeros(1,N);

f(:,1)=s;
f(2:M-1,2)=p/2*(f(3:M,1)-2*f(2:M-1,1)+f(1:M-2,1))+f(2:M-1,1)+dt*g(2:M-1);
f([1 M],2)=[l(2) r(2)];
for n=2:N-1
   f(2:M-1,n+1)=p*(f(3:M,n)-2*f(2:M-1,n)+f(1:M-2,n))+2*f(2:M-1,n)-f(2:M-1,n-1);
   f([1 M],n+1)=[l(n+1) r(n+1)];
end
```

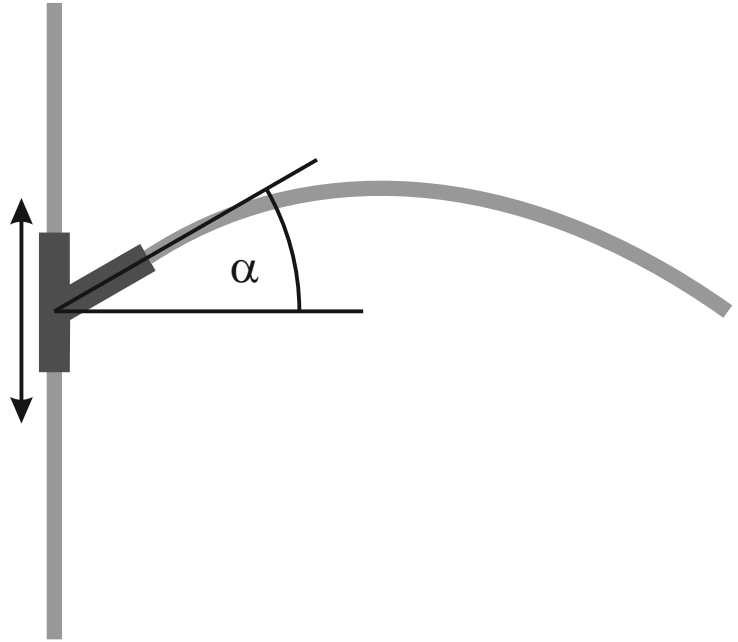


Wave equation – finite difference method

Problem 5.4: Use the finite difference method to simulate a string of length $L = 1$ m. Assume, that the string is fixed at the one end and the angle of the string is fixed at the other end. The initial conditions are known, the tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m.

Solution: If the angle of the string at the end is fixed, it is equivalent with to the fixed value of spatial derivative at this end (Neumann boundary conditions). Hence, the condition can be expressed by

$$\frac{\partial y}{\partial x}(0, t\ ) = h(t)$$



The derivative represents the tangent of the fixed angle (which in general can be time dependent)

$$\frac{\partial y}{\partial x}(0, t\ ) = \tan\big(\alpha(t)\big) = h(t)$$

Wave equation – finite difference method

Problem 5.4: Use the finite difference method to simulate a string of length $L = 1$ m. Assume, that the string is fixed at the one end and the angle of the string is fixed at the other end. The initial conditions are known, the tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m.

Solution: In the discrete version this boundary condition can be written in the following form

$$\frac{y_{2,n} - y_{0,n}}{2\Delta x} = h_n$$

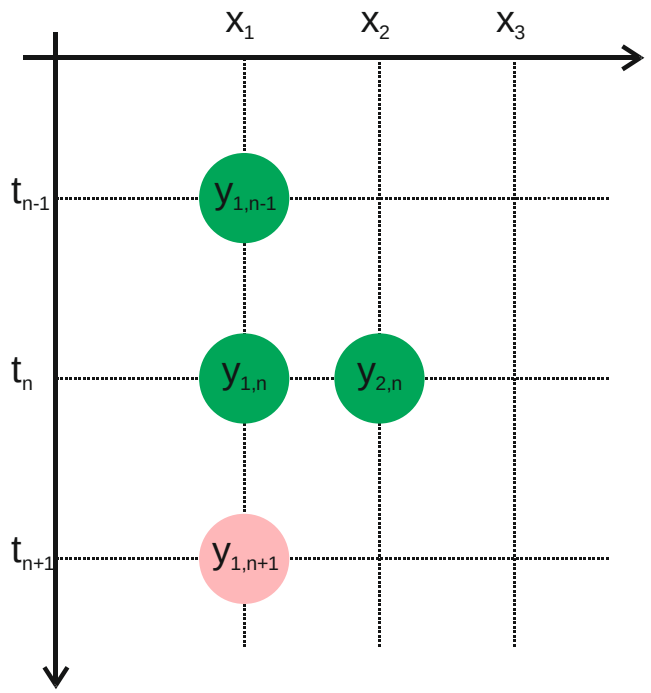
Combining it with the general formula

$$y_{1,n+1} = \left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{2,n} - 2y_{1,n} + y_{0,n}\right) + 2y_{1,n} - y_{1,n-1}$$

we get an iterative expression for the boundary values

$$y_{1,n+1} = 2\left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{2,n} - y_{1,n} - \Delta x h_n\right) + 2y_{1,n} - y_{1,n-1}$$

However, since it requires $y_{1,n-1}$ term, then this formula can be applied for $n \geq 2$ only.



Wave equation – finite difference method

Problem 5.4: Use the finite difference method to simulate a string of length $L = 1$ m. Assume, that the string is fixed at the one end and the angle of the string is fixed at the other end. The initial conditions are known, the tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m.

Solution: For $n = 1$ the relation has the following form

$$y_{1,2} = 2\left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{2,1} - y_{1,1} - \Delta x h_1\right) + 2y_{1,1} - y_{1,0}$$

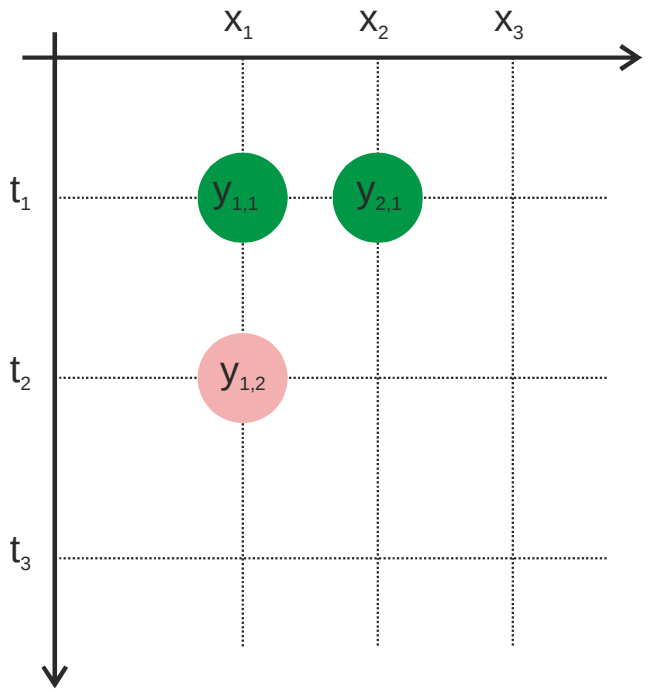
and requires an extra term $y_{1,0}$.

This term can be found from initial condition

$$\frac{y_{1,2} - y_{1,0}}{2\Delta t} = g_1$$

So, finally

$$y_{1,2} = \left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{2,1} - y_{1,1} - \Delta x h_1\right) + y_{1,1} + \Delta t g_1$$

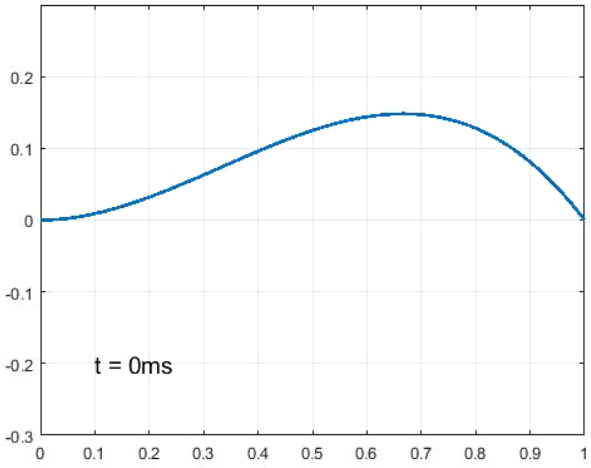


Wave equation – finite difference method

Problem 5.4: Use the finite difference method to simulate a string of length $L = 1$ m. Assume, that the string is fixed at the one end and the angle of the string is fixed at the other end. The initial conditions are known, the tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m.

Solution: The angle is fixed to zero at the left end $h(t) = 0$ and the rest of the implementation is exactly the same as in the previous case.

```
MATLAB CODE EXAMPLE:

mu=0.01; tau=100; L=1; M=100; N=1000; x=linspace(0,L,M)';
v=sqrt(tau/mu); dx=x(2)-x(1); dt=dx/v; p=(v*dt/dx)^2;
s=x.^2.*(L-x); g=zeros(M,1); h=zeros(1,N); r=zeros(1,N);

f(:,1)=s;
f(1,2)=p*(f(2,1)- f(1,1)-dx*h(1)) + f(1,1) +dt*g(1);
f(2:M-1,2)=p/2*(f(3:M,1)-2*f(2:M-1,1)+ f(1:M-2,1))+f(2:M-1,1)+dt*g(2:M-1);
f(M,2)=r(2);
for n=2:N-1
    f(1,n+1)=2*p*(f(2,n)-f(1,n)- dx*h(n) )+2*f(1,n)-f(1,n-1);
    f(2:M-1,n+1)=p*(f(3:M,n)-2*f(2:M-1,n)+f(1:M-2,n))+2*f(2:M-1,n)-f(2:M-1,n-1);
    f(M,n+1)=r(n+1);
end
```



Wave equation – finite difference method

Problem 5.5: Use the finite difference method to simulate a damped string of length $L = 1$ m. The string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m. Assume, that the frictional force acting on a string segment is proportional to the velocity of the string $\Delta F_f = -bv\Delta x$, with $b = 2$ Ns/m².

Solution: The simulation requires some modifications in the general wave equation

$$\Delta m \frac{\partial^2 y}{\partial t^2}(x,t) = T\frac{\partial y}{\partial x}(x+\Delta x,t) - T\frac{\partial y}{\partial x}(x,t) - b\Delta x\frac{\partial y}{\partial t}(x,t)$$

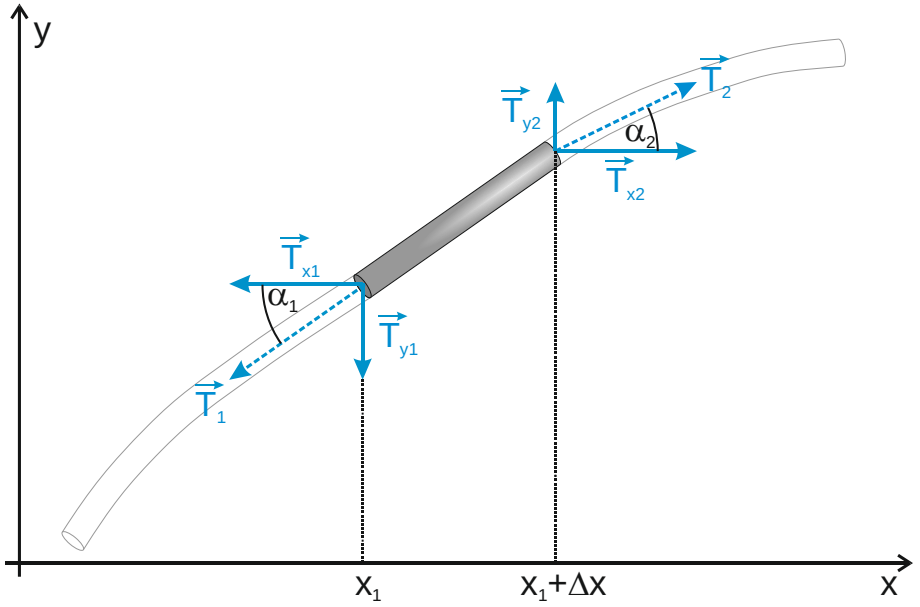
Next, dividing the both sides by Δx

$$\frac{\Delta m}{\Delta x}\frac{\partial^2 y}{\partial t^2}(x,t) = T\frac{\partial^2 y}{\partial x^2}(x,t) - b\frac{\partial y}{\partial t}(x,t)$$

or

$$\frac{\partial^2 y}{\partial t^2}(x,t) = v^2\frac{\partial^2 y}{\partial x^2}(x,t) - 2\beta\frac{\partial y}{\partial t}(x,t)$$

where $\beta = \frac{b}{2\mu}$.



Wave equation – finite difference method

Problem 5.5: Use the finite difference method to simulate a damped string of length $L = 1$ m. The string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m. Assume, that the frictional force acting on a string segment is proportional to the velocity of the string $\Delta F_f = -bv\Delta x$, with $b = 2$ Ns/m^2.

Solution: Using the standard rule of finite difference method, the equation

$$\frac{\partial^2 y}{\partial t^2}(x,t) = v^2 \frac{\partial^2 y}{\partial x^2}(x,t) - 2\beta \frac{\partial y}{\partial t}(x,t)$$

can be expressed by

$$\frac{y_{m,n+1} - 2y_{m,n} + y_{m,n-1}}{\Delta t^2} = v^2 \frac{y_{m+1,n} - 2y_{m,n} + y_{m-1,n}}{\Delta x^2} - \beta \frac{y_{m,n+1} - y_{m,n-1}}{\Delta t}$$

In such a case the general finite difference formula is

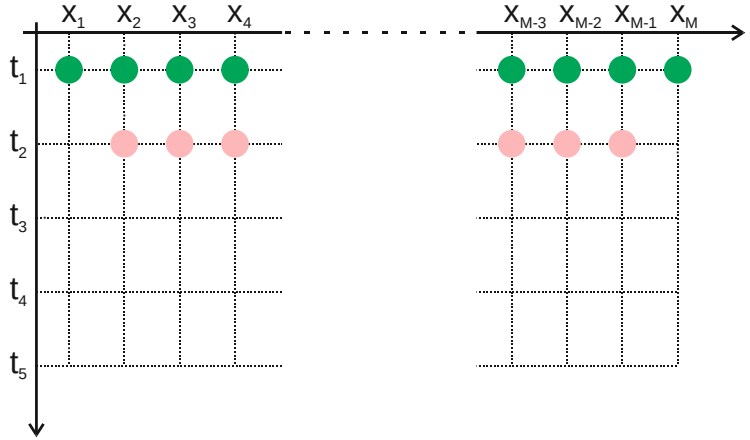
$$y_{m,n+1} = \frac{1}{1+\Delta t\beta}\left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{m+1,n} - 2y_{m,n} + y_{m-1,n}\right) + \frac{2}{1+\Delta t\beta} y_{m,n} - \frac{1-\Delta t\beta}{1+\Delta t\beta} y_{m,n-1}$$

Wave equation – finite difference method

Problem 5.5: Use the finite difference method to simulate a damped string of length $L = 1$ m. The string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m. Assume, that the frictional force acting on a string segment is proportional to the velocity of the string $\Delta F_f = -bv\Delta x$, with $b = 2$ Ns/m.

Solution: From the first initial condition $y(x, t_1) = s(x)$ we get

$$y_{m,1} = s(x_m) = s_m$$



The initial velocity $\frac{\partial y}{\partial t}(x, t_1) = g(x)$ provides the relation

$$\frac{y_{m,2} - y_{m,0}}{2\Delta t} = g(x_m) = g_m$$

which must be combined with the general formula for $n = 1$

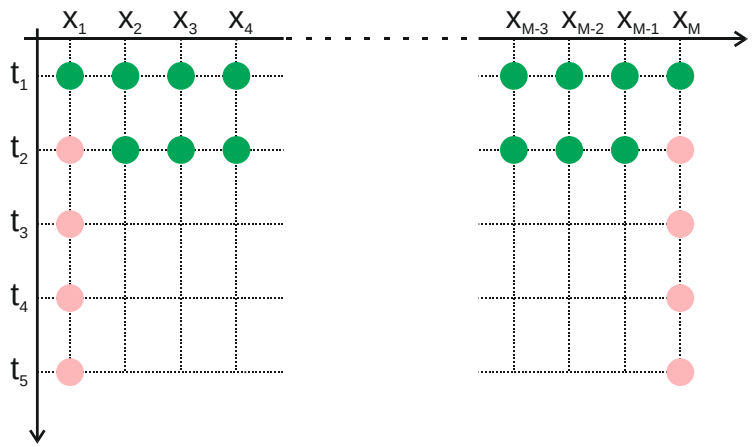
$$y_{m,2} = \frac{1}{1 + \Delta t\beta}\left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{m+1,1} - 2y_{m,1} + y_{m-1,1}\right) + \frac{2}{1 + \Delta t\beta} y_{m,1} - \frac{1 - \Delta t\beta}{1 + \Delta t\beta} y_{m,0}$$

and finally

$$y_{m,2} = \frac{1}{2}\left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{m+1,1} - 2y_{m,1} + y_{m-1,1}\right) + y_{m,1} + (1 - \Delta t\beta)\Delta t g_m$$

Wave equation – finite difference method

Problem 5.5: Use the finite difference method to simulate a damped string of length $L = 1$ m. The string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m. Assume, that the frictional force acting on a string segment is proportional to the velocity of the string $\Delta F_f = -bv\Delta x$, with $b = 2$ Ns/m^2.



Solution: For Dirichlet boundary conditions the relations are simple (as in the case without damping)

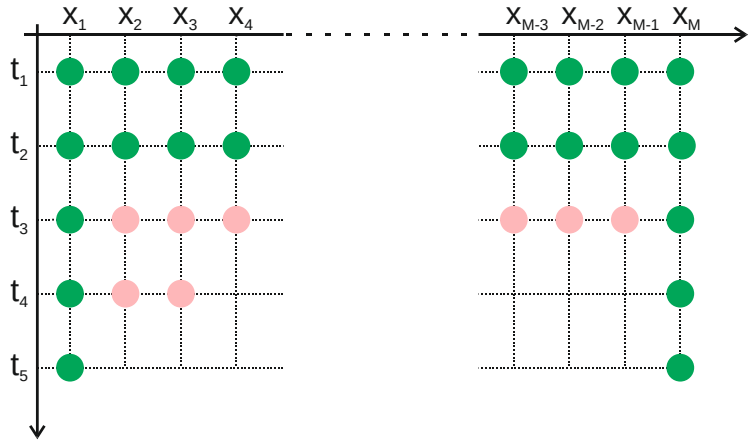
$$y(x_1, t) = l(t) \rightarrow y_{1,n} = l(t_n) = l_n$$

and

$$y(x_M, t) = r(t) \rightarrow y_{M,n} = r(t_n) = r_n$$

Eventually, the general formula can be applied

$$y_{m,n+1} = \frac{1}{1+\Delta t\beta}\left(\frac{v\Delta t}{\Delta x}\right)^2 (y_{m+1,n} - 2y_{m,n} + y_{m-1,n}) + \frac{2}{1+\Delta t\beta} y_{m,n} - \frac{1-\Delta t\beta}{1+\Delta t\beta} y_{m,n-1}$$

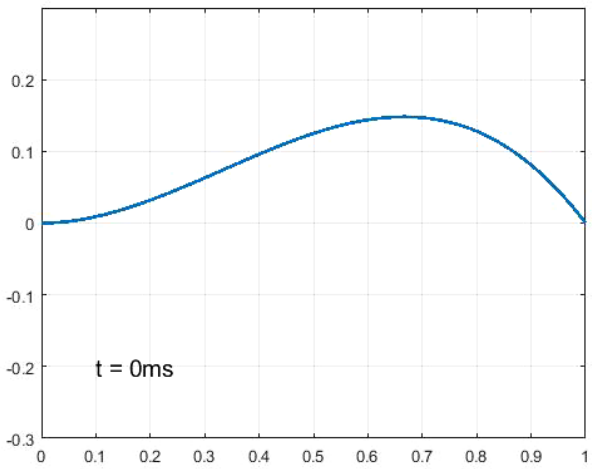


Wave equation – finite difference method

Problem 5.5: Use the finite difference method to simulate a damped string of length $L = 1$ m. The string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m. Assume, that the frictional force acting on a string segment is proportional to the velocity of the string $\Delta F_f = -bv\Delta x$, with $b = 2$ Ns/m^2.

Solution: The implementation is also similar to the previous case. The assumed initial conditions are $y(x,0) = x^2(L-x)$ and $\frac{\partial y}{\partial t}(x,0) = 0$. The boundary conditions are fixed to zero.

```
MATLAB CODE EXAMPLE:

mu=0.01; tau=100; b=2; beta=b/2/mu; L=1; M=100; N=500; x=linspace(0,L,M)';
v=sqrt(tau/mu); dx=x(2)-x(1); dt=dx/v; p=(v*dt/dx)^2; q=1+beta*dt;
u=1-beta*dt; s=x.^2.*(L-x); g=zeros(M,1); l=zeros(1,N); r=zeros(1,N);

f(:,1)=s;
f(2:M-1,2)=p/2*(f(3:M,1)-2*f(2:M-1,1)+ f(1:M-2,1))+f(2:M-1,1)+u*dt*g(2:M-1);
f([1 M],2)=[l(2) r(2)];
for n=2:N-1
   f(2:M-1,n+1)=p/q*(f(3:M,n)-2*f(2:M-1,n)+f(1:M-2,n))+2/q*f(2:M-1,n)
                -u/q*f(2:M-1,n-1);
   f([1 M],n+1)=[l(n+1) r(n+1)];
end
```



Wave equation – finite difference method

Problem 5.6: Use the finite difference method to simulate a damped string of length $L = 1$ m. The string is fixed at the one end and the angle of the string is fixed at the other end. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m. Assume, that the frictional force acting on a string segment is proportional to the velocity of the string $\Delta F_f = -bv\Delta x$, with $b = 2$ Ns/m^2.

Solution: The first part of the simulation (initial conditions) is exactly the same as in the problem 5.5. Hence, $y_{m,1} = s_m$ and

$$y_{m,2} = \frac{1}{2}\left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{m+1,1} - 2y_{m,1} + y_{m-1,1}\right) + y_{m,1} + (1 - \Delta t\beta)\Delta t g_m$$

Assuming Dirichlet boundary condition at the right end $y(x_M, t\,) = r(t)$, we get $y_{M,n} = r_n$.
Also the general finite difference relations is the same

$$y_{m,n+1} = \frac{1}{1 + \Delta t\beta}\left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{m+1,n} - 2y_{m,n} + y_{m-1,n}\right) + \frac{2}{1 + \Delta t\beta} y_{m,n} - \frac{1 - \Delta t\beta}{1 + \Delta t\beta} y_{m,n-1}$$

The problem boils down to finding the relations for $y_{1,n+1}$ and is similar to the considerations in problem 5.4

$$y_{1,2} = \left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{2,1} - y_{1,1} - \Delta x h_1\right) + y_{1,1} + (1 - \Delta t\beta)\Delta t g_1$$

and

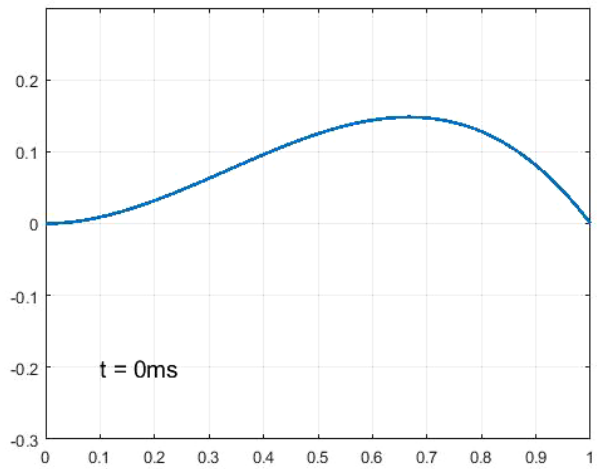
$$y_{1,n+1} = \frac{2}{1 + \Delta t\beta}\left(\frac{v\Delta t}{\Delta x}\right)^2 \left(y_{2,n} - y_{1,n} - \Delta x h_n\right) + \frac{2}{1 + \Delta t\beta} y_{1,n} - \frac{1 - \Delta t\beta}{1 + \Delta t\beta} y_{1,n-1}$$

Wave equation – finite difference method

Problem 5.6: Use the finite difference method to simulate a damped string of length $L = 1$ m. The string is fixed at the one end and the angle of the string is fixed at the other end. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m. Assume, that the frictional force acting on a string segment is proportional to the velocity of the string $\Delta F_f = -bv\Delta x$, with $b = 2$ Ns/m^2.

Solution: The implementation is also similar to the previous cases. The assumed initial conditions are $y(x,0) = x^2(L-x)$ and $\frac{\partial y}{\partial t}(x,0) = 0$. The function value is fixed to zero at the right end and the angle of the left end is also fixed to zero.

```
MATLAB CODE EXAMPLE:

mu=0.01; tau=100; b=2; beta=b/2/mu; L=1; M=100; N=500; x=linspace(0,L,M)';
v=sqrt(tau/mu); dx=x(2)-x(1); dt=dx/v; p=(v*dt/dx)^2; q=1+beta*dt;
u=1-beta*dt; s=x.^2.*(L-x); g=zeros(M,1); h=zeros(1,N); r=zeros(1,N);

f(:,1)=s;
f(1,2)=p*(f(2,1)- f(1,1)-dx*h(1)) + f(1,1) +u*dt*g(1);
f(2:M-1,2)=p/2*(f(3:M,1)-2*f(2:M-1,1)+ f(1:M-2,1))+f(2:M-1,1)+u*dt*g(2:M-1);
f(M,2)=r(2);
for n=2:N-1
   f(1,n+1)=2*p/q*(f(2,n)-f(1,n)- dx*h(n) )+2/q*f(1,n)-u/q*f(1,n-1);
   f(2:M-1,n+1)=p/q*(f(3:M,n)-2*f(2:M-1,n)+f(1:M-2,n))+2/q*f(2:M-1,n)
                 -u/q*f(2:M-1,n-1);
   f(M,n+1)=r(n+1);
end
```



Wave equation – equilibrium solution

Problem 5.7: Find the equilibrium solution (position) of a string of length $L = 1$ m. The string is fixed at the both ends and the initial conditions are known. The tension force $T = 100$ N and linear mass density $\mu = 0.01$ kg/m. Assume, the following boundary conditions $\frac{\partial y}{\partial x}(0, t\ ) = 1$ and $\mathrm{y}(L, t\ ) = 3$.

Solution: The equilibrium solution (position) of the wave equation

$$\frac{\partial^2 y}{\partial t^2}(x, t) = v^2 \frac{\partial^2 y}{\partial x^2}(x, t) - 2\beta \frac{\partial y}{\partial t}(x, t)$$

can be found by simple reduction of time dependences $\frac{\partial^2 y}{\partial t^2}(x, t) = 0$ and $\frac{\partial y}{\partial t}(x, t) = 0$.

Hence, the problem becomes simple

$$\frac{\partial^2 y_E}{\partial x^2}(x) = 0$$

and the general solution is

$$y_E(x) = cx + d$$

where c and d are arbitrary constants.

Taking into account the boundary conditions, we get $c = 1$ and $d = 2$.

Modeling and Simulation of Systems

Lecture 6: Finite difference method

Piotr Kowalczyk

Wave equation in two dimensions

The wave equation is one of the most important hyperbolic **partial differential equations**.

$$\frac{1}{v^2}\frac{\partial^2 f}{\partial t^2}(x,t) = \frac{\partial^2 f}{\partial x^2}(x,t)$$

where v represents a velocity of the wave.

It can be simply generalized to two dimensions

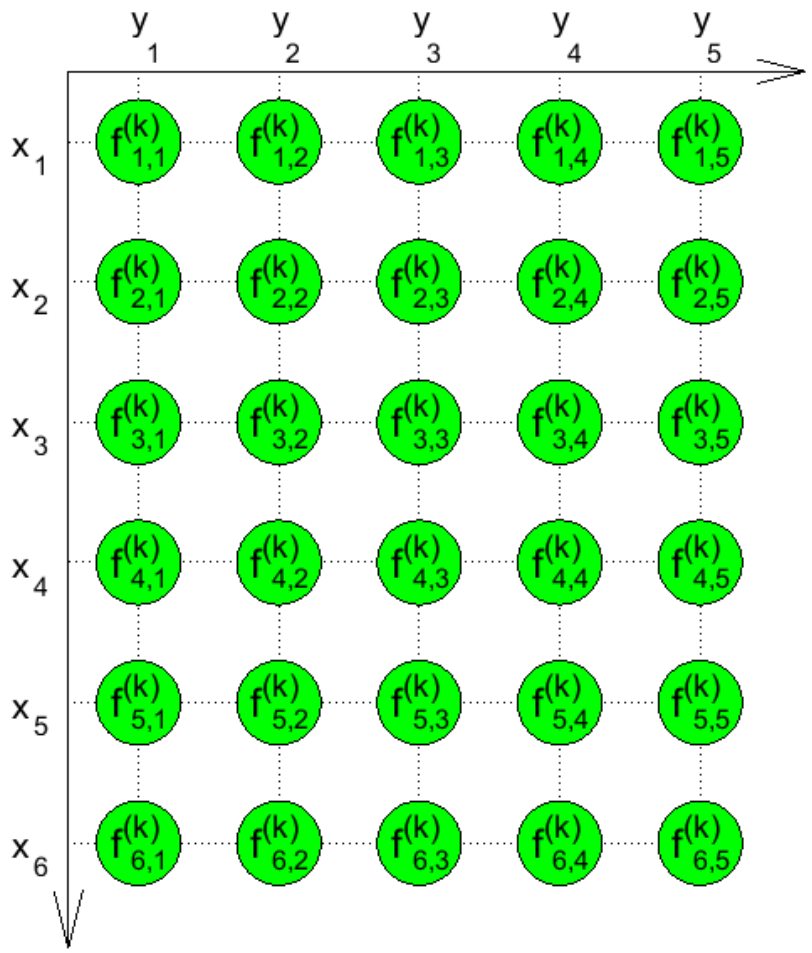
$$\frac{1}{v^2}\frac{\partial^2 f}{\partial t^2}(x,y,t) = \frac{\partial^2 f}{\partial x^2}(x,y,t) + \frac{\partial^2 f}{\partial y^2}(x,y,t)$$

It is the fundamental equation of:

- acoustics (sound waves)
- electrodynamics (electromagnetic waves)
- photonics (light waves)
- fluid dynamics (water waves)
- quantum mechanics (wave functions)
- gravity?
- …

Wave equation in two dimensions

Problem 6.1: Use finite difference method to simulate a vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane is known.



Solution: The main idea of finite difference method is a discretization of the computational domain with the use of regular rectangular mesh

$$f(x, y, t) \rightarrow f(x_m, y_n, t_k)$$

The function is defined on a discrete set of points

$$x_m = x_1 + (m-1)\Delta x$$
$$y_n = y_1 + (n-1)\Delta y$$

and in fixed time moments

$$t_k = t_1 + (k-1)\Delta t$$

Then

$$f(x_m, y_n, t_k) \rightarrow f_{m,n}^{(k)}$$

Wave equation in two dimensions

Problem 6.1: Use finite difference method to simulate a vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane is known.

Solution: In the discretized domain the wave equation

$$\frac{1}{v^2}\frac{\partial^2 f}{\partial t^2}(x,y,t) = \frac{\partial^2 f}{\partial x^2}(x,y,t) + \frac{\partial^2 f}{\partial y^2}(x,y,t)$$

can be modified with the use of the regular finite difference formulas

$$\frac{\partial^2 f}{\partial t^2}(x_m,y_n,t_k) \approx \frac{f_{m,n}^{(k+1)} - 2f_{m,n}^{(k)} + f_{m,n}^{(k-1)}}{\Delta t^2}$$

$$\frac{\partial^2 f}{\partial x^2}(x_m,y_n,t_k) \approx \frac{f_{m+1,n}^{(k)} - 2f_{m,n}^{(k)} + f_{m-1,n}^{(k)}}{\Delta x^2} \qquad \frac{\partial^2 f}{\partial y^2}(x_m,y_n,t_k) \approx \frac{f_{m,n+1}^{(k)} - 2f_{m,n}^{(k)} + f_{m,n-1}^{(k)}}{\Delta y^2}$$

Wave equation in two dimensions

Problem 6.1: Use finite difference method to simulate a vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane is known.

Solution: Hence, the partial differentia equation

$$\frac{1}{v^2}\frac{\partial^2 f}{\partial t^2}(x,y,t) = \frac{\partial^2 f}{\partial x^2}(x,y,t) + \frac{\partial^2 f}{\partial y^2}(x,y,t)$$

becomes a simple algebraic equation

$$\frac{1}{v^2}\frac{f_{m,n}^{(k+1)} - 2f_{m,n}^{(k)} + f_{m,n}^{(k-1)}}{\Delta t^2} = \frac{f_{m+1,n}^{(k)} - 2f_{m,n}^{(k)} + f_{m-1,n}^{(k)}}{\Delta x^2} + \frac{f_{m,n+1}^{(k)} - 2f_{m,n}^{(k)} + f_{m,n-1}^{(k)}}{\Delta y^2}$$

or in the more useful form

$$f_{m,n}^{(k+1)} = 2\left(1 - \left(\frac{v\Delta t}{\Delta x}\right)^2 - \left(\frac{v\Delta t}{\Delta y}\right)^2\right) f_{m,n}^{(k)} + \left(\frac{v\Delta t}{\Delta x}\right)^2 \left(f_{m+1,n}^{(k)} + f_{m-1,n}^{(k)}\right) + \left(\frac{v\Delta t}{\Delta y}\right)^2 \left(f_{m,n+1}^{(k)} + f_{m,n-1}^{(k)}\right) - f_{m,n}^{(k-1)}$$

Wave equation in two dimensions

Problem 6.1: Use finite difference method to simulate a vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane is known.

Solution: The obtained formula can be used only for $k \geq 2$, $1 < m < M$ and $1 < n < N$

$$f_{m,n}^{(k+1)} = 2\left(1 - \left(\frac{v\Delta t}{\Delta x}\right)^2 - \left(\frac{v\Delta t}{\Delta y}\right)^2\right) f_{m,n}^{(k)} + \left(\frac{v\Delta t}{\Delta x}\right)^2 \left(f_{m+1,n}^{(k)} + f_{m-1,n}^{(k)}\right) + \left(\frac{v\Delta t}{\Delta y}\right)^2 \left(f_{m,n+1}^{(k)} + f_{m,n-1}^{(k)}\right) - f_{m,n}^{(k-1)}$$

In order to find $f_{m,n}^{(1)}$ and $f_{m,n}^{(2)}$ the initial conditions must be used. To find elements with $m = 1$, $m = M$, $n = 1$ and $n = N$ the boundary conditions must be known.

Let us start with the initial condition $f(x, y, t = t_1) = s(x, y)$ (the initial shape of the membrane). In the discrete form it can be directly applied to determine $f_{m,n}^{(1)}$

$$f_{m,n}^{(1)} = s(x_m, y_n) = s_{m,n}$$

Wave equation in two dimensions

Problem 6.1: Use finite difference method to simulate a vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane is known.

Solution: In order to find $f_{m,n}^{(2)}$ the initial condition $\frac{\partial f}{\partial t}(x, y, t = t_1) = g(x, y)$

$$\frac{f_{m,n}^{(2)} - f_{m,n}^{(0)}}{2\Delta t} = g(x_m, y_n) = g_{m,n}$$

must be combined with the general formula for $k = 1$

$$f_{m,n}^{(2)} = 2\left(1 - \left(\frac{v\Delta t}{\Delta x}\right)^2 - \left(\frac{v\Delta t}{\Delta y}\right)^2\right) f_{m,n}^{(1)} + \left(\frac{v\Delta t}{\Delta x}\right)^2 \left(f_{m+1,n}^{(1)} + f_{m-1,n}^{(1)}\right) + \left(\frac{v\Delta t}{\Delta y}\right)^2 \left(f_{m,n+1}^{(1)} + f_{m,n-1}^{(1)}\right) - f_{m,n}^{(0)}$$

Finally, after some algebra

$$f_{m,n}^{(2)} = \left(1 - \left(\frac{v\Delta t}{\Delta x}\right)^2 - \left(\frac{v\Delta t}{\Delta y}\right)^2\right) f_{m,n}^{(1)} + \frac{1}{2}\left(\frac{v\Delta t}{\Delta x}\right)^2 \left(f_{m+1,n}^{(1)} + f_{m-1,n}^{(1)}\right) + \frac{1}{2}\left(\frac{v\Delta t}{\Delta y}\right)^2 \left(f_{m,n+1}^{(1)} + f_{m,n-1}^{(1)}\right) + \Delta t g_{m,n}$$

Wave equation in two dimensions

Problem 6.1: Use finite difference method to simulate a vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane is known.

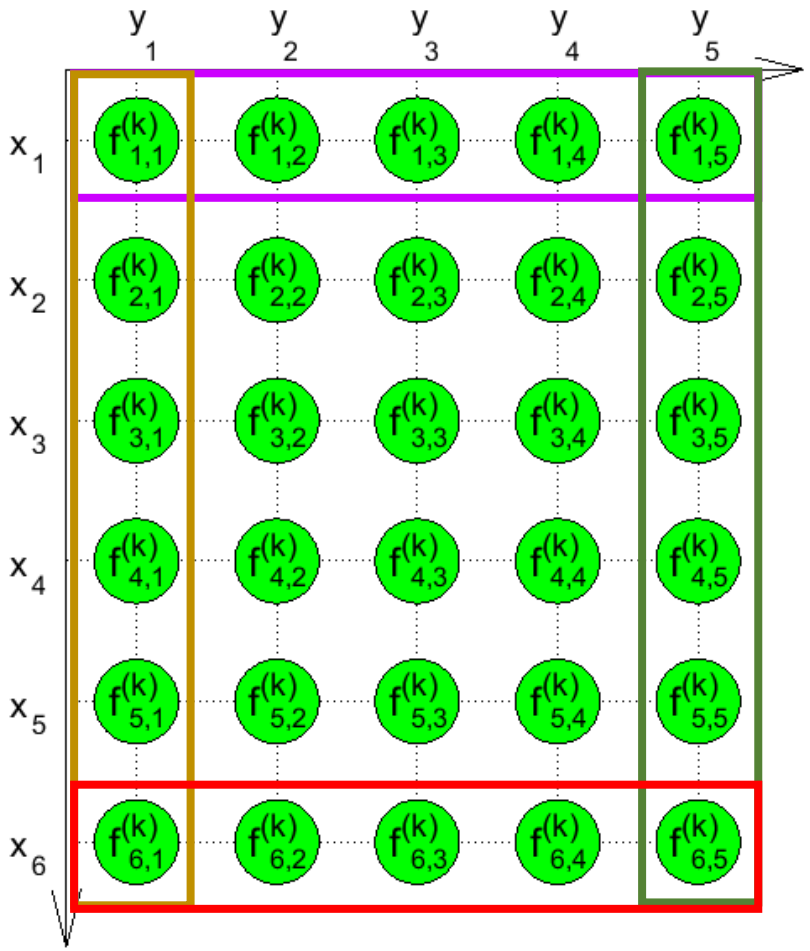
Solution: The boundary conditions can be applied directly at the domain limits

$$f(x = x_M, y, t) = d(y, t) \quad \rightarrow f_{M,n}^{(k)} = d(y_n, t_k) = d_n^{(k)}$$

$$f(x, y = y_N, t) = r(x, t) \quad \rightarrow f_{m,N}^{(k)} = r(x_m, t_k) = r_m^{(k)}$$

$$f(x = x_1, y, t) = u(y, t) \quad \rightarrow f_{1,n}^{(k)} = u(y_n, t_k) = u_n^{(k)}$$

$$f(x, y = y_1, t) = l(x, t) \quad \rightarrow f_{1,m}^{(k)} = l(x_m, t_k) = l_m^{(k)}$$



Wave equation in two dimensions

Problem 6.1: Use finite difference method to simulate a vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane is known.

Solution: Let us assume $d(y,t) = r(x,t) = u(y,t) = l(x,t) = 0$, $\mathrm{s}(x,y) = x^2(x-a)y^2(y-b)$ and $\mathrm{g}(x,y) = 0$.

To obtain the stable results the Courant–Friedrichs–Lewy condition must be fulfilled $\frac{v\Delta t}{\Delta x} + \frac{v\Delta t}{\Delta y} \leq 1$

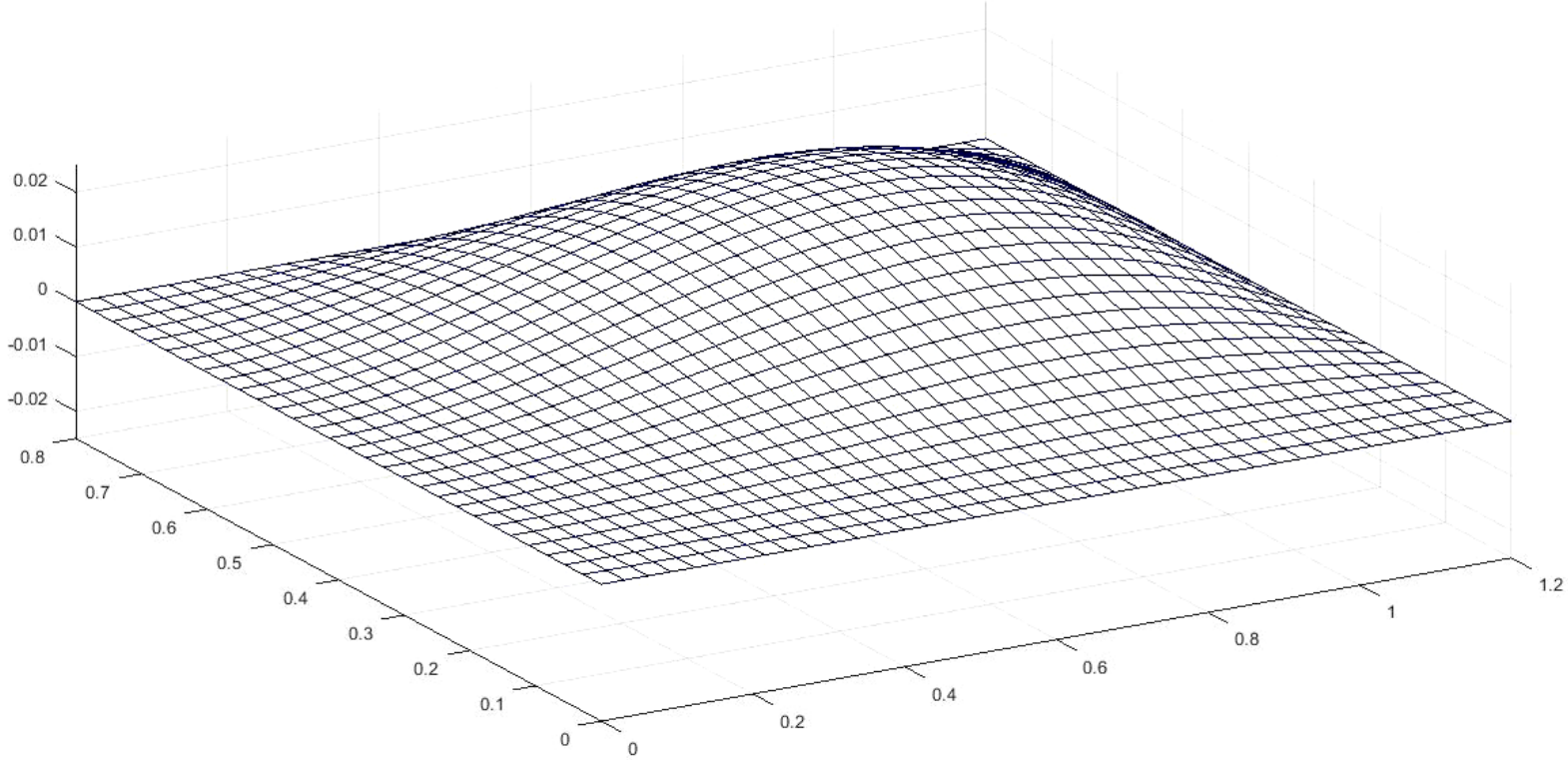
```
MATLAB CODE EXAMPLE:

v=100; a=1.2; b=0.8; M=40; N=30; K=1000; x=linspace(0,a,M); y=linspace(0,b,N);
dx=x(2)-x(1); dy=y(2)-y(1); dt=dx*dy/v/(dx+dy); px=(v*dt/dx)^2; py=(v*dt/dy)^2;
[X,Y]=meshgrid(x,y); X=X'; Y=Y'; s=X.^2.*(a-X).*Y.^2.*(b-Y); g=zeros(M,N);
d=zeros(1,N,K); r=zeros(M,1,K); u=zeros(1,N,K); l=zeros(M,1,K); f=zeros(M,N,K);
f(M,:,:)=d(1,:,:); f(:,N,:)=r(:,1,:); f(1,:,:)=u(1,:,:); f(:,1,:)=l(:,1,:);

f(:,:,1)=s;
f(2:M-1,2:N-1,2)=(1-px-py)*f(2:M-1,2:N-1,1)+px*(f(3:M,2:N-1,1)+f(1:M-2,2:N-1,1))/2
                 +py*(f(2:M-1,3:N,1)+f(2:M-1,1:N-2,1))/2+dt*g(2:M-1,2:N-1);
for k=2:K-1
    f(2:M-1,2:N-1,k+1)=2*(1-px-py)*f(2:M-1,2:N-1,k)+px*(f(3:M,2:N-1,k)+f(1:M-2,2:N-1,k))
                       +py*(f(2:M-1,3:N,k)+f(2:M-1,1:N-2,k))-f(2:M-1,2:N-1,k-1);
end
```

Wave equation in two dimensions

Problem 6.1: Use finite difference method to simulate a vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane is known.

Solution: The results of the analysis for $v = 100$, $a = 1.2$, $b = 0.8$ with the assumed conditions

$$d(y,t) = 0$$
$$r(x,t) = 0$$
$$u(y,t) = 0$$
$$l(x,t) = 0$$
$$\mathrm{s}(x,y) = x^2(x-a)y^2(y-b)$$
$$\mathrm{g}(x,y) = 0$$



Wave equation in two dimensions

Problem 6.2: Use finite difference method to simulate a damped vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: The damped wave equation has one extra term which represents frictional force (proportional to the velocity of the membrane)

$$\frac{1}{v^2}\frac{\partial^2 f}{\partial t^2}(x,y,t) = \frac{\partial^2 f}{\partial x^2}(x,y,t) + \frac{\partial^2 f}{\partial y^2}(x,y,t) - 2\beta\frac{\partial f}{\partial t}(x,y,t)$$

In the discrete form

$$\frac{1}{v^2}\frac{f_{m,n}^{(k+1)} - 2f_{m,n}^{(k)} + f_{m,n}^{(k-1)}}{\Delta t^2} = \frac{f_{m+1,n}^{(k)} - 2f_{m,n}^{(k)} + f_{m-1,n}^{(k)}}{\Delta x^2} + \frac{f_{m,n+1}^{(k)} - 2f_{m,n}^{(k)} + f_{m,n-1}^{(k)}}{\Delta y^2} - 2\beta\frac{f_{m,n}^{(k+1)} - f_{m,n}^{(k-1)}}{2\Delta t}$$

Where the first time derivative is approximated as follows

$$\frac{\partial f}{\partial t}(x_m, y_n, t_k) \approx \frac{f_{m,n}^{(k+1)} - f_{m,n}^{(k-1)}}{2\Delta t}$$

Wave equation in two dimensions

Problem 6.2: Use finite difference method to simulate a damped vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: The general formula for the $k+1$ time step is similar to the undamped case

$$f_{m,n}^{(k+1)} = \frac{2}{1+\beta\Delta t}\left(1-\left(\frac{v\Delta t}{\Delta x}\right)^2-\left(\frac{v\Delta t}{\Delta y}\right)^2\right)f_{m,n}^{(k)} + \frac{1}{1+\beta\Delta t}\left(\frac{v\Delta t}{\Delta x}\right)^2\left(f_{m+1,n}^{(k)}+f_{m-1,n}^{(k)}\right)$$
$$+\frac{1}{1+\beta\Delta t}\left(\frac{v\Delta t}{\Delta y}\right)^2\left(f_{m,n+1}^{(k)}+f_{m,n-1}^{(k)}\right)-\frac{1-\beta\Delta t}{1+\beta\Delta t}f_{m,n}^{(k-1)}$$

Also the second time step formula is similar

$$f_{m,n}^{(2)} = \left(1-\left(\frac{v\Delta t}{\Delta x}\right)^2-\left(\frac{v\Delta t}{\Delta y}\right)^2\right)f_{m,n}^{(1)} + \frac{1}{2}\left(\frac{v\Delta t}{\Delta x}\right)^2\left(f_{m+1,n}^{(1)}+f_{m-1,n}^{(1)}\right)$$
$$+\frac{1}{2}\left(\frac{v\Delta t}{\Delta y}\right)^2\left(f_{m,n+1}^{(1)}+f_{m,n-1}^{(1)}\right)+(1-\beta\Delta t)\Delta t g_{m,n}$$

Wave equation in two dimensions

Problem 6.2: Use finite difference method to simulate a damped vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: Let us assume $d(y,t) = r(x,t) = u(y,t) = l(x,t) = 0$, $s(x,y) = x^2(x-a)y^2(y-b)$ and $g(x,y) = 0$, once again.

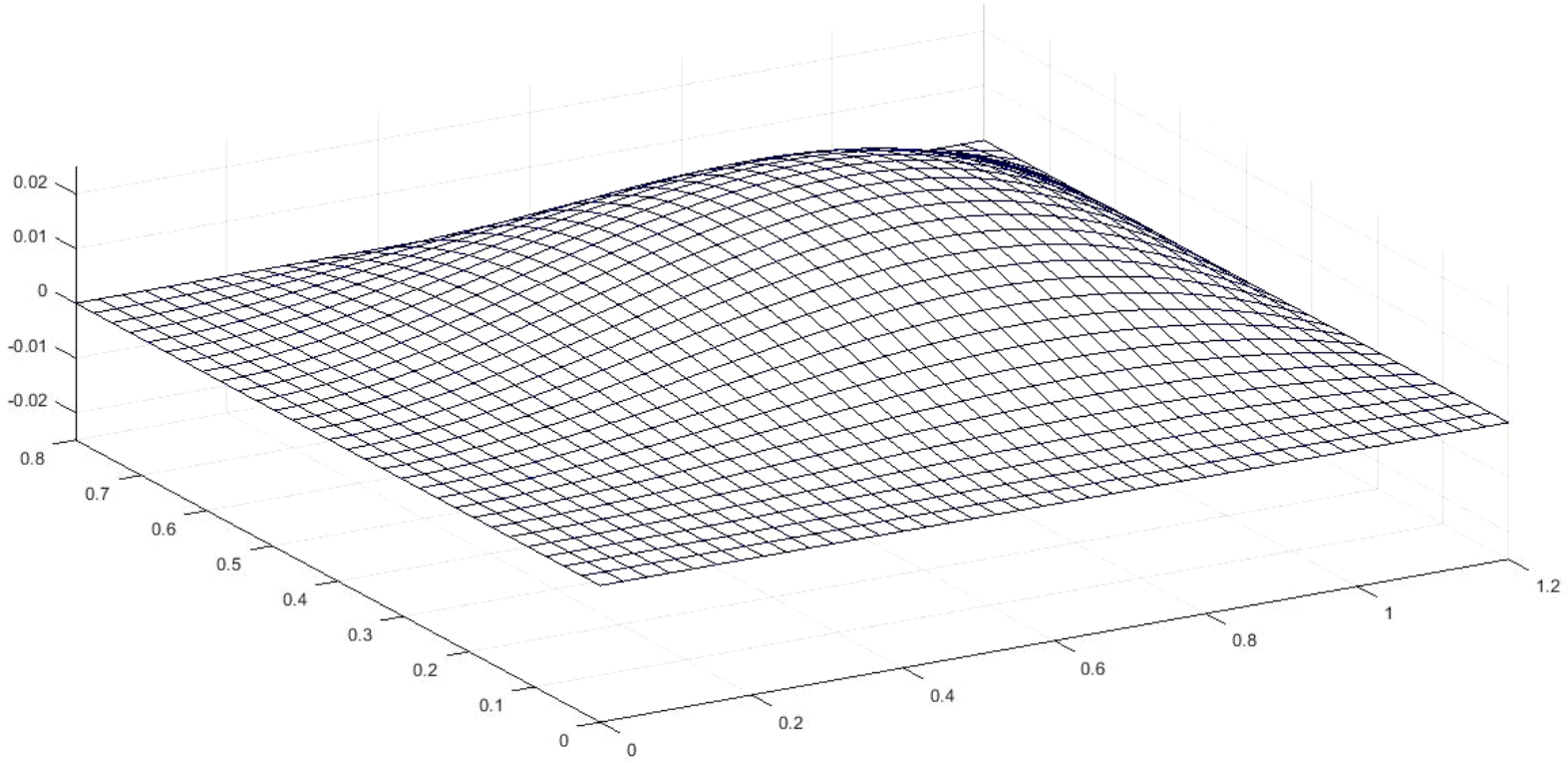
```
MATLAB CODE EXAMPLE:

v=100; a=1.2; b=0.8; M=40; N=30; K=1000; x=linspace(0,a,M); y=linspace(0,b,N);
dx=x(2)-x(1); dy=y(2)-y(1); dt=dx*dy/v/(dx+dy); px=(v*dt/dx)^2; py=(v*dt/dy)^2;
beta=20; q=1+beta*dt; w=1-beta*dt;
[X,Y]=meshgrid(x,y); X=X'; Y=Y'; s=X.^2.*(a-X).*Y.^2.*(b-Y); g=zeros(M,N);
d=zeros(1,N,K); r=zeros(M,1,K); u=zeros(1,N,K); l=zeros(M,1,K); f=zeros(M,N,K);
f(M,:,:)=d(1,:,:); f(:,N,:)=r(:,1,:); f(1,:,:)=u(1,:,:); f(:,1,:)=l(:,1,:);

f(:,:,1)=s;
f(2:M-1,2:N-1,2)=(1-px-py)*f(2:M-1,2:N-1,1)+px*(f(3:M,2:N-1,1)+f(1:M-2,2:N-1,1))/2
                  +py*(f(2:M-1,3:N,1)+f(2:M-1,1:N-2,1))/2 + w*dt*g(2:M-1,2:N-1);
for k=2:K-1
    f(2:M-1,2:N-1,k+1)=2/q*(1-px-py)*f(2:M-1,2:N-1,k)+px*(f(3:M,2:N-1,k)+f(1:M-2,2:N-1,k))/q
                        +py*( f(2:M-1,3:N,k)+f(2:M-1,1:N-2,k))/q-w*f(2:M-1,2:N-1,k-1)/q;
end
```

Wave equation in two dimensions

Problem 6.2: Use finite difference method to simulate a damped vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: The results of the analysis for $v = 100$, $\beta = 20$, $a = 1.2$, $b = 0.8$ with the assumed conditions

$$d(y,t) = 0$$
$$r(x,t) = 0$$
$$u(y,t) = 0$$
$$l(x,t) = 0$$
$$\mathrm{s}(x,y) = x^2(x-a)y^2(y-b)$$
$$\mathrm{g}(x,y) = 0$$



Wave equation in two dimensions

Problem 6.2: Use finite difference method to simulate a damped vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: The boundary conditions can be simply modified, for example, let us take : $d(y,t) = r(x,t) = u(y,t) = 0$, $l(x,t) = b\left(\frac{a}{2} - \left|x - \frac{a}{2}\right|\right)$, $s(x,y) = \left(\frac{a}{2} - \left|x - \frac{a}{2}\right|\right)(b-y)$ and $g(x,y) = 0$.

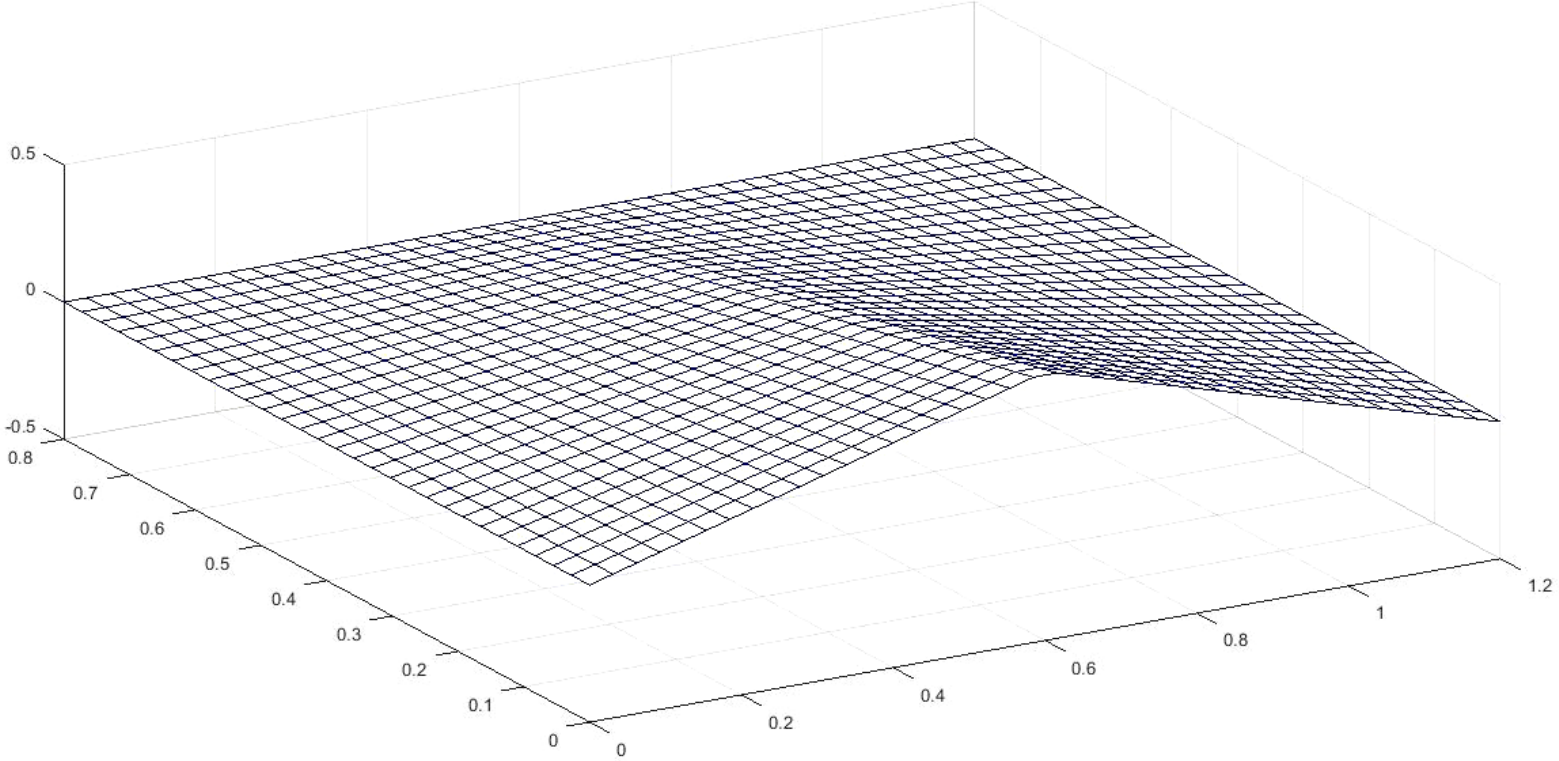
```
MATLAB CODE EXAMPLE:

v=100; a=1.2; b=0.8; M=40; N=30; K=1000; x=linspace(0,a,M); y=linspace(0,b,N);
dx=x(2)-x(1); dy=y(2)-y(1); dt=dx*dy/v/(dx+dy); px=(v*dt/dx)^2; py=(v*dt/dy)^2;
beta=20; q=1+beta*dt; w=1-beta*dt; [X,Y]=meshgrid(x,y); X=X'; Y=Y';
s=X.^2.*(a-X).*Y.^2.*(b-Y); s=1*(b-Y).*(a/2-abs(X-a/2)); g=zeros(M,N); d=zeros(1,N,K);
r=zeros(M,1,K); u=d; l(1:M,1,1:K)=(a/2-abs(x-a/2))'*ones(1,K); f=zeros(M,N,K);
f(M,:,:)=d(1,:,:); f(:,N,:)=r(:,1,:); f(1,:,:)=u(1,:,:); f(:,1,:)=l(:,1,:);

f(:,:,1)=s;
f(2:M-1,2:N-1,2)=(1-px-py)*f(2:M-1,2:N-1,1)+px*(f(3:M,2:N-1,1)+f(1:M-2,2:N-1,1))/2
                  +py*(f(2:M-1,3:N,1)+f(2:M-1,1:N-2,1))/2 + w*dt*g(2:M-1,2:N-1);
for k=2:K-1
    f(2:M-1,2:N-1,k+1)=2/q*(1-px-py)*f(2:M-1,2:N-1,k)+px*(f(3:M,2:N-1,k)+f(1:M-2,2:N-1,k))/q
                        +py*( f(2:M-1,3:N,k)+f(2:M-1,1:N-2,k))/q-w*f(2:M-1,2:N-1,k-1)/q;
end
```

Wave equation in two dimensions

Problem 6.2: Use finite difference method to simulate a damped vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: The results of the analysis for $v = 100$, $\beta = 20$, $a = 1.2$, $b = 0.$ 8with the assumed conditions

$$d(y,t) = 0$$
$$r(x,t) = 0$$
$$u(y,t) = 0$$
$$l(x,t) = b\left(\frac{\mathrm{a}}{2} - \left|\mathrm{x} - \frac{\mathrm{a}}{2}\right|\right)$$
$$\mathrm{s}(x,y) = \left(\frac{\mathrm{a}}{2} - \left|\mathrm{x} - \frac{\mathrm{a}}{2}\right|\right)(b - y)$$
$$\mathrm{g}(x,y) = 0$$



Wave equation in two dimensions

Problem 6.2: Use finite difference method to simulate a damped vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: The boundary conditions can be time dependent, for example, let us take : $d(y,t) = r(x,t) = u(y,t) = 0$, $l(x,t) = \left(\frac{a}{2} - \left|x - \frac{a}{2}\right|\right)\sin(5t)$, $s(x,y) = 0$ and $g(x,y) = 0$.

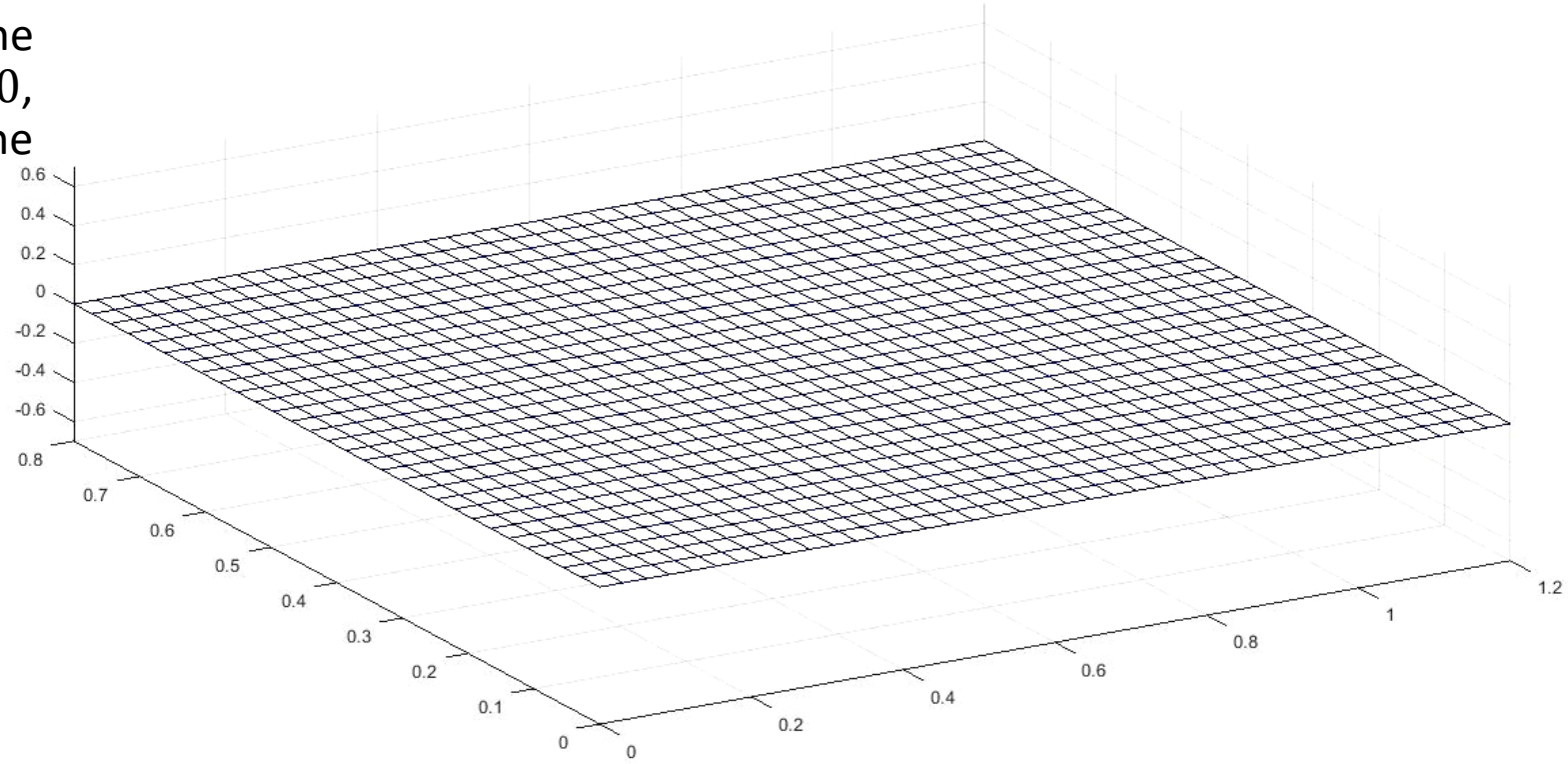
```
MATLAB CODE EXAMPLE:

v=100; a=1.2; b=0.8; M=40; N=30; K=1000; x=linspace(0,a,M); y=linspace(0,b,N);
dx=x(2)-x(1); dy=y(2)-y(1); dt=dx*dy/v/(dx+dy); px=(v*dt/dx)^2; py=(v*dt/dy)^2;
beta=20; q=1+beta*dt; w=1-beta*dt; [X,Y]=meshgrid(x,y); X=X'; Y=Y';
s=X.^2.*(a-X).*Y.^2.*(b-Y); s=zeros(M,N); g=s; d=zeros(1,N,K); r=zeros(M,1,K);
u=d; t=dx*[0:(K-1)]; l(1:M,1,1:K)=(a/2-abs(x-a/2))'*sin(5*t); f=zeros(M,N,K);
f(M,:,:)=d(1,:,:); f(:,N,:)=r(:,1,:); f(1,:,:)=u(1,:,:); f(:,1,:)=l(:,1,:);

f(:,:,1)=s;
f(2:M-1,2:N-1,2)=(1-px-py)*f(2:M-1,2:N-1,1)+px*(f(3:M,2:N-1,1)+f(1:M-2,2:N-1,1))/2
                  +py*(f(2:M-1,3:N,1)+f(2:M-1,1:N-2,1))/2 + w*dt*g(2:M-1,2:N-1);
for k=2:K-1
    f(2:M-1,2:N-1,k+1)=2/q*(1-px-py)*f(2:M-1,2:N-1,k)+px*(f(3:M,2:N-1,k)+f(1:M-2,2:N-1,k))/q
                        +py*( f(2:M-1,3:N,k)+f(2:M-1,1:N-2,k))/q-w*f(2:M-1,2:N-1,k-1)/q;
end
```

Wave equation in two dimensions

Problem 6.2: Use finite difference method to simulate a damped vibrating rectangular membrane. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: The results of the analysis for $v = 100$, $\beta = 20$, $a = 1.2$, $b = 0.8$ with the assumed conditions

$$d(y,t) = 0$$
$$r(x,t) = 0$$
$$u(y,t) = 0$$
$$l(x,t) = \left(\frac{a}{2} - \left|x - \frac{a}{2}\right|\right)\sin(5t)$$
$$s(x,y) = 0$$
$$g(x,y) = 0$$



Wave equation in two dimensions

Problem 6.3: Use finite difference method to simulate a vibrating membrane of an arbitrary shape. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: It is convenient to replace the double space indexes of the function to a single index (also for rectangular domain)

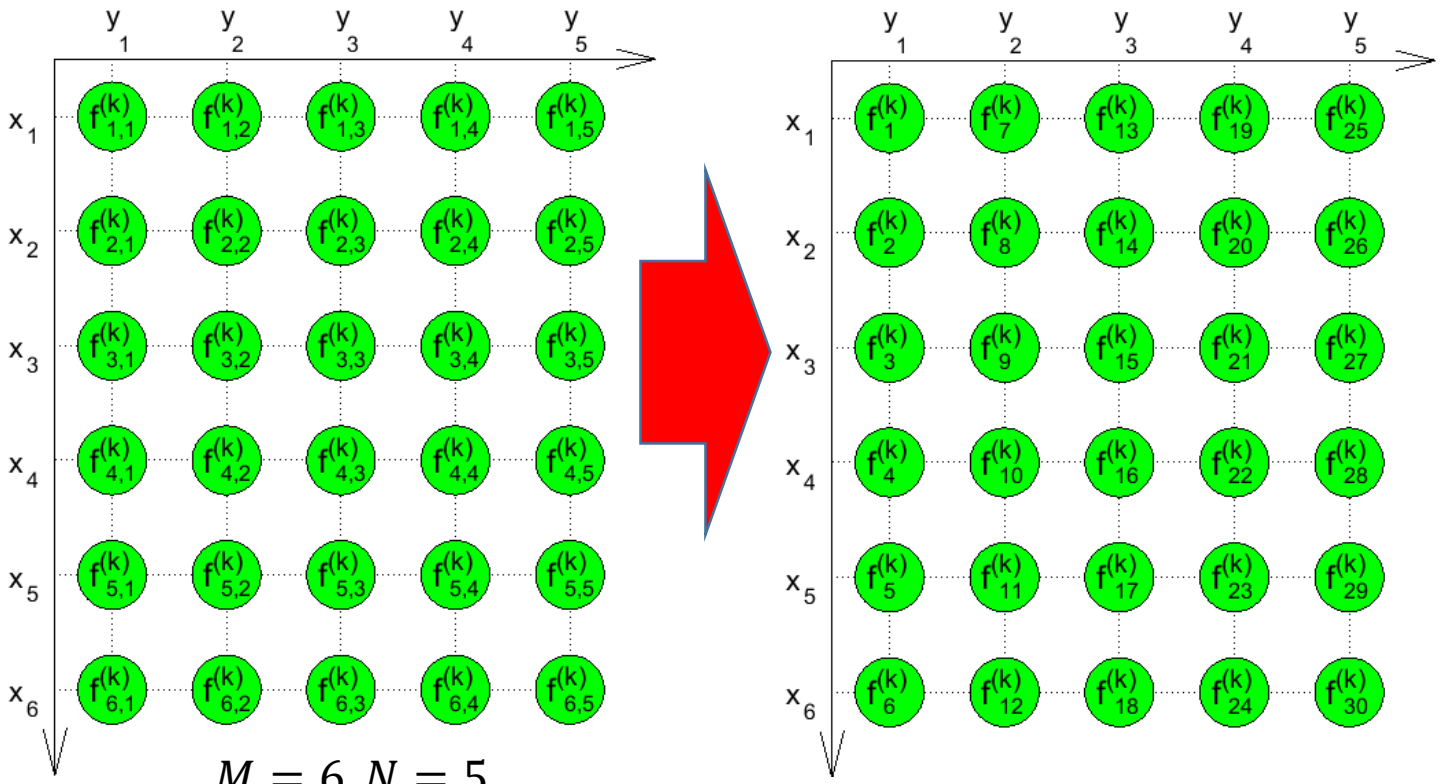
If $m = 1, \dots, M$ and $n = 1, \dots, N$, then

$$f_{m,n}^{(k)} \to f_i^{(k)}$$

where

$$i = m + (n-1)M$$

and $i = 1, \dots, MN$.



$M = 6, N = 5$

Wave equation in two dimensions

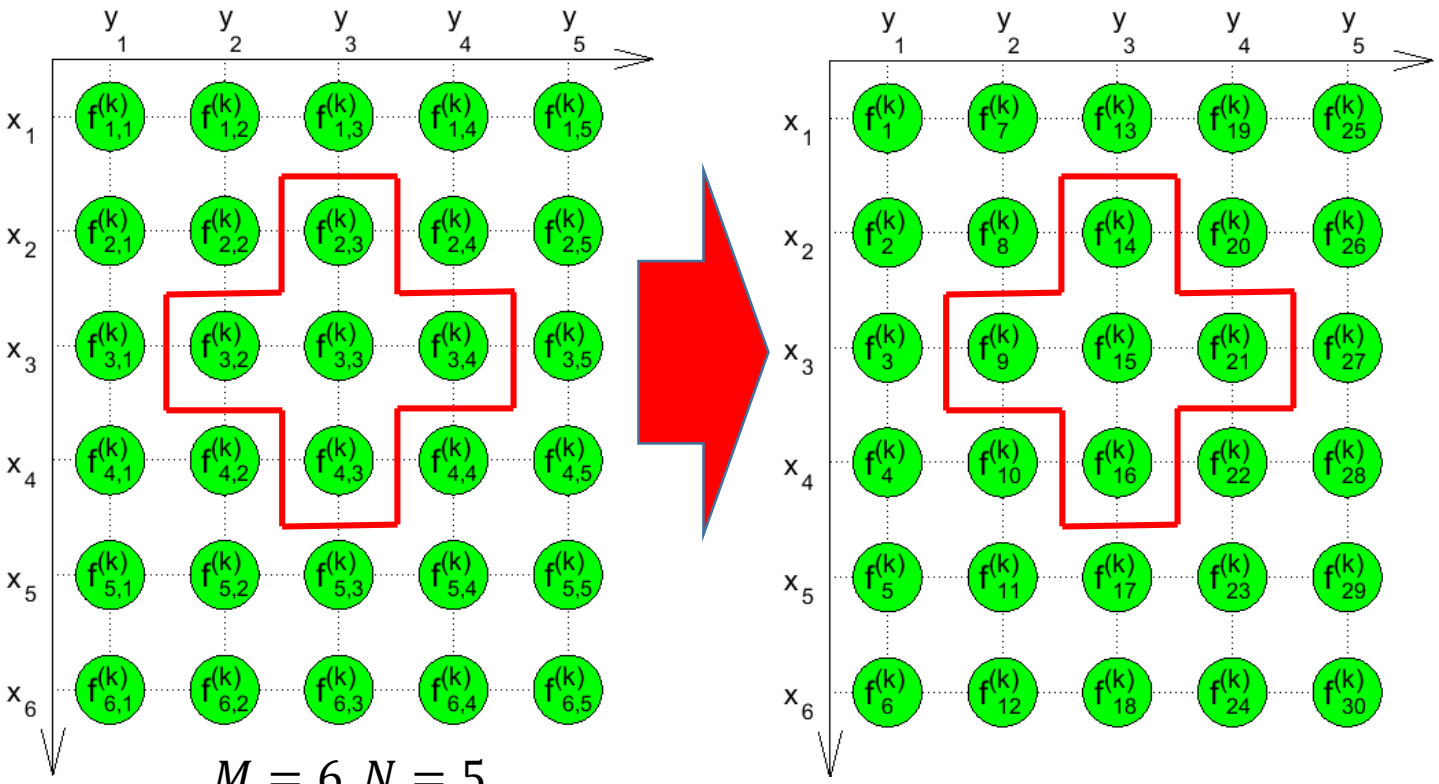
Problem 6.3: Use finite difference method to simulate a vibrating membrane of an arbitrary shape. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: By the simple substitution $i = m + (1 - n)M$ all the finite difference formulas can by modified:

$$f_{m,n}^{(k)} \rightarrow f_i^{(k)}$$

$$f_{m\pm1,n}^{(k)} \rightarrow f_{i\pm1}^{(k)}$$

$$f_{m,n\pm1}^{(k)} \rightarrow f_{i\pm M}^{(k)}$$



$M = 6, N = 5$

Wave equation in two dimensions

Problem 6.3: Use finite difference method to simulate a vibrating membrane of an arbitrary shape. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: Substituting $f_{m,n}^{(k)} \to f_i^{(k)}$, $f_{m\pm1,n}^{(k)} \to f_{i\pm1}^{(k)}$, $f_{m,n\pm1}^{(k)} \to f_{i\pm M}^{(k)}$ we get the general formula

$$f_i^{(k+1)} = \frac{2}{1+\beta\Delta t}\left(1-\left(\frac{v\Delta t}{\Delta x}\right)^2-\left(\frac{v\Delta t}{\Delta y}\right)^2\right)f_i^{(k)} + \frac{1}{1+\beta\Delta t}\left(\frac{v\Delta t}{\Delta x}\right)^2\left(f_{i+1}^{(k)}+f_{i-1}^{(k)}\right) + \frac{1}{1+\beta\Delta t}\left(\frac{v\Delta t}{\Delta y}\right)^2\left(f_{i+M}^{(k)}+f_{i-M}^{(k)}\right) - \frac{1-\beta\Delta t}{1+\beta\Delta t}f_i^{(k-1)}$$

and the initial relations $f_i^{(1)} = s_i$ and

$$f_i^{(2)} = \left(1-\left(\frac{v\Delta t}{\Delta x}\right)^2-\left(\frac{v\Delta t}{\Delta y}\right)^2\right)f_i^{(1)} + \frac{1}{2}\left(\frac{v\Delta t}{\Delta x}\right)^2\left(f_{i+1}^{(1)}+f_{i-1}^{(1)}\right) + \frac{1}{2}\left(\frac{v\Delta t}{\Delta y}\right)^2\left(f_{i+M}^{(1)}+f_{i-M}^{(1)}\right) + (1-\beta\Delta t)\Delta t g_i$$

The boundary conditions must be modified in the same way, but since the domain is irregular, then the separation into four edges is unreasonable.

Wave equation in two dimensions

Problem 6.3: Use finite difference method to simulate a vibrating membrane of an arbitrary shape. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: In the first step the nodes must be divided into three groups:

- inside nodes

$$P_{in} = [8, 9, 10, 11, 14, 15, 16, 20, 21]$$

- boundary nodes

$$P_{bnd} = [1,2,3,4,5,6,7, 12,13,17,18,19,22,23,25,26,27,28]$$

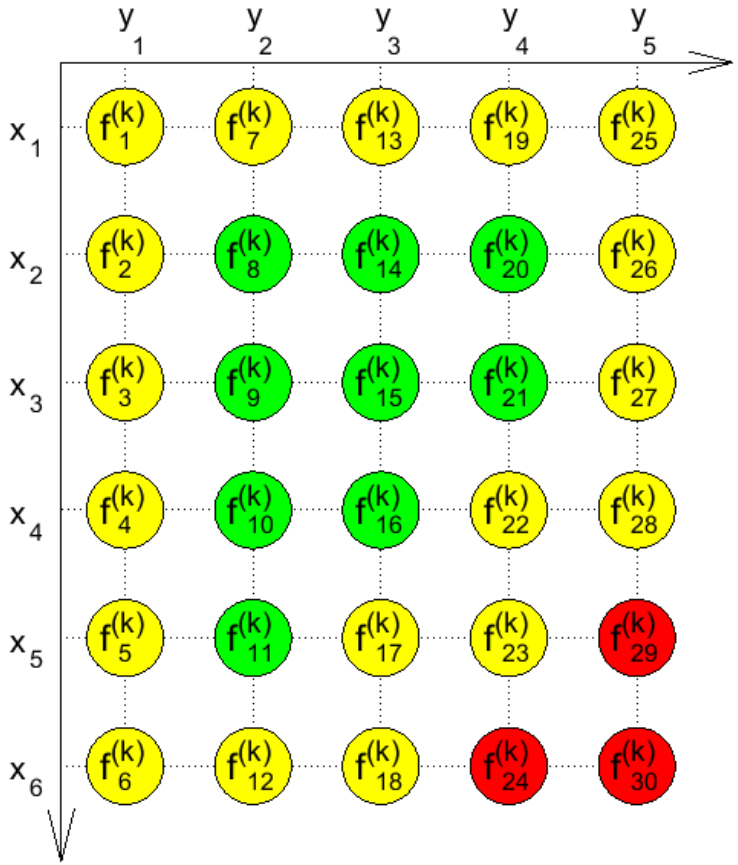
- outside nodes

$$P_{out} = [24,29,30]$$

Then the values of the function at the boundary L represented by P_{bnd} must be imposed

$$f(x,y)\Big|_{(x,y)\in L} = B(x,y)$$

Finally, the initial conditions can be applied and the derived formulas for $f_i^{(k+1)}$ can be used for $i \in P_{in}$.



Wave equation in two dimensions

Problem 6.3: Use finite difference method to simulate a vibrating membrane of an arbitrary shape. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: For simple boundary conditions $B_i = \{2\ for\ i = 15\ or\ i = 16\ and\ \ 0\ otherwise\}$ and initial conditions $s_i = \{2\ for\ i = 3\ and\ i = 4,\ \ 1\ for\ i = 2\ and\ i = 5\ and\ 0\ otherwise\ \}$ and $g_i = 0$.

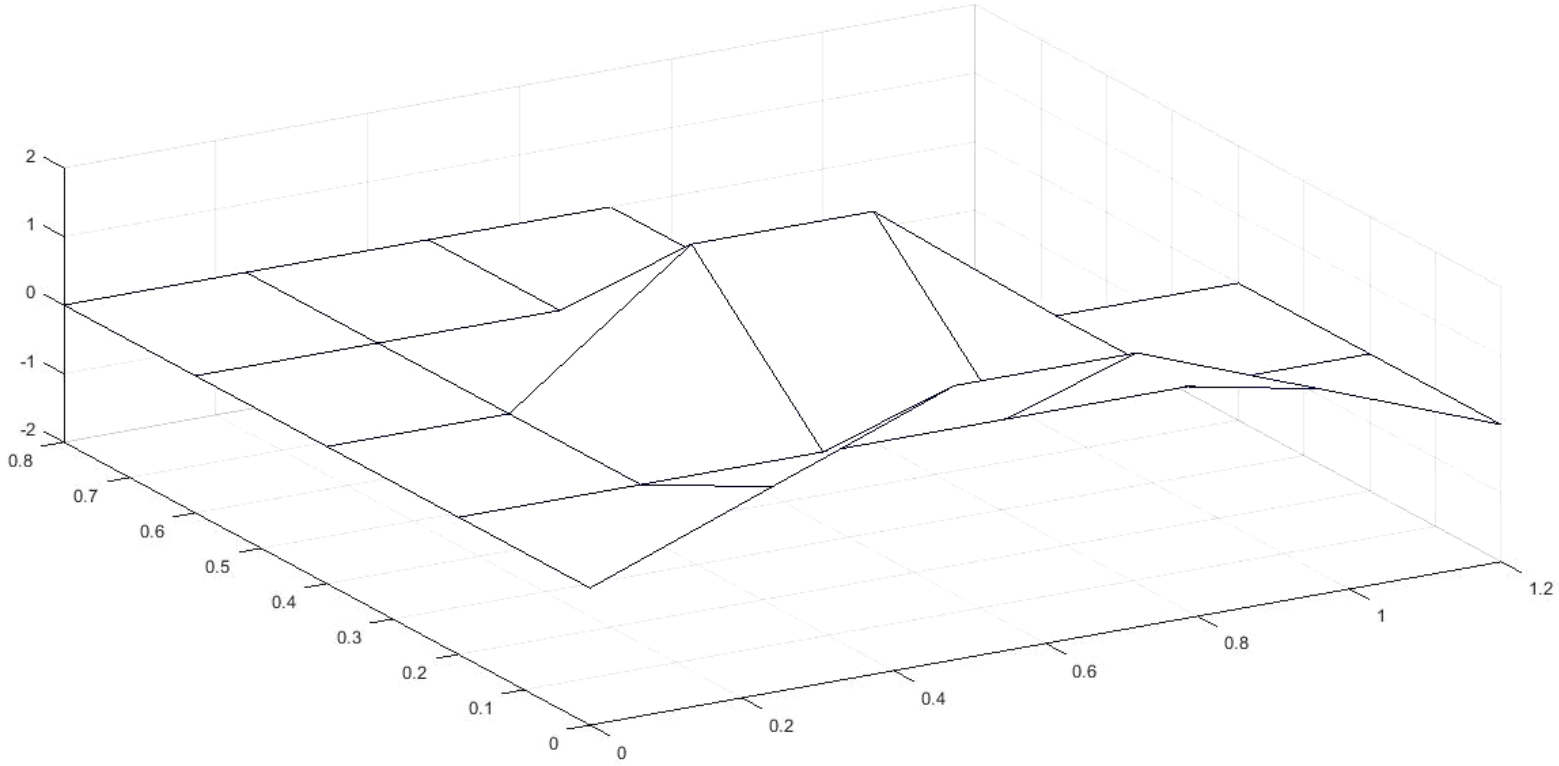
```
MATLAB CODE EXAMPLE:

v=100; beta=40; a=1.2; b=0.8; M=6; N=5; K=1000; x=linspace(0,a,M); y=linspace(0,b,N); dx=x(2)-x(1); dy=y(2)-y(1);
dt=dx*dy/v/(dx+dy)/10; px=(v*dt/dx)^2; py=(v*dt/dy)^2; q=1+beta*dt; w=1-beta*dt; [X,Y]=meshgrid(x,y); X=X'; Y=Y';
s=zeros(N*M,1); s([15,16],1)=2; g=zeros(N*M,1); Pin=[8, 9, 10, 11,14,15,16,20,21]; Pout=[24,29,30];
Pbnd=[1,2,3,4,5,6,7,12,13,17,18,19,22,23,25,26,27,28]; Bnd= [0,1,2,2,1,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
f=zeros(M*N,K);
f(Pout,:)=ones(length(Pout),K)*NaN;
f(Pbnd,:)=Bnd'*ones(1,K);
f(Pin,1)=s(Pin);
for i=Pin
    f(i,2)=(1-px-py)*f(i,1)+px*(f(i+1,1)+f(i-1,1))/2+py*(f(i+M,1)+f(i-M,1))/2+w*dt*g(i);
end
for k=2:K-1
    for i=Pin
        f(i,k+1)=2/q*(1-px-py)*f(i,k)+px*(f(i+1,k)+f(i-1,k))/q + py*(f(i+M,k)+f(i-M,k))/q-w*f(i,k-1)/q;
    end
end
```

Wave equation in two dimensions

Problem 6.3: Use finite difference method to simulate a vibrating membrane of an arbitrary shape. Assume, that the membrane edges are fixed at the boundaries and the initial conditions are known. The wave velocity v in the membrane and the damping parameter β are known.

Solution: The results of the analysis for $v = 100$, $\beta = 20$, $a = 1.2$, $b = 0.8$ with the assumed boundary conditions

$$B_i = \{2 \; for \; i = 15 \; or \; i =$$



Wave equation in two dimensions

Problem 6.4: Use finite difference method to simulate the equilibrium solution for a vibrating membrane. Assume, that the membrane edges are fixed at the boundaries.

Solution: The equilibrium solution (position) of the wave equation

$$\frac{1}{v^2}\frac{\partial^2 f}{\partial t^2}(x,y,t) = \frac{\partial^2 f}{\partial x^2}(x,y,t) + \frac{\partial^2 f}{\partial y^2}(x,y,t) - 2\beta\frac{\partial f}{\partial t}(x,y,t)$$

can be found by simple reduction of time dependences $\frac{\partial^2 f}{\partial t^2}(x,y,t) = 0$ and $\frac{\partial f}{\partial t}(x,y,t) = 0$.

Hence, the problem becomes simple

$$\boxed{\frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y) = 0}$$

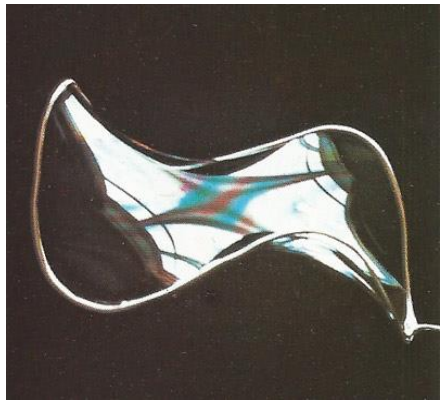
Laplace's equation

Pierre-Simon, marquis de Laplace (1749 – 1827) was an influential French scholar whose work was important to the development of mathematics, statistics, physics and astronomy.

Source: https://en.wikipedia.org/wiki/Pierre-Simon_Laplace

Wave equation in two dimensions

Problem 6.4: Use finite difference method to simulate the equilibrium solution for a vibrating membrane. Assume, that the membrane edges are fixed at the boundaries.



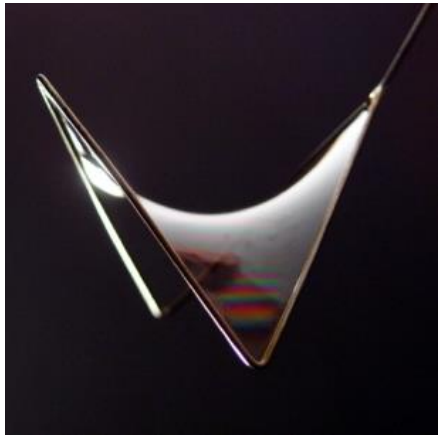
Solution: The Laplace's equation (equilibrium solution)

$$\frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y) = 0$$

can be solved with the use of finite difference method.

In the discrete form

$$\frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} + \frac{f_{i+M} - 2f_i + f_{i-M}}{\Delta y^2} = 0$$



where $f_i = f(x_m, y_n)$, $i = m + (1-n)M$.

Or, introducing coefficients $c_x = \frac{1}{\Delta x^2}$, $c_y = \frac{1}{\Delta y^2}$, $c = -\left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)$

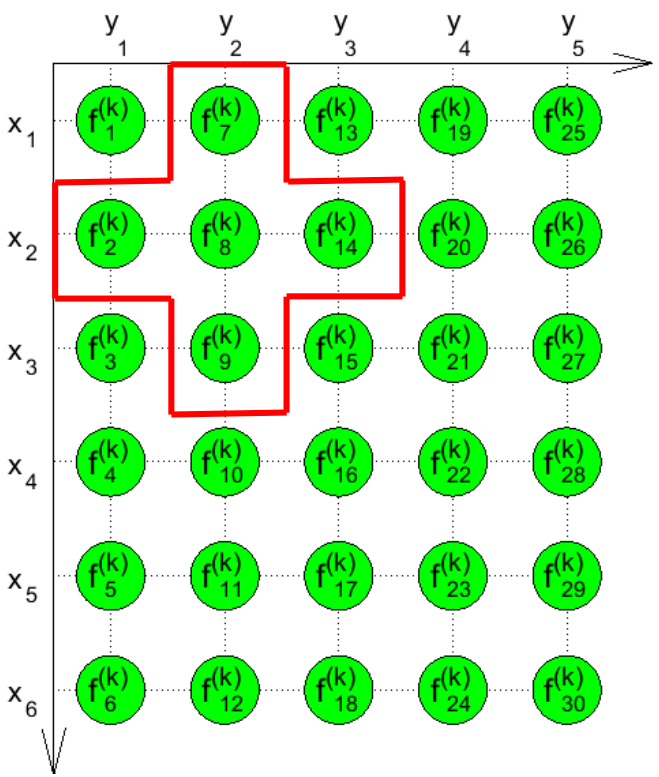
$$c_y f_{i-M} + c_x f_{i-1} + c f_i + c_x f_{i+1} + c_y f_{i+M} = 0$$

Wave equation in two dimensions

Problem 6.4: Use finite difference method to simulate the equilibrium solution for a vibrating membrane. Assume, that the membrane edges are fixed at the boundaries.

Solution: In the formula $c_y f_{i-M} + c_x f_{i-1} + c f_i + c_x f_{i+1} + c_y f_{i+M} = 0$ some of f_i are known (boundary) and some of f_i are unknown (inside). For instance (see the figure):

$$\begin{cases} c_y f_2 + c_x f_7 + c f_8 + c_x f_9 + c_y f_{14} = 0 \\ c_y f_3 + c_x f_8 + c f_9 + c_x f_{10} + c_y f_{15} = 0 \\ c_y f_4 + c_x f_9 + c f_{10} + c_x f_{11} + c_y f_{16} = 0 \\ c_y f_5 + c_x f_{10} + c f_{11} + c_x f_{12} + c_y f_{17} = 0 \\ c_y f_8 + c_x f_{13} + c f_{14} + c_x f_{15} + c_y f_{20} = 0 \\ c_y f_9 + c_x f_{14} + c f_{15} + c_x f_{16} + c_y f_{21} = 0 \\ c_y f_{10} + c_x f_{15} + c f_{16} + c_x f_{17} + c_y f_{22} = 0 \\ c_y f_{11} + c_x f_{16} + c f_{17} + c_x f_{18} + c_y f_{23} = 0 \\ c_y f_{14} + c_x f_{19} + c f_{20} + c_x f_{21} + c_y f_{26} = 0 \\ c_y f_{15} + c_x f_{20} + c f_{21} + c_x f_{22} + c_y f_{27} = 0 \\ c_y f_{16} + c_x f_{21} + c f_{22} + c_x f_{23} + c_y f_{28} = 0 \\ c_y f_{17} + c_x f_{22} + c f_{23} + c_x f_{24} + c_y f_{29} = 0 \end{cases}$$

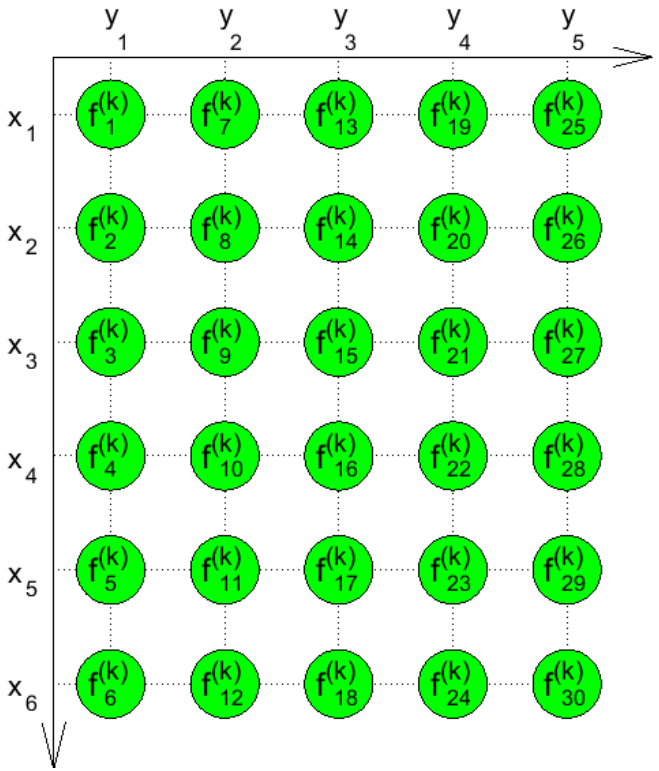


Wave equation in two dimensions

Problem 6.4: Use finite difference method to simulate the equilibrium solution for a vibrating membrane. Assume, that the membrane edges are fixed at the boundaries.

Solution: The problem boils down to the following system of equations

$$
\begin{cases}
cf_8 + c_x f_9 + c_y f_{14} = -c_y f_2 - c_x f_7 \\
c_x f_8 + cf_9 + c_x f_{10} + c_y f_{15} = -c_y f_3 \\
c_x f_9 + cf_{10} + c_x f_{11} + c_y f_{16} = -c_y f_4 \\
c_x f_{10} + cf_{11} + c_y f_{17} = -c_y f_5 - c_x f_{12} \\
c_y f_8 + cf_{14} + c_x f_{15} + c_y f_{20} = -c_x f_{13} \\
c_y f_9 + c_x f_{14} + cf_{15} + c_x f_{16} + c_y f_{21} = 0 \\
c_y f_{10} + c_x f_{15} + cf_{16} + c_x f_{17} + c_y f_{22} = 0 \\
c_y f_{11} + c_x f_{16} + cf_{17} + c_y f_{23} = -c_x f_{18} \\
c_y f_{14} + cf_{20} + c_x f_{21} = -c_x f_{19} - c_y f_{26} \\
c_y f_{15} + c_x f_{20} + cf_{21} + c_x f_{22} = -c_y f_{27} \\
c_y f_{16} + c_x f_{21} + cf_{22} + c_x f_{23} = -c_y f_{28} \\
c_y f_{17} + c_x f_{22} + cf_{23} = -c_x f_{24} - c_y f_{29} 0
\end{cases}
$$



Wave equation in two dimensions

Problem 6.4: Use finite difference method to simulate the equilibrium solution for a vibrating membrane. Assume, that the membrane edges are fixed at the boundaries.

Solution: Or in a matrix notation $\boldsymbol{A}\boldsymbol{f}_{\boldsymbol{unknown}} = \boldsymbol{B}\boldsymbol{f}_{\boldsymbol{known}}$

$$\begin{bmatrix}
c & c_x & 0 & 0 & 0 & c_y & 0 & 0 & 0 & 0 & 0 & 0 \\
c_x & c & c_x & 0 & 0 & 0 & c_y & 0 & 0 & 0 & 0 & 0 \\
0 & c_x & c & c_x & 0 & 0 & 0 & c_y & 0 & 0 & 0 & 0 \\
0 & 0 & c_x & c & c_x & 0 & 0 & 0 & c_y & 0 & 0 & 0 \\
0 & 0 & 0 & c_x & c & c_x & 0 & 0 & 0 & c_y & 0 & 0 \\
c_y & 0 & 0 & 0 & c_x & c & c_x & 0 & 0 & 0 & c_y & 0 \\
0 & c_y & 0 & 0 & 0 & c_x & c & c_x & 0 & 0 & 0 & c_y \\
0 & 0 & c_y & 0 & 0 & 0 & c_x & c & c_x & 0 & 0 & 0 \\
0 & 0 & 0 & c_y & 0 & 0 & 0 & c_x & c & c_x & 0 & 0 \\
0 & 0 & 0 & 0 & c_y & 0 & 0 & 0 & c_x & c & c_x & 0 \\
0 & 0 & 0 & 0 & 0 & c_y & 0 & 0 & 0 & c_x & c & c_x \\
0 & 0 & 0 & 0 & 0 & 0 & c_y & 0 & 0 & 0 & c_x & c
\end{bmatrix}
\begin{bmatrix}
f_8 \\ f_9 \\ f_{10} \\ f_{11} \\ f_{14} \\ f_{15} \\ f_{16} \\ f_{17} \\ f_{20} \\ f_{21} \\ f_{22} \\ f_{23}
\end{bmatrix}
= -
\begin{bmatrix}
0 & c_y & 0 & 0 & 0 & 0 & c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & c_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & c_y & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & c_y & 0 & 0 & c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c & 0 & 0 & c_y & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_y & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_y & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c & 0 & 0 & 0 & 0 & c_y & 0
\end{bmatrix}
\begin{bmatrix}
f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_{12} \\ f_{13} \\ f_{18} \\ f_{19} \\ f_{24} \\ f_{25} \\ f_{26} \\ f_{27} \\ f_{28} \\ f_{29} \\ f_{30}
\end{bmatrix}$$

Finally,

$$\boldsymbol{f}_{\boldsymbol{unknown}} = \boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{f}_{\boldsymbol{known}}$$

Wave equation in two dimensions

Problem 6.4: Use finite difference method to simulate the equilibrium solution for a vibrating membrane. Assume, that the membrane edges are fixed at the boundaries.

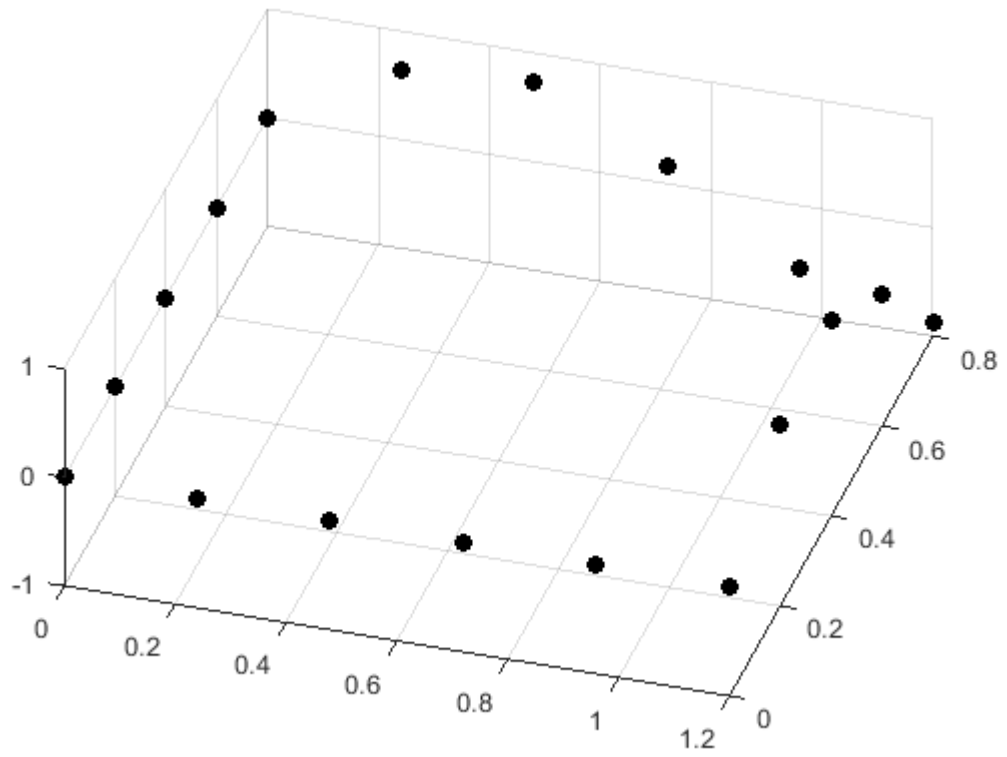
Solution: The implementation is quite simple

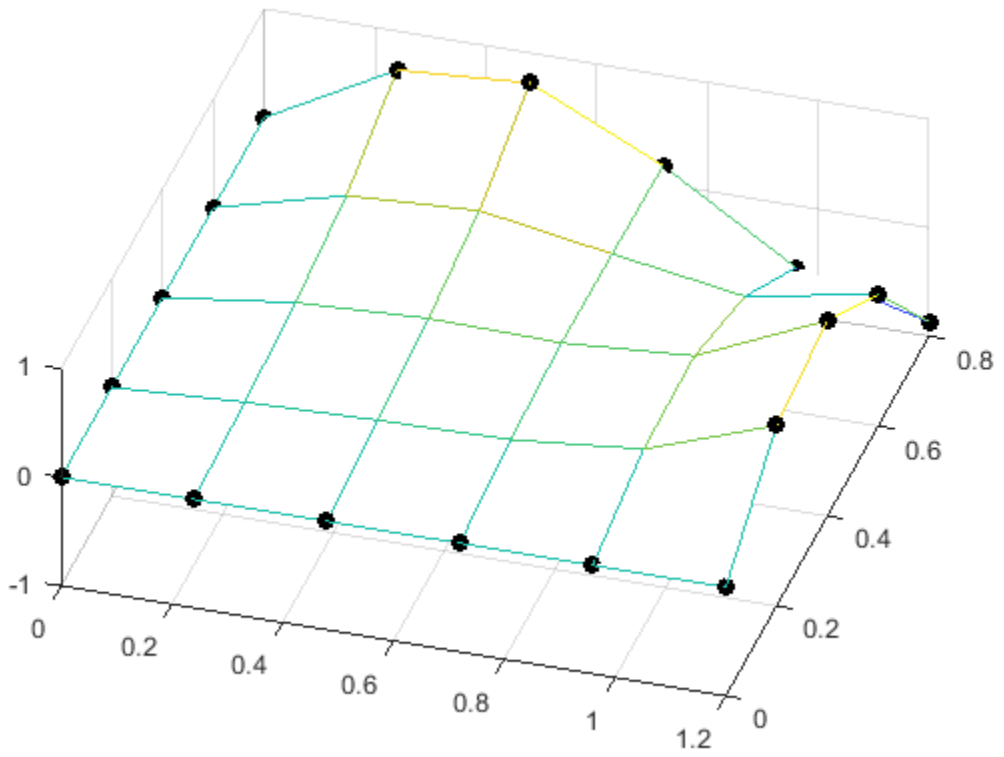
```
MATLAB CODE EXAMPLE:

M=6; N=5; a=1.2; b=0.8; x=linspace(0,a,M); y=linspace(0,b,N); dx=x(2)-x(1); dy=y(2)-y(1);
X,Y]=meshgrid(x,y); X=reshape(X',M*N,1); Y=reshape(Y',M*N,1);
Pout=[]; Pbnd=sort([1:M 1+M:M:1+(N-2)*M 2*M:M:(N-1)*M 1+(N-1)*M:N*M]); Pin=[1:M*N]; Pin(Pbnd)=[];
Xbnd=X(Pbnd); Ybnd=Y(Pbnd); Bnd=Xbnd.*sin(4*Ybnd)+Ybnd.*sin(4*Xbnd);
cx=(1/dx)^2; cy=(1/dy).^2; c=-2/dx^2-2/dy^2;

tmp=ones(1,M*N); tmp1=ones(1,M*N-1); tmpM=ones(1,M*N-M);
Temp=c*diag(tmp)+cx*diag(tmp1,-1)+cx*diag(tmp1,1)+cy*diag(tmpM,-M)+cy*diag(tmpM,M);
Temp(Pbnd,:)=[];
A=Temp(:,Pin);
B=-Temp(:,Pbnd);
f(Pbnd)=Bnd;
f(Pin)=A^(-1)*(B*Bnd);
mesh(reshape(X,M,N),reshape(Y,M,N),reshape(f,M,N))
```

Wave equation in two dimensions

Problem 6.4: Use finite difference method to simulate the equilibrium solution for a vibrating membrane. Assume, that the membrane edges are fixed at the boundaries.

Solution: The results are inaccurate and unreliable due to a very rare mesh

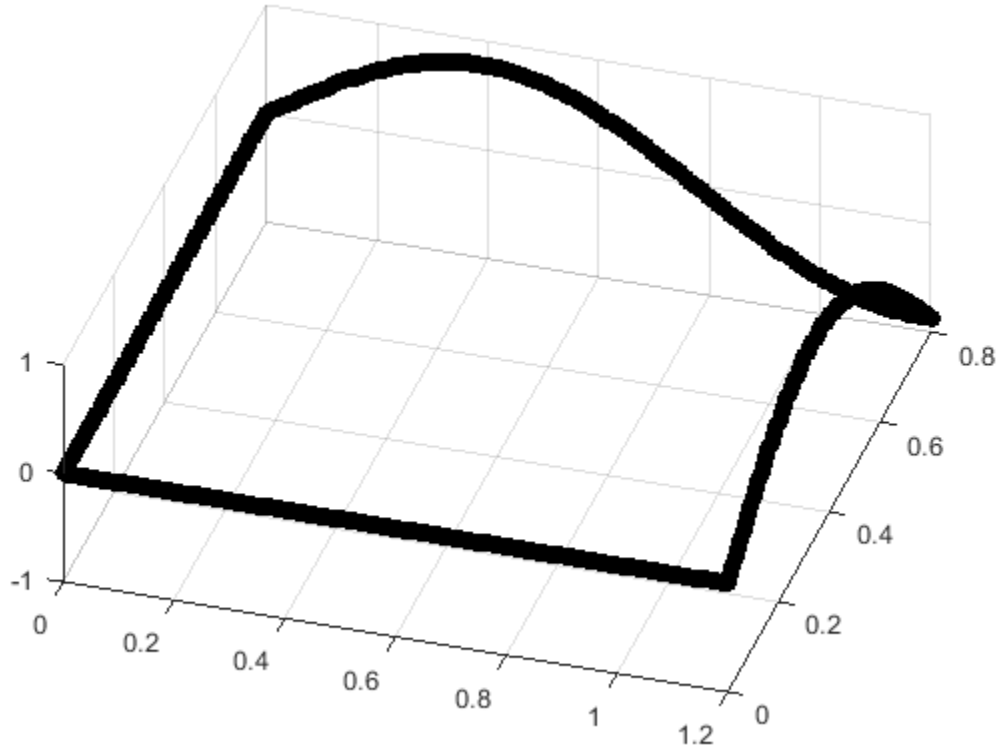


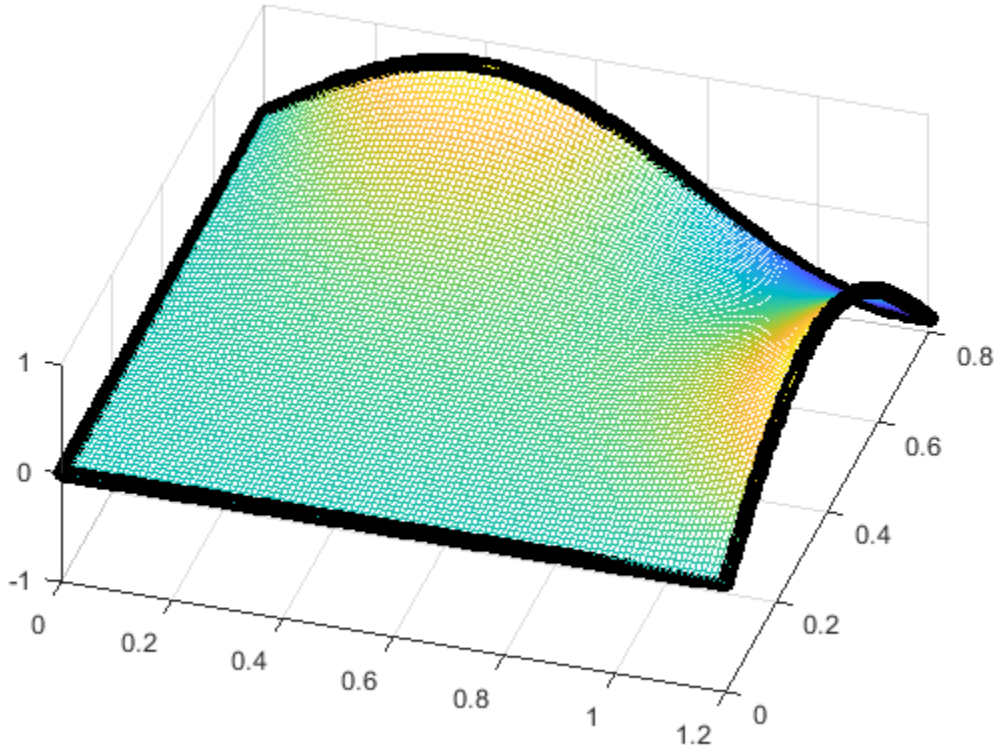


Wave equation in two dimensions

Problem 6.4: Use finite difference method to simulate the equilibrium solution for a vibrating membrane. Assume, that the membrane edges are fixed at the boundaries.

Solution: For dense mesh $M = 120$ and $N = 80$ we get





Wave equation in two dimensions

Problem 6.4: Use finite difference method to simulate the equilibrium solution for a vibrating membrane. Assume, that the membrane edges are fixed at the boundaries.

A few words about efficiency!

For $M = 120$ and $N = 80$

Crucial matrices are **A** (size 9204x9204, 677708928 Bytes) and **B** (size 9204x396, 29158272 Bytes)

```
MATLAB CODE EXAMPLE:

tic;
f(Pin)=A^(-1)*(B*Bnd);
time=toc
```

Time = 17.08 s

Inversion of the matrix is time-consuming

```
MATLAB CODE EXAMPLE:

tic;
f(Pin)=A\(B*Bnd);
time=toc
```

Time = 4.22 s

It is seldom necessary to form the explicit inverse of a matrix. A better way is to use the matrix backslash operator. This produces the solution using Gaussian elimination, without explicitly forming the inverse (see „mldivide”)

Machine: Intel(R) Core(TM) i7-2600K CPU @ 3.40 GHz 3.70 RAM: 16.0 GB

Wave equation in two dimensions

Problem 6.4: Use finite difference method to simulate the equilibrium solution for a vibrating membrane. Assume, that the membrane edges are fixed at the boundaries.

A few words about efficiency!

For $M = 120$ and $N = 80$

Using sparse matrices , we get **A** (size 9204x9204, 803688 Bytes) and **B** (size 9204x396, 9448 Bytes)

```
MATLAB CODE EXAMPLE:

tic;
f(Pin)=A^(-1)*(B*Bnd);
time=toc
```

Time = 7.84 s

```
MATLAB CODE EXAMPLE:

tic;
f(Pin)=A\(B*Bnd);
time=toc
```

Time = 0.16 s

Sparse matrices provide efficient storage of double or logical data that has a large percentage of zeroes. While full (or dense) matrices store every single element in memory regardless of value, sparse matrices store only the nonzero elements and their row indices. For this reason, using sparse matrices can significantly reduce the amount of memory required for data storage.

Machine: Intel(R) Core(TM) i7-2600K CPU @ 3.40 GHz 3.70 RAM: 16.0 GB

Modeling and Simulation of Systems

Lecture 7: Matrix operators and eigenvalue problems

Piotr Kowalczyk

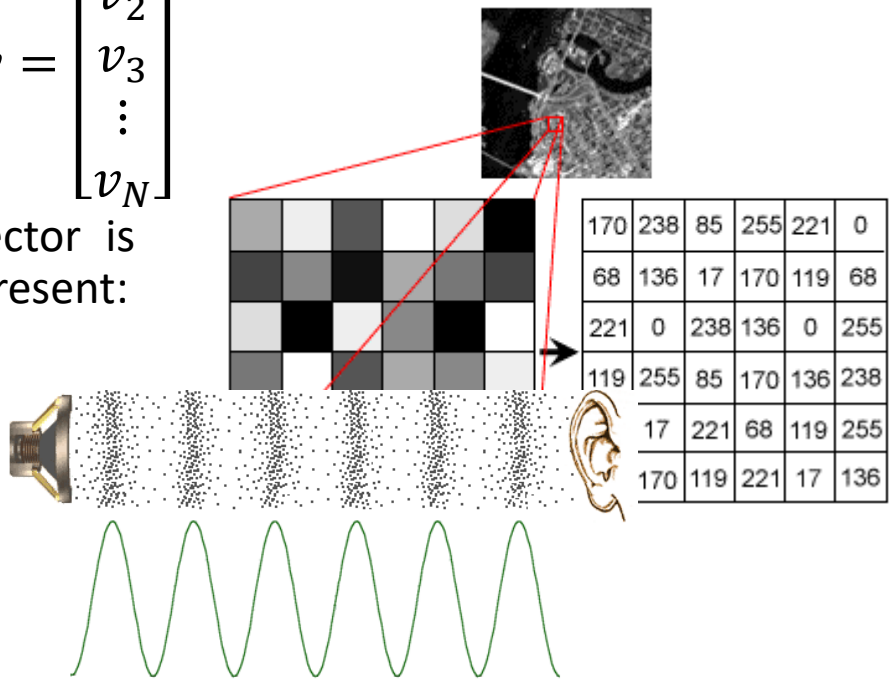
Examples of N-dimensional vectors

From mathematical point of view:

vectors are elements of a vector space (linear space), which may be added together and multiplied (scaled) by numbers (scalars)

If $\boldsymbol{v_1}, \boldsymbol{v_2} \in V$ and $\alpha, \beta \in R$, then $\alpha \boldsymbol{v_1} + \beta \boldsymbol{v_2} \in V$

$$v = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_N \end{bmatrix}$$



In practice (for computer science applications) N-dimensional vector is often treated as a set of N real (or complex) numbers, which can represent:

- sound waves (air pressure vs. time)
- Images, videos (pixel vs. position)
- other data storage (e.g. informations, measurements, ???)
- numerical analysis results (e.g. vibration, force, temperature)
- …

Source: http://www.mediacollege.com/audio/01/sound-waves.html, http://hosting.soonet.ca/eliris/remotesensing/bl130lec10.html

Matrix operators

Problem 6.1a: Use the matrix operator $\boldsymbol{S} = \begin{bmatrix} 10 & 0 \\ 0 & 2 \end{bmatrix}$ to modify a given set of points (located on xy plane).

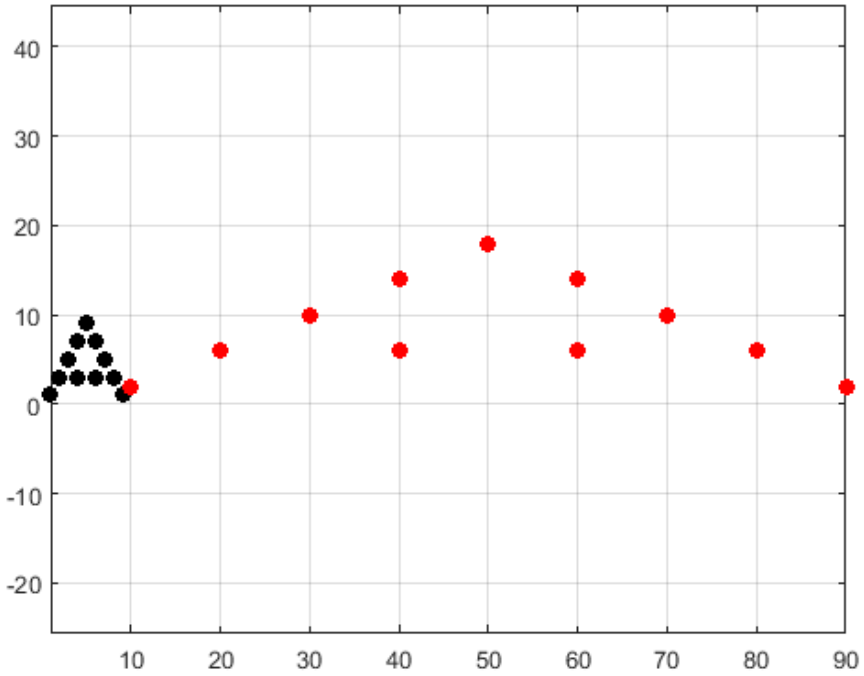
Solution: Any single point (x_0, y_0) can be represented by the following vector $\mathbf{P} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$. The multiplication of this vector and any square matrix (2×2) results in another vector from the same space. Hence,

$$\boldsymbol{P}' = \boldsymbol{SP} = \begin{bmatrix} 10 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} 10x_0 \\ 2y_0 \end{bmatrix}$$

In general

$$\begin{bmatrix} x'_1 & x'_2 & \dots & x'_P \\ y'_1 & y'_2 & \dots & y'_P \end{bmatrix} = \begin{bmatrix} 10 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \dots & x_P \\ y_1 & y_2 & \dots & y_P \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 4 & 6 \\ 1 & 3 & 5 & 7 & 9 & 7 & 5 & 3 & 1 & 3 & 3 \end{bmatrix}$$



Matrix operators

Problem 6.1b: Use the matrix operator $\boldsymbol{S} = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}$ to modify a given set of points (located on xy plane).

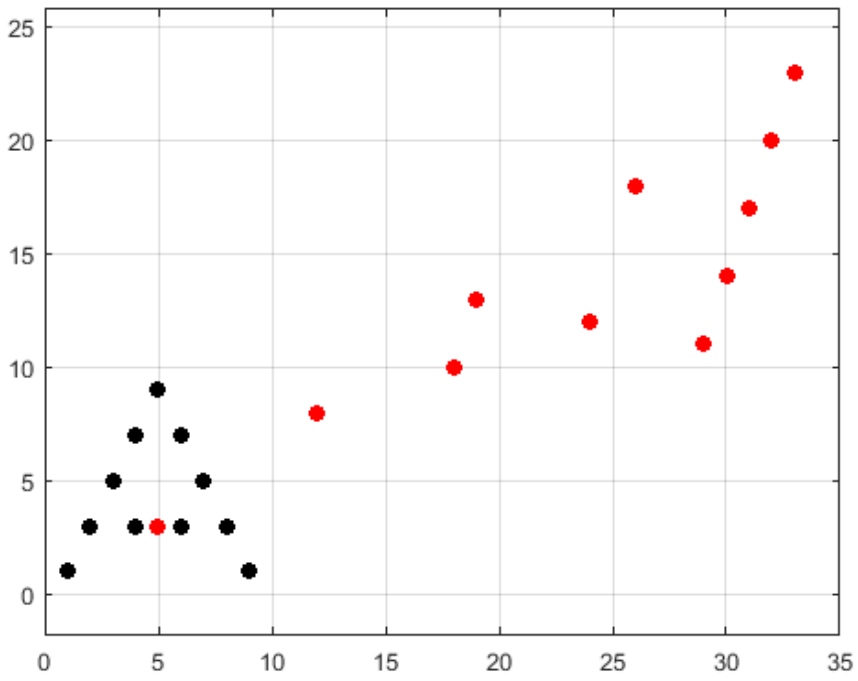
Solution: Any single point (x_0, y_0) can be represented by the following vector $\mathbf{P} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$. The multiplication of this vector and any square matrix (2×2) results in another vector from the same space. Hence,

$$\boldsymbol{P}' = \boldsymbol{SP} = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} 3x_0 + 2y_0 \\ 1x_0 + 2y_0 \end{bmatrix}$$

In general

$$\begin{bmatrix} x'_1 & x'_2 & \dots & x'_P \\ y'_1 & y'_2 & \dots & y'_P \end{bmatrix} = \begin{bmatrix} 10 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \dots & x_P \\ y_1 & y_2 & \dots & y_P \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 4 & 6 \\ 1 & 3 & 5 & 7 & 9 & 7 & 5 & 3 & 1 & 3 & 3 \end{bmatrix}$$

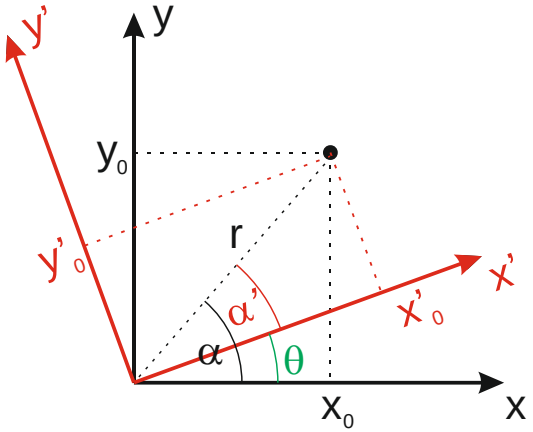


Matrix operators

Problem 6.2: Find a matrix operator which rotates a given set of points (located on xy plane).

Solution: For a given point (x_0, y_0) we can write

$$\begin{cases} x_0 = r\cos(\alpha) \\ y_0 = r\sin(\alpha) \end{cases}$$



In the new coordinates system (primed) the relations are almost the same

$$\begin{cases} x'_0 = r\cos(\alpha') \\ y'_0 = r\sin(\alpha') \end{cases}$$

For a fixed rotation angle θ the coordinates in both systems are related by

$$\begin{cases} x'_0 = r\cos(\alpha') = r\cos(\alpha-\theta) = r\cos(\alpha)\cos(\theta) + r\sin(\alpha)\sin(\theta) = x_0\cos(\theta) + y_0\sin(\theta) \\ y'_0 = r\sin(\alpha') = r\sin(\alpha-\theta) = r\sin(\alpha)\cos(\theta) - r\cos(\alpha)\sin(\theta) = y_0\cos(\theta) - x_0\sin(\theta) \end{cases}$$

or in the matrix notation

$$\begin{bmatrix} x'_0 \\ y'_0 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

rotation matrix!

Matrix operators

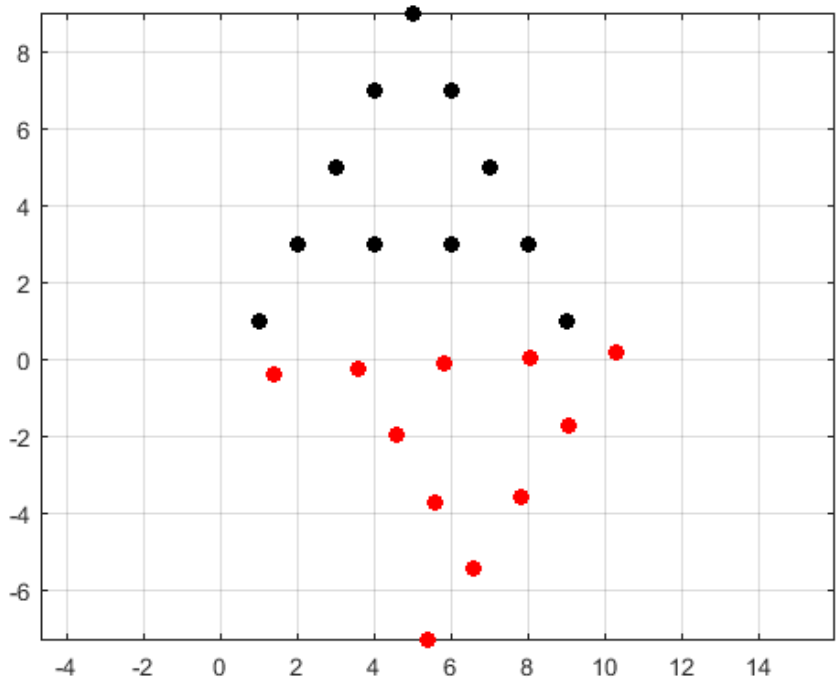
Problem 6.2: Find a matrix operator which rotates a given set of points (located on xy plane).

Solution: For instance if $\theta = -\frac{\pi}{3}$ the rotation matrix has the following form

$$\boldsymbol{R} = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$$

and any point represented by vector $\mathbf{P} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$ can be transformed into the new coordinates system.

$$\mathbf{P} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 4 & 6 \\ 1 & 3 & 5 & 7 & 9 & 7 & 5 & 3 & 1 & 3 & 3 \end{bmatrix}$$



In general

$$\begin{bmatrix} x'_1 & x'_2 & \dots & x'_P \\ y'_1 & y'_2 & \dots & y'_P \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \dots & x_P \\ y_1 & y_2 & \dots & y_P \end{bmatrix}$$

Matrix operators

Problem 6.3: Find a matrix operator which estimates the first derivative of a function (using forward difference)

Solution: Let us consider a function discretized in N points

$$f_n = f(x_n)$$

where $x_n = x_1 + (n-1)\Delta x$. For any point x_n we can estimate the derivative (forward difference)

$$\frac{df}{dx}(x_n) = f_n' \approx \frac{f_{n+1} - f_n}{\Delta x}$$

The function can be represented by a one column vector. Then

$$\boldsymbol{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ \vdots \\ f_{N-2} \\ f_{N-1} \\ f_N \end{bmatrix}$$

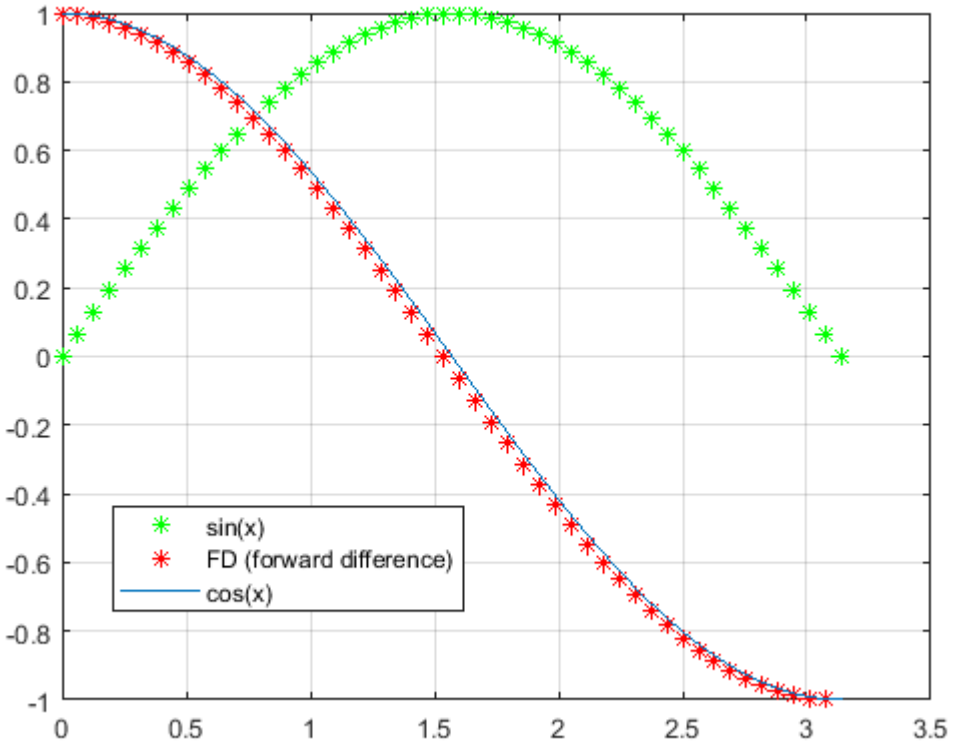
$$\begin{bmatrix} f'_1 \\ f'_2 \\ f'_3 \\ f'_4 \\ \vdots \\ f'_{N-2} \\ f'_{N-1} \\ f'_N \end{bmatrix} = \frac{1}{\Delta x} \begin{bmatrix} f_2 - f_1 \\ f_3 - f_2 \\ f_4 - f_3 \\ f_5 - f_4 \\ \vdots \\ f_{N-1} - f_{N-2} \\ f_N - f_{N-1} \\ -f_N \end{bmatrix} = \boxed{\frac{1}{\Delta x} \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & -1 \end{bmatrix}} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ \vdots \\ f_{N-2} \\ f_{N-1} \\ f_N \end{bmatrix}$$

Matrix operators

Problem 6.3: Find a matrix operator which estimates the first derivative of a function (using forward difference)

Solution: Let us use the obtained operator to the discretized function $f(x) = \sin(x)$ for $N = 50$ and $x \in [0, \pi]$.

```
MATLAB CODE EXAMPLE:

N=50;
x=linspace(0,pi,N)';
f=sin(x);
dx=x(2)-x(1);
D=(diag(-ones(N,1))+diag(ones(N-1,1),1))/dx;
Df=D*f;
plot(x,Df,'r*')
```



Matrix operators

Problem 6.4: Find a matrix operator which estimates the second derivative of a function (using central difference)

Solution: Again, let us consider a function discretized in N points

$$f_n = f(x_n)$$

where $x_n = x_1 + (n-1)\Delta x$. For any point x_n we can estimate the second derivative (central difference)

$$\frac{d^2 f}{dx^2}(x_n) = f_n'' \approx \frac{f_{n+1} - 2f_n + f_{n-1}}{\Delta x^2}$$

The function can be represented by a one column vector. Then

$$\boldsymbol{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ \vdots \\ f_{N-2} \\ f_{N-1} \\ f_N \end{bmatrix}$$

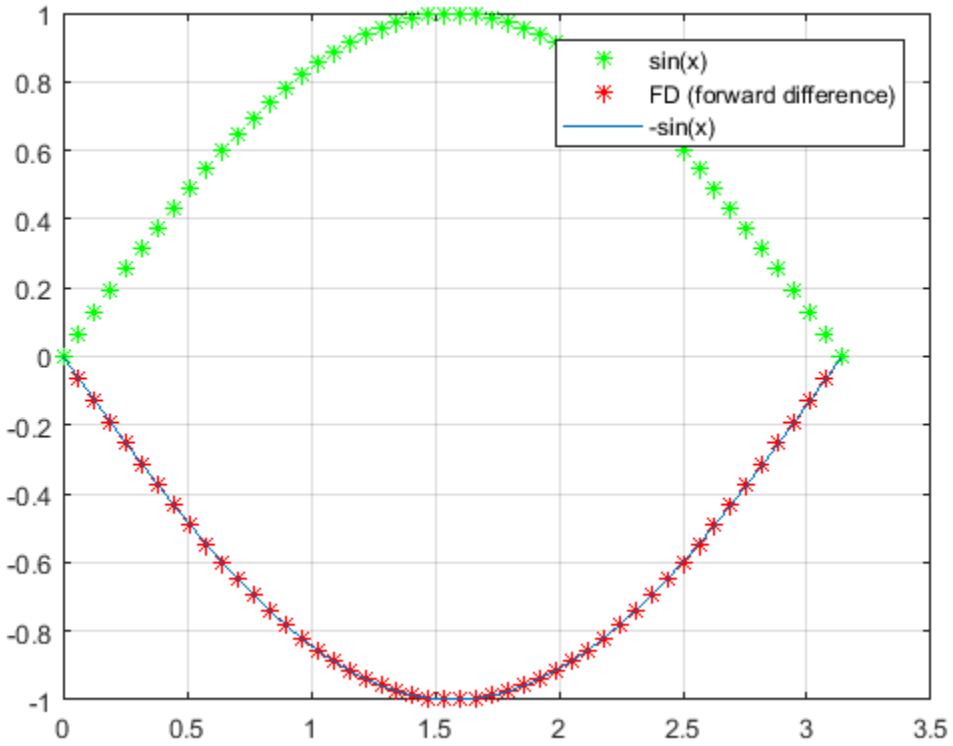
$$\begin{bmatrix} f''_1 \\ f''_2 \\ f''_3 \\ f''_4 \\ \vdots \\ f''_{N-2} \\ f''_{N-1} \\ f''_N \end{bmatrix} = \frac{1}{\Delta x^2} \begin{bmatrix} f_2 - 2f_1 \\ f_3 - 2f_2 + f_1 \\ f_4 - 2f_3 + f_2 \\ f_5 - 2f_4 + f_3 \\ \vdots \\ f_{N-1} - 2f_{N-2} + f_{N-3} \\ f_N - 2f_{N-1} + f_{N-2} \\ -2f_N + f_{N-1} \end{bmatrix} = \boxed{\frac{1}{\Delta x^2} \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & -2 \end{bmatrix}} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ \vdots \\ f_{N-2} \\ f_{N-1} \\ f_N \end{bmatrix}$$

Matrix operators

Problem 6.4: Find a matrix operator which estimates the second derivative of a function (using central difference)

Solution: Let us use the obtained operator to the discretized function $f(x) = \sin(x)$ for $N = 50$ and $x \in [0, \pi]$.

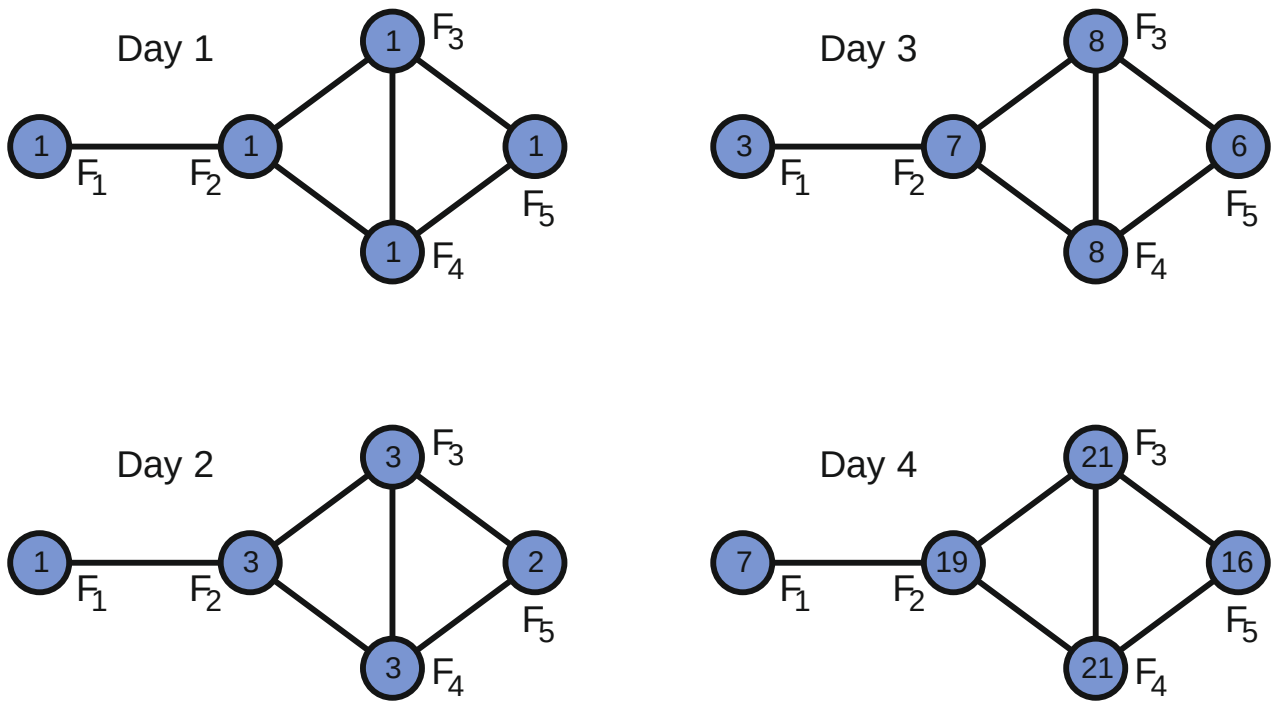
```
MATLAB CODE EXAMPLE:

N=50;
x=linspace(0,pi,N)';
f=sin(x);
dx=x(2)-x(1);
tmp=ones(N,1);
tmp1=ones(N-1,1);
D=(-2*diag(tmp)+diag(tmp1,1)+diag(tmp1,-1))/dx^2;
Df=D*f;
plot(x,Df,'r*')
```



Matrix operators

Problem 6.5: Find a matrix operator which describes number of comments made by a group of friends on social media in one day. Assume that each comment is commented each day.

Solution: Let us consider the graph below. We assume that in the first day each member of the group wrote one post.



Simple adjacency matrix

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

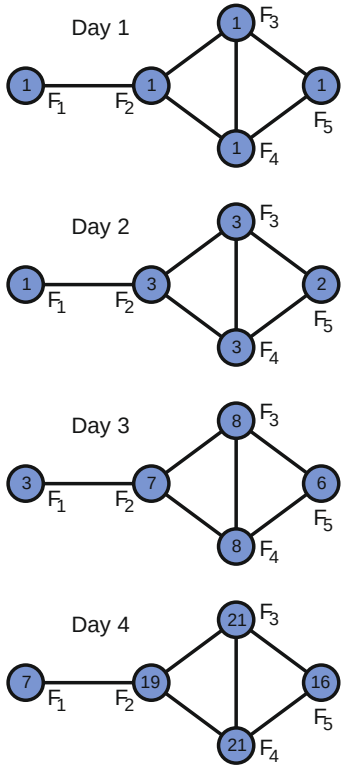
$$\boldsymbol{x}^{(1)} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \boldsymbol{x}^{(2)} = \boldsymbol{A}\boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 3 \\ 3 \\ 3 \\ 2 \end{bmatrix}$$

$$\boldsymbol{x}^{(3)} = \boldsymbol{A}\boldsymbol{x}^{(2)} = \begin{bmatrix} 3 \\ 7 \\ 8 \\ 8 \\ 6 \end{bmatrix} \qquad \boldsymbol{x}^{(4)} = \boldsymbol{A}\boldsymbol{x}^{(3)} = \begin{bmatrix} 7 \\ 19 \\ 21 \\ 21 \\ 16 \end{bmatrix}$$

Matrix operators

Problem 6.5: Find a matrix operator which describes number of comments made by a group of friends on social media in one day. Assume that each comment is commented each day.

Solution: Let us consider the graph below. We assume that in the first day each member of the group wrote one post.



number of comments

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 20
F1	1	1	3	7	19	49	131	343	911	2397	39655527
F2	1	3	7	19	49	131	343	911	2397	6347	104742659
F3	1	3	8	21	56	147	390	1027	2718	7169	118490229
F4	1	3	8	21	56	147	390	1027	2718	7169	118490229
F5	1	2	6	16	42	112	294	780	2054	5436	89727304

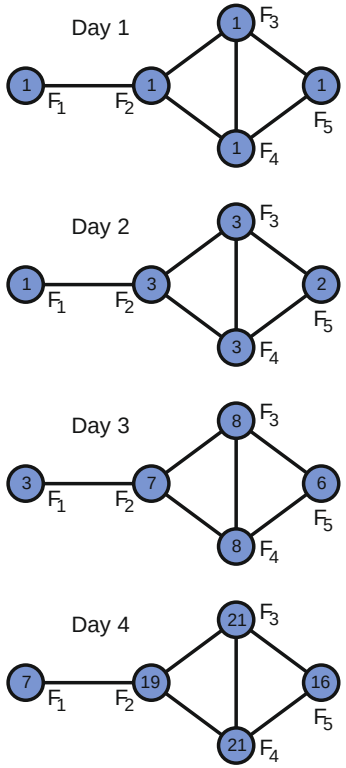
ratio of comments

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 20
F1	--	1.000	3.000	2.333	2.714	2.579	2.673	2.618	2.656	2.631	2.641
F2	--	3.000	2.333	2.714	2.579	2.673	2.618	2.656	2.631	2.648	2.641
F3	--	3.000	2.667	2.625	2.667	2.625	2.653	2.633	2.647	2.638	2.641
F4	--	3.000	2.667	2.625	2.667	2.625	2.653	2.633	2.647	2.638	2.641
F5	--	2.000	3.000	2.667	2.625	2.667	2.625	2.653	2.633	2.647	2.641

Matrix operators

Problem 6.5: Find a matrix operator which describes number of comments made by a group of friends on social media in one day. Assume that each comment is commented each day.

Solution: Let us consider the graph below. We assume that in the first day each member of the group wrote one post.



$$\boldsymbol{x}^{(n+1)} = \boldsymbol{A}\boldsymbol{x}^{(n)} \xrightarrow{n \gg 1} \boldsymbol{x}^{(n+1)} = 2.641\boldsymbol{x}^{(n)}$$

$$\boldsymbol{A}\boldsymbol{x}^{(n)} = 2.641\boldsymbol{x}^{(n)}$$

$$\boldsymbol{x}^{(20)} = \boxed{\begin{bmatrix} 39655527 \\ 104742659 \\ 118490229 \\ 118490229 \\ 89727304 \end{bmatrix}} = \left\|\boldsymbol{x}^{(20)}\right\| \boxed{\begin{bmatrix} 0.1797 \\ 0.4748 \\ 0.5371 \\ 0.5371 \\ 0.4067 \end{bmatrix}}$$

Eigenvector centrality is a measure of the influence of a node in a network. It assigns relative scores to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes.

Google's PageRank and the Katz centrality are variants of the eigenvector centrality.

Eigenvalues and eigenvectors of matrices

Let us assume that $\boldsymbol{A}$ is $N \times N$ matrix. There is a vector $\boldsymbol{x}$ that, when operated on by $\boldsymbol{A}$, returns a vector that points in the same direction (up to the sign of the vector) with a length scaled by the constant λ

$$\boxed{\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}}$$

An eigenvector is determined up to a multiplicative constant!

$\boldsymbol{x}$ – eigenvector

λ – eigenvalue

$N \times N$ matrix has N eigenvalues λ_n and N eigenvectors $\boldsymbol{x_n}$

Set of all eigenvalues $\{\lambda_n\}$ is called a spectrum of the operator

Eigenvalues and eigenvectors of matrices

Eigenvectors and eigenvalues of the operator (matrix) provide a lot of information about it properties!

How?

If the eigenvectors and eigenvalues of the operator are known we can expect the result of the application of this operator to a different vector (without direct calculations)!

If the vector „is similar” to one of the eigenvectors, then we know that the operator does not change the vector much - the same direction but scaled, due to corresponding eigenvalue.

Applications:
- data compression
- image processing
- efficient numerical techniques

Eigenvalues and eigenvectors of matrices

Problem 6.6: : Find eigenvectors and eigenvalues of a 3×3 matrix

$$\boldsymbol{A} = \begin{bmatrix} 57 & 0 & -24 \\ 0 & 2 & 0 \\ -24 & 0 & 43 \end{bmatrix}$$

Solution: Using Matlab function „eig" we get

$$\boldsymbol{x_1} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \lambda_1 = 2, \boldsymbol{x_2} = \begin{bmatrix} -0.6 \\ 0 \\ -0.8 \end{bmatrix}, \lambda_2 = 25,$$

$$\boldsymbol{x_3} = \begin{bmatrix} -0.8 \\ 0 \\ 0.6 \end{bmatrix}, \lambda_3 = 25$$

```
MATLAB CODE EXAMPLE:

A=[ 57 0 -24 ; 0 2 0 ; -24 0 43 ];
[x,l]=eig(A)

test1=max(abs(A*x(:,1)-l(1,1)*x(:,1)))
test2=max(abs(A*x(:,2)-l(2,2)*x(:,2)))
test3=max(abs(A*x(:,3)-l(3,3)*x(:,3)))

RESULT:

x =       0   -0.6000   -0.8000
    1.0000         0         0
         0   -0.8000    0.6000

l =  2     0     0
     0    25     0
     0     0    75

test1 = 0
test2 = 3.5527e-15
test3 = 0
```

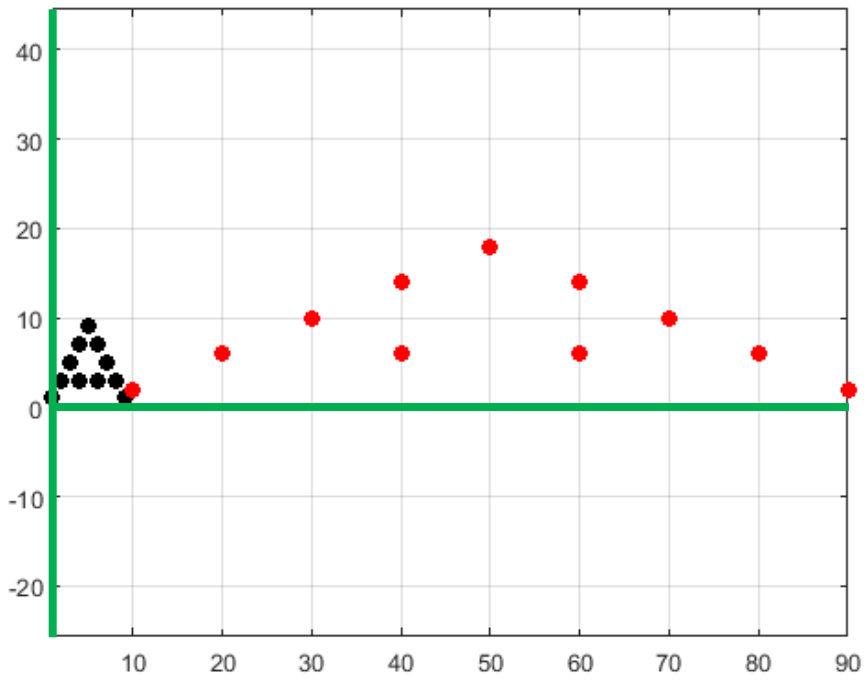
Eigenvalues and eigenvectors of matrices

Problem 6.7a: Find eigenvectors and eigenvalues of matrix operator $\boldsymbol{S} = \begin{bmatrix} 10 & 0 \\ 0 & 2 \end{bmatrix}$ which was considered in problem 6.1a.

Solution: For such simple case the eigensolver is unnecessary

$$\boldsymbol{x_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \lambda_1 = 10 \ \rightarrow \boldsymbol{S}\boldsymbol{x_1} = \begin{bmatrix} 10 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \end{bmatrix} = 10 \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\boldsymbol{x_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \lambda_2 = 2 \ \rightarrow \boldsymbol{S}\boldsymbol{x_2} = \begin{bmatrix} 10 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} = 2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



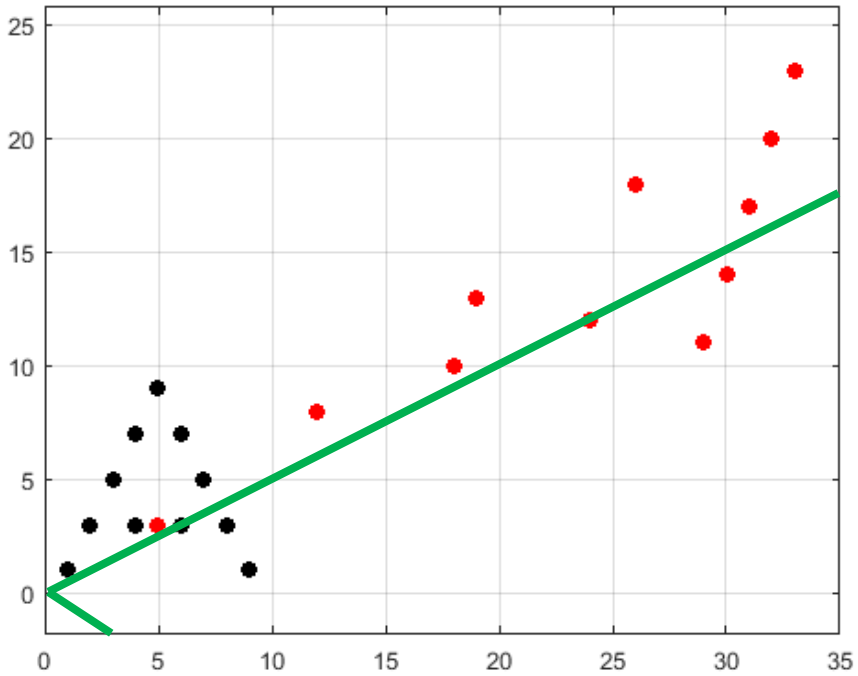
Eigenvalues and eigenvectors of matrices

Problem 6.7b: Find eigenvectors and eigenvalues of matrix operator $\boldsymbol{S} = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}$ which was considered in problem 6.1b.

Solution: The eigensolver can be helpful

$$\boldsymbol{x_1} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \lambda_1 = 4 \ \rightarrow \boldsymbol{S}\boldsymbol{x_1} = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \end{bmatrix} = 4 \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\boldsymbol{x_2} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \lambda_2 = 1 \ \rightarrow \boldsymbol{S}\boldsymbol{x_2} = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 1 \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$



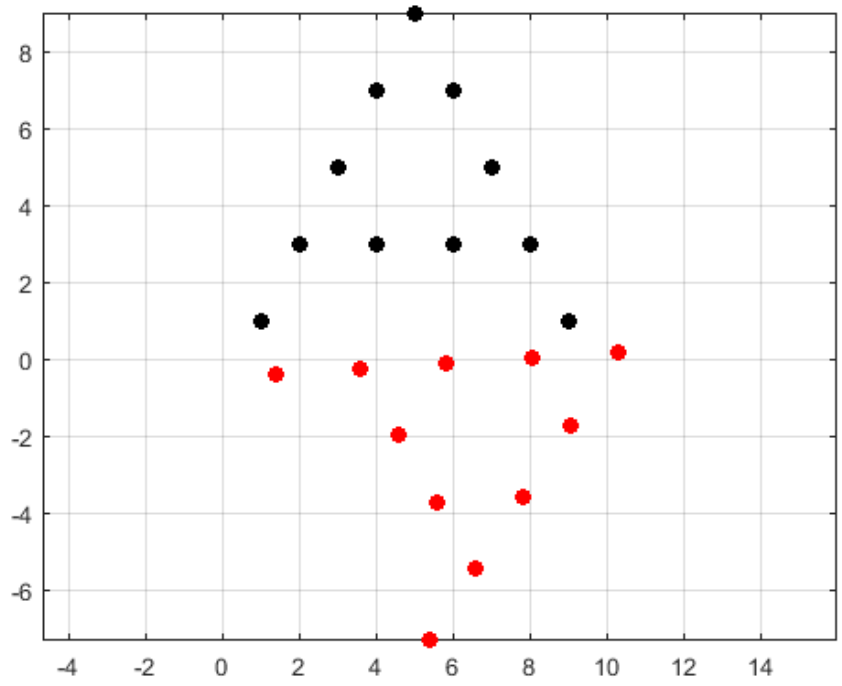
Eigenvalues and eigenvectors of matrices

Problem 6.8: Find eigenvectors and eigenvalues of rotation matrix $\boldsymbol{R} = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$ which was considered in problem 6.2.

Solution: Using Matlab function „eig" we get

$$\boldsymbol{x_1} = \begin{bmatrix} 0.7071 + 0.0000i \\ 0.0000 - 0.7071i \end{bmatrix}, \lambda_1 = 0.5000 + 0.8660i$$

$$\boldsymbol{x_2} = \begin{bmatrix} 0.7071 + 0.0000i \\ 0.0000 + 0.7071i \end{bmatrix}, \lambda_2 = 0.5000 - 0.8660i$$



Eigenvalues and eigenvectors of matrices

Problem 6.9: Find eigenvectors and eigenvalues of the operator representing the second derivative of a function (using central difference) which was considered in problem 6.4 for $N = 50$ and $x \in [0, \pi]$.

$$\boldsymbol{D} = \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & -2 \end{bmatrix}$$

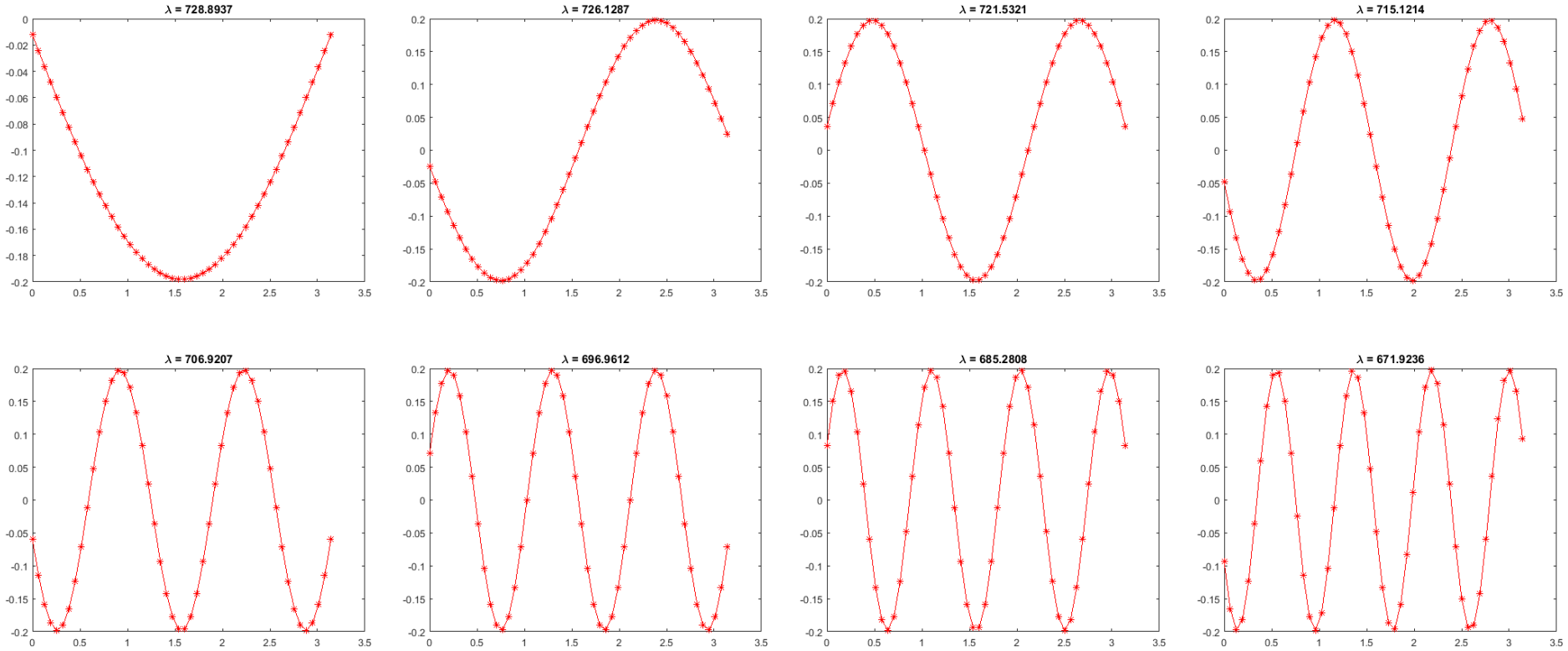
Solution: Using Matlab function „eig" we get $N = 50$ eigenvectors and $N = 50$ corresponding eigenvalues.

```
MATLAB CODE EXAMPLE:

N=50;
x=linspace(0,pi,N)';
dx=x(2)-x(1);
tmp=ones(N,1);
tmp1=ones(N-1,1);
D=(-2*diag(tmp)+diag(tmp1,1)+diag(tmp1,-1))/dx^2;

[X,L]=eig(D);

for i=1:N
    L(i,i)
    plot(x,X(:,i),'r*-')
    title(['\lambda = ',num2str(L(i,i))])
    pause
end
```

Eigenvalues and eigenvectors of matrices

Problem 6.9: Find eigenvectors and eigenvalues of the operator representing the second derivative of a function (using central difference) which was considered in problem 6.4 for $N = 50$ and $x \in [0, \pi]$.

Solution: 8 examples of eigenvectors and corresponding eigenvalues.



Eigenvalues and eigenvectors of matrices

Problem 6.10: Find eigenvectors and eigenvalues of adjacency matrix $\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$ which was considered in problem 6.5.

Solution: Using Matlab function „eig" we get

```
RESULT:

X = 0.3549    0.0000   -0.5989   -0.6950    0.1797
   -0.6301   -0.0000    0.3529   -0.5030    0.4747
    0.3820   -0.7071    0.1955    0.1655    0.5371
    0.3820    0.7071    0.1955    0.1655    0.5371
   -0.4303         0   -0.6636    0.4573    0.4067


L =-1.7757         0         0         0         0
         0   -1.0000         0         0         0
         0         0   -0.5892         0         0
         0         0         0    0.7237         0
         0         0         0         0    2.6412
```

$$\boldsymbol{A}\boldsymbol{x}^{(20)} = 2.641\boldsymbol{x}^{(20)}$$

$$\boldsymbol{x}^{(20)} = \begin{bmatrix} 39655527 \\ 104742659 \\ 118490229 \\ 118490229 \\ 89727304 \end{bmatrix} = \left\|\boldsymbol{x}^{(20)}\right\| \begin{bmatrix} 0.1797 \\ 0.4748 \\ 0.5371 \\ 0.5371 \\ 0.4067 \end{bmatrix}$$

The adjacency matrix of an undirected simple graph is symmetric, and therefore has a complete set of real eigenvalues and an orthogonal eigenvector basis. The set of eigenvalues of a graph is the <u>spectrum of the graph</u>.

System of equations & matrix operator

Let us assume that $\boldsymbol{a}, \boldsymbol{b} \in R^N$ and $\boldsymbol{A}$ is $N \times N$ matrix

$$\boxed{\boldsymbol{A}\boldsymbol{a} = \boldsymbol{b}}$$

$\boldsymbol{a}$-vector
$\boldsymbol{b}$-image of the vector $\boldsymbol{a}$
$\boldsymbol{A}$-matrix operator

Finding the vector if its image is known

$\boldsymbol{a}$-set of unknown variables
$\boldsymbol{b}$-set of known variables
$\boldsymbol{A}$-represents system of equations

Solving systems of equations

For instance

$$\begin{bmatrix} 1 & 2 & -1 \\ 2 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$$

$$\begin{cases} x + 2y - z = 1 \\ 2x - y + z = 3 \\ x + y - z = 0 \end{cases}$$

Solution

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\begin{cases} x = 1 \\ y = 2 \\ z = 3 \end{cases}$$

System of equations & matrix operator

Nonhomogeneous Systems of Linear Equations

$$\boldsymbol{Aa} = \boldsymbol{b}$$

One of the options:

- single unique solution

$$\begin{cases} x + y = 3 \\ x - y = -1 \end{cases}$$

- infinitely many solutions

$$\begin{cases} x + y = 3 \\ 2x + 2y = 6 \end{cases}$$

- no solution

$$\begin{cases} x + y = 3 \\ x + y = 2 \end{cases}$$

Homogeneous Systems of Linear Equations

$$\boldsymbol{Aa} = \boldsymbol{0}$$

Has always zero solution ($x = 0, \mathrm{y} = 0$), if the equations are linearly dependent infinitely many solutions also.

The problem is how to find the nonzero solution?

Find the eigenvector corresponding to the smallest (zero) eigenvalue

System of equations & matrix operator

Problem 6.11: Suppose, in a small country town called Matrixville, there lived a Farmer, a Tailor, a Carpenter, a Coal Miner and Slacker Bob. The Farmer produced food; the Tailor, clothes; the Carpenter, housing; the Coal Miner supplied energy; and Slacker Bob made high-quality hooch, half of which he drank himself. Assumptions: everyone buys from and sells to the central pool (i.e. there is no outside supply and demand), everything produced is consumed. Moreover, we must specify what fraction of each of the goods is consumed by each person in our town – see the table. How should we price the goods in order for the citizens of Matrixville?

	Food	Clothes	Housing	Energy	Hooch
Farmer	0.25	0.15	0.25	0.18	0.2
Tailor	0.15	0.28	0.18	0.17	0.05
Carpenter	0.22	0.19	0.22	0.22	0.1
Coal Miner	0.2	0.15	0.2	0.28	0.15
Slacker Bob	0.18	0.23	0.15	0.15	0.5

Solution: Now, let $p_F, p_T, p_C, p_{CM}, p_{SB}$ denote the incomes of the Farmer, Tailor, Carpenter, Coal Miner and Slacker Bob, respectively.

Note that each of these quantities not only denotes the incomes of each of our esteemed citizens, but also the cost of the corresponding goods. So for example, p_F **is the Farmer's income as well as the cost of all the food**. So if the Farmer produces \$100 worth of food, then his income will also be \$100 since all of this food is bought out and the profits go to the Farmer.

System of equations & matrix operator

Problem 6.11: Suppose, in a small country town called Matrixville, there lived a Farmer, a Tailor, a Carpenter, a Coal Miner and Slacker Bob. The Farmer produced food; the Tailor, clothes; the Carpenter, housing; the Coal Miner supplied energy; and Slacker Bob made high-quality hooch, half of which he drank himself. Assumptions: everyone buys from and sells to the central pool (i.e. there is no outside supply and demand), everything produced is consumed. Moreover, we must specify what fraction of each of the goods is consumed by each person in our town – see the table. How should we price the goods in order for the citizens of Matrixville?

	Food	Clothes	Housing	Energy	Hooch
Farmer	0.25	0.15	0.25	0.18	0.2
Tailor	0.15	0.28	0.18	0.17	0.05
Carpenter	0.22	0.19	0.22	0.22	0.1
Coal Miner	0.2	0.15	0.2	0.28	0.15
Slacker Bob	0.18	0.23	0.15	0.15	0.5

Solution: Under the model assumptions, we can write

$$\begin{cases} 0.25p_F + 0.15p_T + 0.25p_C + 0.18p_{CM} + 0.20p_{SB} = p_F \\ 0.15p_F + 0.28p_T + 0.18p_C + 0.17p_{CM} + 0.05p_{SB} = p_T \\ 0.22p_F + 0.19p_T + 0.22p_C + 0.22p_{CM} + 0.10p_{SB} = p_C \\ 0.20p_F + 0.15p_T + 0.20p_C + 0.28p_{CM} + 0.15p_{SB} = p_{CM} \\ 0.18p_F + 0.23p_T + 0.15p_C + 0.15p_{CM} + 0.50p_{SB} = p_{SB} \end{cases}$$

Or in a homogeneous form

$$\begin{cases} -0.75p_F + 0.15p_T + 0.25p_C + 0.18p_{CM} + 0.20p_{SB} = 0 \\ 0.15p_F - 0.72p_T + 0.18p_C + 0.17p_{CM} + 0.05p_{SB} = 0 \\ 0.22p_F + 0.19p_T - 0.78p_C + 0.22p_{CM} + 0.10p_{SB} = 0 \\ 0.20p_F + 0.15p_T + 0.20p_C - 0.72p_{CM} + 0.15p_{SB} = 0 \\ 0.18p_F + 0.23p_T + 0.15p_C + 0.15p_{CM} - 0.50p_{SB} = 0 \end{cases}$$

System of equations & matrix operator

Problem 6.11: Suppose, in a small country town called Matrixville, there lived a Farmer, a Tailor, a Carpenter, a Coal Miner and Slacker Bob. The Farmer produced food; the Tailor, clothes; the Carpenter, housing; the Coal Miner supplied energy; and Slacker Bob made high-quality hooch, half of which he drank himself. Assumptions: everyone buys from and sells to the central pool (i.e. there is no outside supply and demand), everything produced is consumed. Moreover, we must specify what fraction of each of the goods is consumed by each person in our town – see the table. How should we price the goods in order for the citizens of Matrixville?

	Food	Clothes	Housing	Energy	Hooch
Farmer	0.25	0.15	0.25	0.18	0.2
Tailor	0.15	0.28	0.18	0.17	0.05
Carpenter	0.22	0.19	0.22	0.22	0.1
Coal Miner	0.2	0.15	0.2	0.28	0.15
Slacker Bob	0.18	0.23	0.15	0.15	0.5

Solution: To solve such system we have to find the eigenvector corresponding to the smallest eigenvalue. Using „eig" function we get the following spectrum:

$$\{-0.0000 \quad -0.7006 \quad -1.0092 \quad -0.8377 \quad -0.9226\}$$

The eigenvector corresponding to the smallest eigenvalue represents the results

$$p_F = 0.4584,\ p_T = 0.3381,\ p_C = 0.4060, p_{CM} = 0.4295,\ p_{SB} = 0.5712$$

Vector basis

A set of vectors $\boldsymbol{v_1}, \boldsymbol{v_2}, \dots, \boldsymbol{v_N} \in \mathbb{V}$ is called a basis if the vectors are <u>linearly independent</u> and every vector in the vector space $\boldsymbol{v} \in \mathbb{V}$ is a linear combination of this set

$$\boldsymbol{w} = \alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} + \cdots + \alpha_N \boldsymbol{v_N}$$

the coefficients are referred to as <u>vector coordinates or components</u> in this space. A vector space can have several distinct sets of basis vectors; however each such set has the same number of elements, with this number being the <u>dimension</u> of the vector space.

Example – standard basis (also called natural basis):

$$\boldsymbol{e_1} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \boldsymbol{e_2} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \dots, \boldsymbol{e_N} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

$$\boldsymbol{w} = \alpha_1 \boldsymbol{e_1} + \alpha_2 \boldsymbol{e_2} + \cdots + \alpha_N \boldsymbol{e_N} = \alpha_1 \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + \alpha_N \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix}$$

Vector basis

A set of vectors $\boldsymbol{v_1}, \boldsymbol{v_2}, \dots, \boldsymbol{v_N} \in \mathbb{V}$ are <u>linearly independent</u>, if there exist scalars $\alpha_1, \alpha_2, \dots, \alpha_N$ not all zero, such that

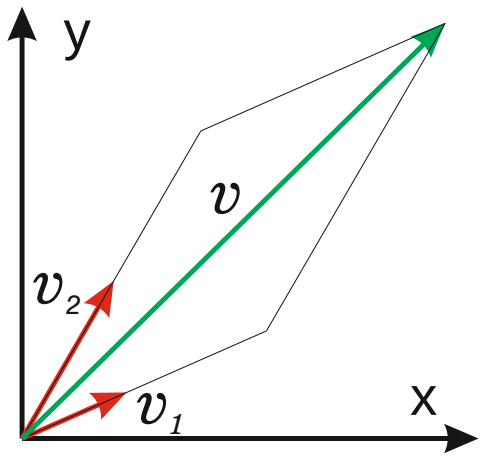
$$\alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2} + \cdots + \alpha_N \boldsymbol{v_N} = \boldsymbol{0}$$

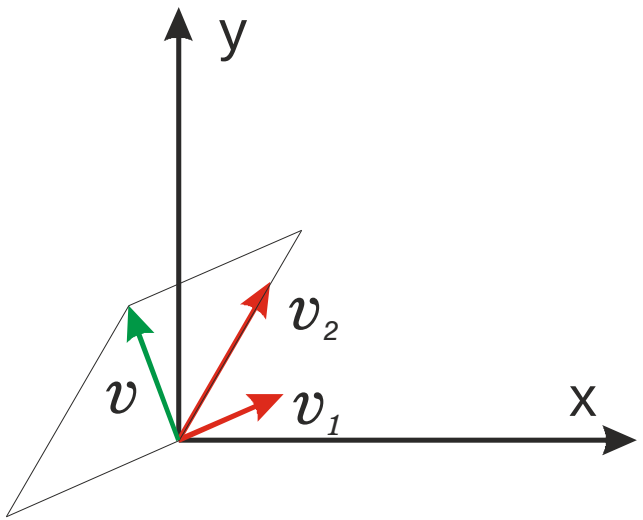
This implies that at least one of the scalars is nonzero, say $\alpha_1 \neq 0$, so we can rewrite the above expression as

$$\boldsymbol{v_1} = -\frac{\alpha_2}{\alpha_1}\boldsymbol{v_2} - \cdots - \frac{\alpha_N}{\alpha_1}\boldsymbol{v_N}$$

Which means that vector $\boldsymbol{v_1}$ can be expressed as a linear combination of the remaining vectors, so this vector cannot add any new direction (information) to the basis. Only linearly independent vectors create a basis!!!

2D Example: any vector $\boldsymbol{v}$ can be expressed as $\boldsymbol{v} = \alpha_1 \boldsymbol{v_1} + \alpha_2 \boldsymbol{v_2}$
if $\boldsymbol{v_1}, \boldsymbol{v_2}$ are not collinear (not on one line)





In case of 3D vectors $\boldsymbol{v_1}, \boldsymbol{v_2}, \boldsymbol{v_3}$ must not lie in one plane or on one line.

Vector basis

Problem 6.12: Find coordinates of the vector $\boldsymbol{w} = \begin{bmatrix} 1 \\ -5/4 \\ 2 \end{bmatrix} = \alpha_1 \boldsymbol{e_1} + \alpha_2 \boldsymbol{e_2} + \alpha_3 \boldsymbol{e_3}$ (represented in the standard basis) in the new following basis: $\boldsymbol{v_1} = \begin{bmatrix} 1/2 \\ 1/4 \\ 0 \end{bmatrix}, \boldsymbol{v_2} = \begin{bmatrix} 3/4 \\ 5/4 \\ 0 \end{bmatrix}, \boldsymbol{v_3} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Solution: Directly

$$\boldsymbol{w} = \begin{bmatrix} 1 \\ -5/4 \\ 2 \end{bmatrix} = \beta_1 \boldsymbol{v_1} + \beta_2 \boldsymbol{v_2} + \beta_3 \boldsymbol{v_3} = \begin{bmatrix} \frac{1}{2}\beta_1 + \frac{3}{4}\beta_2 \\ \frac{1}{4}\beta_1 + \frac{5}{4}\beta_2 \\ \beta_3 \end{bmatrix}$$

Then by solving simple system of equation

$$\begin{cases} \beta_1 = 5 \\ \beta_2 = -2 \\ \beta_3 = 2 \end{cases} \longrightarrow \boldsymbol{w} = \begin{bmatrix} 5 \\ -2 \\ 2 \end{bmatrix}_{in\ the\ new\ basis}$$

In general

$$\boldsymbol{w} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = [\boldsymbol{v_1}|\boldsymbol{v_2}|\boldsymbol{v_3}] \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}_{in\ the\ new\ basis}$$

Vector basis - subspaces

In many cases the size of the vector space can be significantly reduced. Let us imagine that we have to perform some specific operation $\boldsymbol{A}_{3\times3}$ on a huge set of vectors $\boldsymbol{w} = \alpha_1\boldsymbol{e_1} + \alpha_2\boldsymbol{e_2} + \alpha_3\boldsymbol{e_3}$ from 3D space (e.g. perform a projection or rotation) and we know that all the vectors $\boldsymbol{w}$ are located on a single plane

$$\boldsymbol{w}' = \boldsymbol{A}\boldsymbol{w}$$

So we can replace the standard basis the profit from such an operation is rather insignificant of two vectors $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ also located on that plane (the third vector $\boldsymbol{v_3}$ does not matter, as long as the set is linearly independent).

For all vectors from the considered (3D) set the third component is zero, so can be neglected – these vectors are located in a specific subspace (2D)

$$\boldsymbol{w} = \begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix} = [\boldsymbol{v_1}|\boldsymbol{v_2}|\boldsymbol{v_3}]\begin{bmatrix}\beta_1\\ \beta_2\\ 0\end{bmatrix}_{in\ the\ new\ basis} = [\boldsymbol{v_1}|\boldsymbol{v_2}]\begin{bmatrix}\beta_1\\ \beta_1\end{bmatrix}_{sub}$$

Then the operation $\boldsymbol{w}' = \boldsymbol{A}\boldsymbol{w}$ reduces to

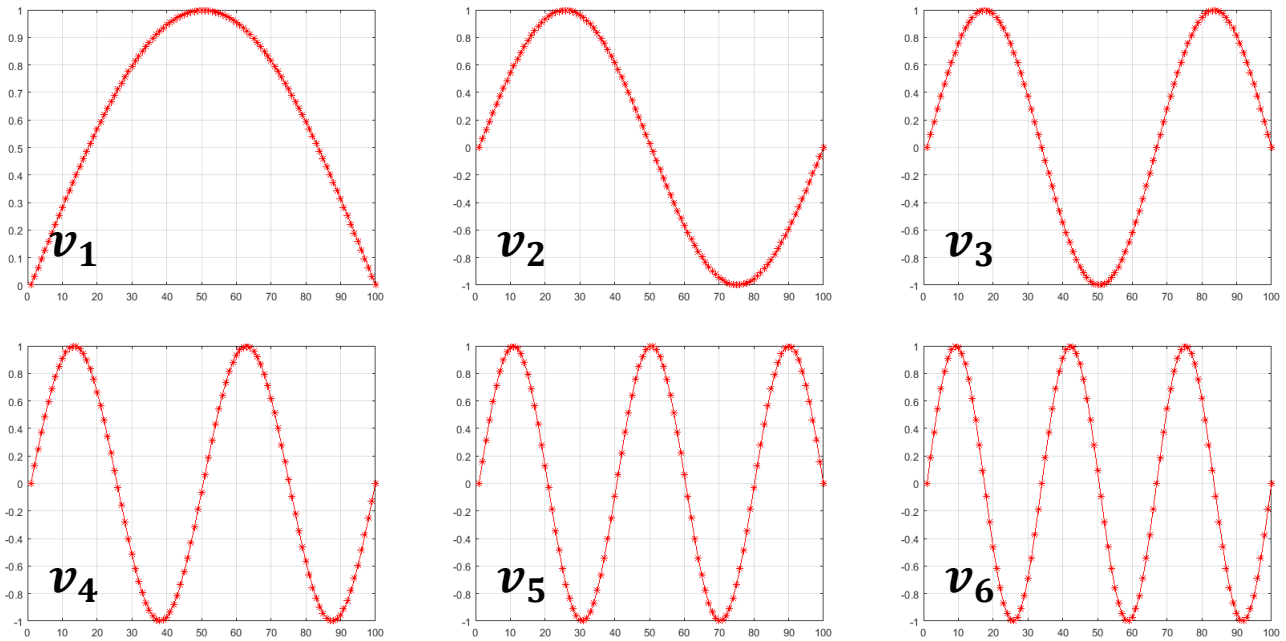
$$\begin{bmatrix}\alpha'_1\\ \alpha'_2\\ \alpha'_3\end{bmatrix} = \boldsymbol{A}\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix} \longrightarrow [\boldsymbol{v_1}|\boldsymbol{v_2}]\begin{bmatrix}\beta'_1\\ \beta'_1\end{bmatrix}_{sub} = \boldsymbol{A}[\boldsymbol{v_1}|\boldsymbol{v_2}]\begin{bmatrix}\beta_1\\ \beta_1\end{bmatrix}_{sub} \longrightarrow \begin{bmatrix}\beta'_1\\ \beta'_1\end{bmatrix}_{sub} = \underbrace{\left([\boldsymbol{v_1}|\boldsymbol{v_2}]^{\boldsymbol{T}}[\boldsymbol{v_1}|\boldsymbol{v_2}]\right)^{-\boldsymbol{1}}[\boldsymbol{v_1}|\boldsymbol{v_2}]^{\boldsymbol{T}}\boldsymbol{A}[\boldsymbol{v_1}|\boldsymbol{v_2}]}_{\boldsymbol{A}^{sub}_{2\times2}}\begin{bmatrix}\beta_1\\ \beta_1\end{bmatrix}_{sub}$$

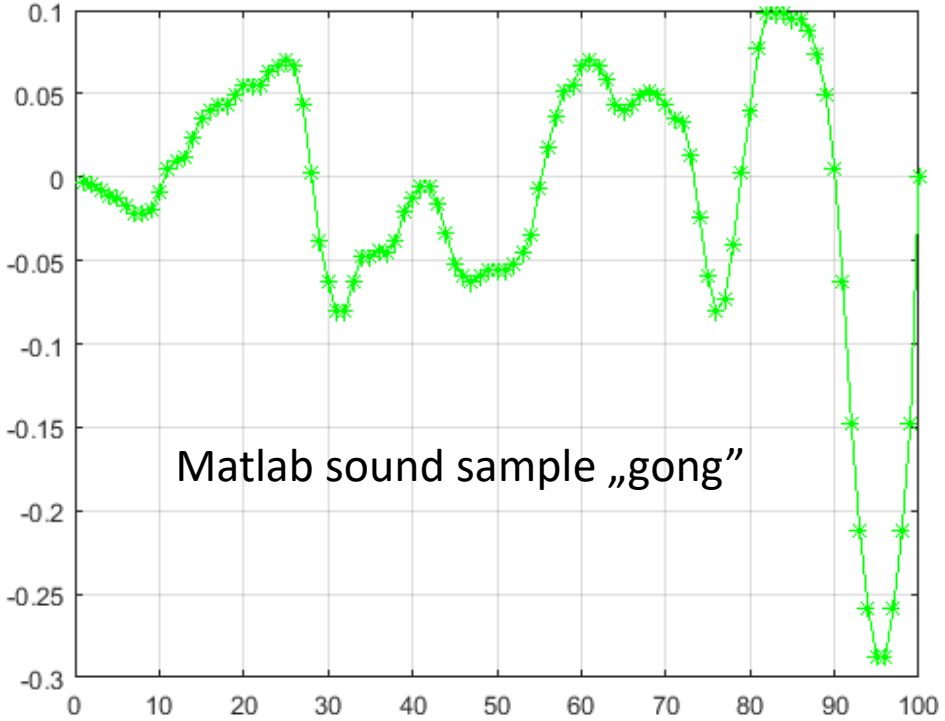
The profit from such an operation seems to be rather insignificant, but in some cases a space size of 1 milion can be reduced to subspace of size 10!!!

Vector basis - subspaces

Problem 6.13: Find coordinates of the vector $\boldsymbol{w} \in R^{100}$ (see the figure) in the new basis $\boldsymbol{v_n} = \begin{bmatrix} \sin\frac{n\pi 0}{100} \\ \sin\frac{n\pi 1}{100} \\ \sin\frac{n\pi 2}{100} \\ \vdots \\ \sin\frac{n\pi 100}{100} \end{bmatrix}_{100x1}$

Solution: First, let us plot a few of the new basis vectors





Vector basis - subspaces

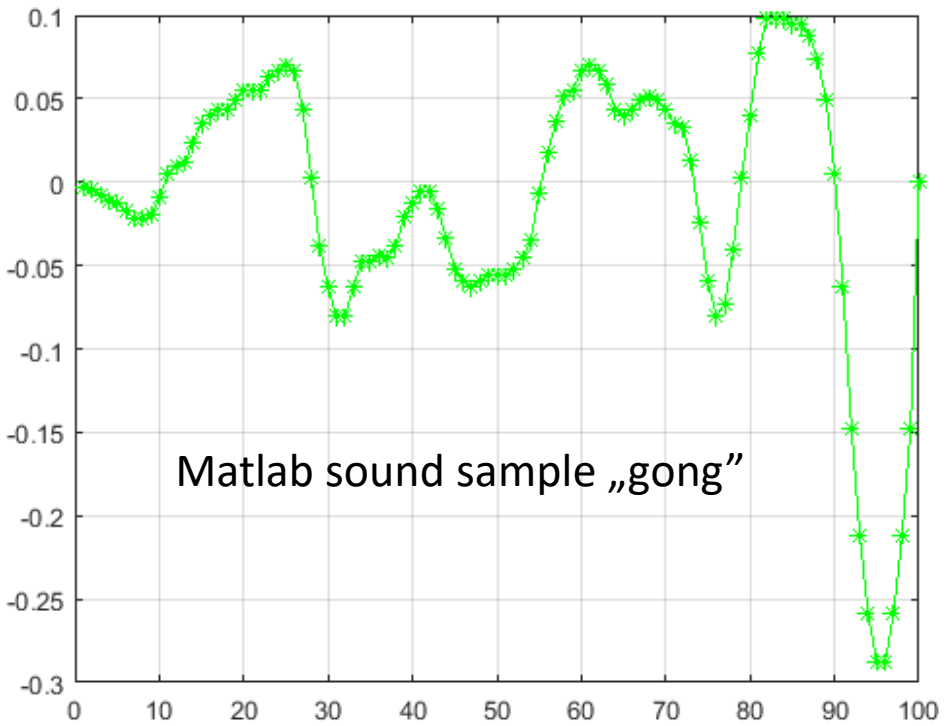
Problem 6.13: Find coordinates of the vector $\boldsymbol{w} \in R^{100}$ (see the figure) in the new basis $\boldsymbol{v_n} = \begin{bmatrix} \sin\frac{n\pi 0}{100} \\ \sin\frac{n\pi 1}{100} \\ \sin\frac{n\pi 2}{100} \\ \vdots \\ \sin\frac{n\pi 100}{100} \end{bmatrix}_{100x1}$

Solution: In order to find the whole set of the new coordinates, we can use the derived relations

$$\boldsymbol{w} = \beta_1 \boldsymbol{v_1} + \beta_2 \boldsymbol{v_2} + \cdots + \beta_{100} \boldsymbol{v_{100}}$$

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{100} \end{bmatrix} = [\boldsymbol{v_1}|\boldsymbol{v_2}|\ldots|\boldsymbol{v_{100}}] \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{100} \end{bmatrix}_{in\ the\ new\ basis}$$

In some cases a smaller number of the new basis vectors is sufficient to express the considered vector.

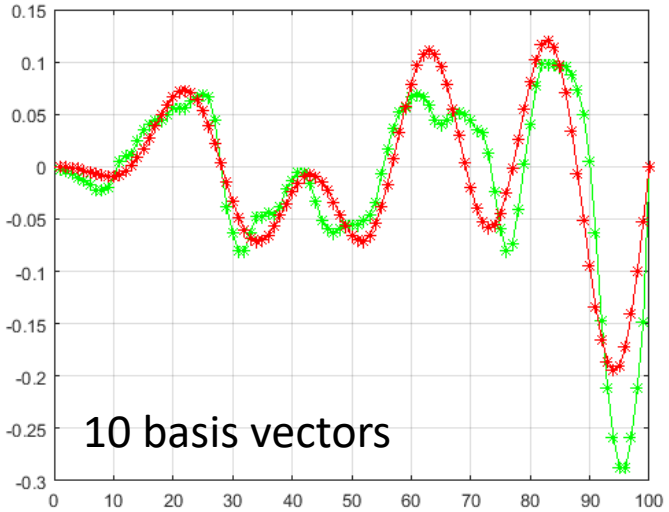


Vector basis - subspaces

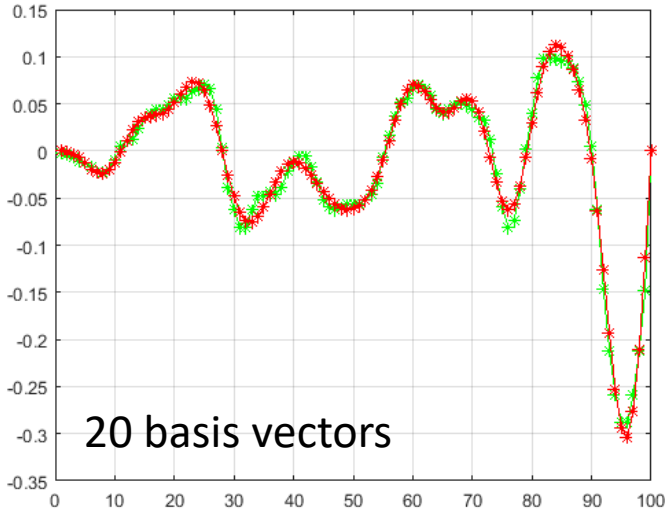
Problem 6.13: Find coordinates of the vector $\boldsymbol{w} \in R^{100}$ (see the figure) in the new basis $\boldsymbol{v_n} = \begin{bmatrix} \sin\frac{n\pi 0}{100} \\ \sin\frac{n\pi 1}{100} \\ \sin\frac{n\pi 2}{100} \\ \vdots \\ \sin\frac{n\pi 100}{100} \end{bmatrix}_{100x1}$

Solution: Let us consider three cases of expansion with the use of smaller number of basis vectors $\boldsymbol{v_n}$.

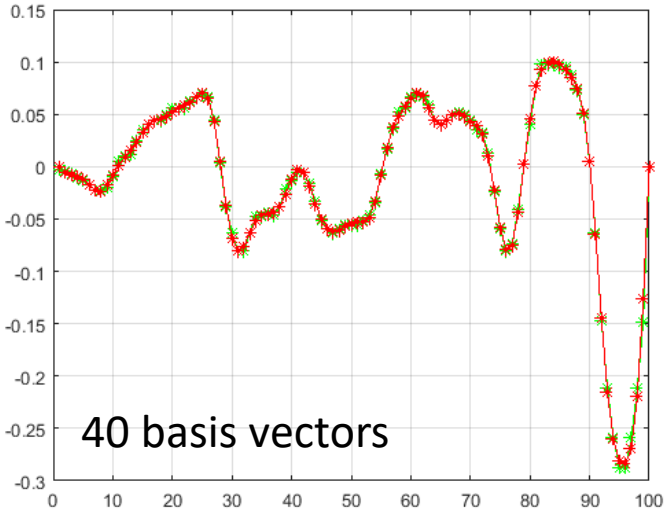
$\boldsymbol{w} = \beta_1\boldsymbol{v_1} + \beta_2\boldsymbol{v_2} + \cdots + \beta_{10}\boldsymbol{v_{10}}$



$\boldsymbol{w} = \beta_1\boldsymbol{v_1} + \beta_2\boldsymbol{v_2} + \cdots + \beta_{20}\boldsymbol{v_{20}}$



$\boldsymbol{w} = \beta_1\boldsymbol{v_1} + \beta_2\boldsymbol{v_2} + \cdots + \beta_{40}\boldsymbol{v_{40}}$



Eigenvectors as a basis

How do you find the most appropriate subspace for a given problem?

It depends on the issue and there is no general (universal) solution.

The necessary condition is, of course, the linear independence of the vectors.

In some cases eigenvectors corresponding to the smallest or greatest eigenvalues can be applied as a set of vectors defining the most suitable subspace.

Eigenproblem

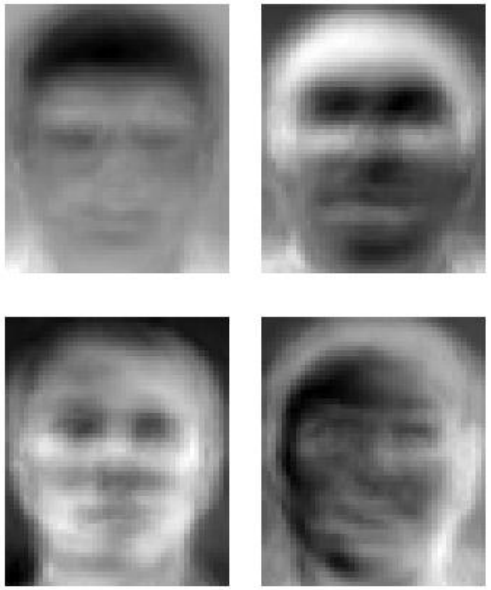
$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$$

generates N eigenvectors $\boldsymbol{x_n}$ and eigenvalues λ_n if matrix $\boldsymbol{A}$ is of size $N \times N$

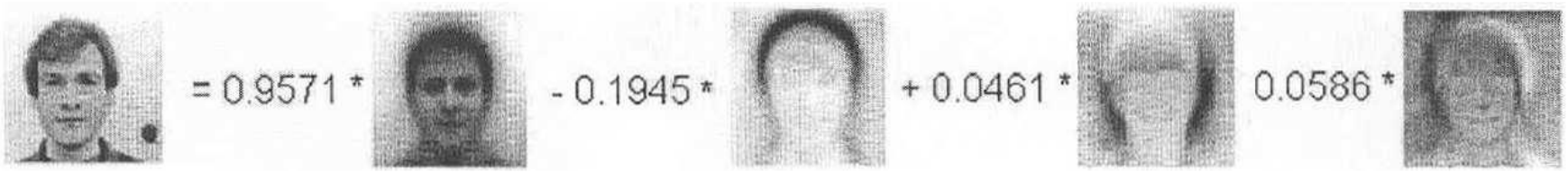
Eigenvectors as a basis

Eigenfaces is the name given to a set of eigenvectors when they are used in the computer vision problem of human face recognition. The approach of using eigenfaces for recognition was developed by Sirovich and Kirby (1987) and used by Matthew Turk and Alex Pentland in face classification. The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images. The eigenfaces themselves form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the original training images. Classification can be achieved by comparing how faces are represented by the basis set.

Source: https://en.wikipedia.org/wiki/Eigenface



Some eigenfaces from AT&T Laboratories Cambridge



M. Turk and A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, vol. 3, no. 1, 1991,

Modeling and Simulation of Systems

Lecture 8: Helmholtz equation

Piotr Kowalczyk

Helmholtz equation

The Helmholtz equation, named for Hermann von Helmholtz, is the partial differential equation

$$\frac{\partial^2 f}{\partial x^2}(x) + k^2 f(x) = 0$$

where k is a real or a complex number (constant).

It can be simply generalized into two or three dimensions

$$\frac{\partial^2 f}{\partial x^2}(x, y) + \frac{\partial^2 f}{\partial y^2}(x, y) + k^2 f(x, y) = 0$$

The Helmholtz equation arises in problems in such areas as:

- Acoustics
- Seismology
- Electromagnetics
- Many different physics models

Hermann L. F. von Helmholtz (1821 – 1894) was a German physician and physicist who made significant contributions in several scientific fields. The largest German association of research institutions, the Helmholtz Association, is named after him.

Source: https://en.wikipedia.org/wiki/Helmholtz_equation

Helmholtz equation in one dimension

Using the Helmholtz equation, we can analyze vibrations of string, membrane or any different periodic movements **without time!**

Let us consider once again the wave equation

$$\frac{\partial^2 f}{\partial t^2}(x,t) = v^2 \frac{\partial^2 f}{\partial x^2}(x,t)$$

The unknown function can be expressed as a product of two functions (separation of variables)

$$f(x,t) = S(x)T(t)$$

Then,

$$S(x)\frac{\partial^2 T}{\partial t^2}(t) = v^2 T(t)\frac{\partial^2 S}{\partial x^2}(x)$$

Next,

$$\frac{1}{T(t)}\frac{\partial^2 T}{\partial t^2}(t) = v^2 \frac{1}{S(x)}\frac{\partial^2 S}{\partial x^2}(x) \ = -\omega^2 = const$$

Helmholtz equation in one dimension

So, we have two separate equations

$$\frac{1}{T(t)}\frac{\partial^2 T}{\partial t^2}(t) = -\omega^2 \qquad v^2\frac{1}{S(x)}\frac{\partial^2 S}{\partial x^2}(x) = -\omega^2$$

Both equations have the same form

$$\frac{\partial^2 T}{\partial t^2}(t) + \omega^2 T(t) = 0$$

Helmholtz equation

$$\boxed{\frac{\partial^2 S}{\partial x^2}(x) + \frac{\omega^2}{v^2}S(x) = 0}$$

and can be easily solved

$$T(t) = C_1\cos(\omega t) + C_2\sin(\omega t) \qquad S(x) = C_3\cos\left(\frac{\omega}{v}x\right) + C_4\sin\left(\frac{\omega}{v}x\right)$$

This is not the solution of the string problem yet!!!

The parameter ω can be arbitrary, so in order to find the general solution we have to sum all the possible cases

$$f(x,t) = \int_\Omega S_\omega(x)\,T_\omega(t)\,d\omega$$

$$= \int_\Omega \left(C_3(\omega)\cos\left(\frac{\omega}{v}x\right) + C_4(\omega)\sin\left(\frac{\omega}{v}x\right)\right)\left(C_1(\omega)\cos(\omega t) + C_2(\omega)\sin(\omega t)\right)d\omega$$

Helmholtz equation in one dimension

Problem 8.1: Use the finite difference method to find solutions of the Helmholtz equation for a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the wave velocity is known $v = 100$ m/s.

Solution: The Helmholtz equation can be written in the following form

$$\frac{\partial^2 S}{\partial x^2}(x) + k^2 S(x) = 0$$

where $k = \frac{\omega}{v}$.

The finite difference method requires discretization of the domain (the same as for wave equation)

$$S_m = S(x_m) \qquad\qquad x_m = x_1 + (m-1)\Delta x$$

Then the standard finite difference formula can be applied

$$\frac{\partial^2 S}{\partial x^2}(x_m) \approx \frac{S_{m+1} - 2S_m + S_{m-1}}{\Delta x^2}$$

Hence, we get the Helmholtz equation in the discrete form

$$\frac{S_{m+1} - 2S_m + S_{m-1}}{\Delta x^2} + k^2 S_m = 0$$

Helmholtz equation in one dimension

Problem 8.1: Use the finite difference method to find the solutions of Helmholtz equation for a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the wave velocity is known $v = 100$ m/s.

Solution: Assuming the boundary conditions equal zero: $S_1 = S_M = 0$, we get

$$\begin{cases} \frac{1}{\Delta x^2}(S_3 - 2S_2) = -k^2 S_2 \\ \frac{1}{\Delta x^2}(S_4 - 2S_3 + S_2) = -k^2 S_3 \\ \frac{1}{\Delta x^2}(S_5 - 2S_4 + S_3) = -k^2 S_4 \\ \vdots \\ \frac{1}{\Delta x^2}(S_{M-1} - 2S_{M-2} + S_{M-3}) = -k^2 S_{M-2} \\ \frac{1}{\Delta x^2}(-2S_{M-1} + S_{M-2}) = -k^2 S_{M-1} \end{cases}$$

$$\frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & \dots & 0 & 0 \\ 1 & -2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -2 & 1 \\ 0 & 0 & \dots & 1 & -2 \end{bmatrix}\begin{bmatrix} S_2 \\ S_3 \\ \vdots \\ S_{M-2} \\ S_{M-1} \end{bmatrix} = -k^2 \begin{bmatrix} S_2 \\ S_3 \\ \vdots \\ S_{M-2} \\ S_{M-1} \end{bmatrix}$$

$$\boxed{\boldsymbol{Ax} = \lambda \boldsymbol{x}}$$

Helmholtz equation in one dimension

Problem 8.1: Use the finite difference method to find the solutions of Helmholtz equation for a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the wave velocity is known $v = 100$ m/s.

Solution: Then, the eigenvectors of the operator

$$A = \frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & \dots & 0 & 0 \\ 1 & -2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -2 & 1 \\ 0 & 0 & \dots & 1 & -2 \end{bmatrix}$$

represent the solutions $S(x)$ and corresponding eigenvalues represent the <u>allowed values of parameter k</u>.

In details,

$$S(x) \to S_m = S(x_m)$$

and

$$k \to \sqrt{-\lambda}$$

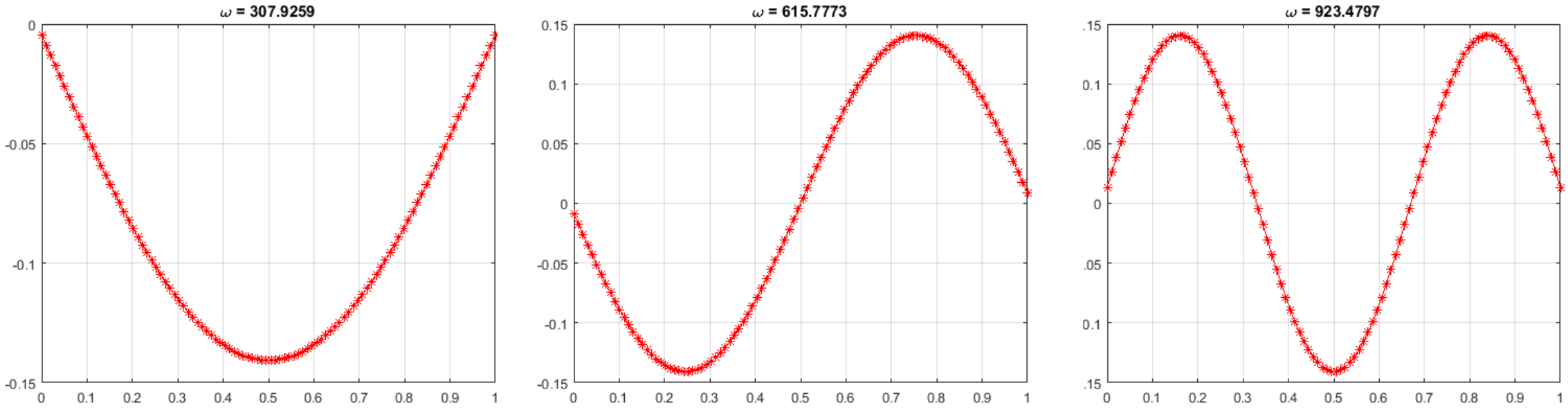
The parameter is related to the angular frequency $\omega = kv = v\sqrt{-\lambda}$

```
MATLAB CODE EXAMPLE:

L=1; v=100; M=100;
x=linspace(0,L,M)';
dx=x(2)-x(1);
tmp=ones(M,1); tmp1=ones(M-1,1);
A=(-2*diag(tmp)+diag(tmp1,1)+diag(tmp1,-1))/dx^2;
[X,L]=eig(A);


for i=M:-1:M-10
    plot(x,X(:,i),'r*-')
    lambda=L(i,i);
    omega=v*sqrt(-lambda);
    title(['\omega = ',num2str(omega)])
    grid on
    pause
end
```

Helmholtz equation in one dimension

Problem 8.1: Use the finite difference method to find the solutions of Helmholtz equation for a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the wave velocity is known $v = 100$ m/s.

Solution: The eigenvectors and eigenvalues (resonant angular frequency)



Helmholtz equation in one dimension

Problem 8.1: Use the finite difference method to find the solutions of Helmholtz equation for a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the wave velocity is known $v = 100$ m/s.

Solution: Since the eigenvectors and the eigenvalues are known the simulation of the string vibrations can be performed without solving the wave equation. The angular frequency can take only discrete values and the time dependence is known (amplitudes depends on initial conditions)

$$T(t) = C_1 \cos(\omega t) + C_2 \sin(\omega t)$$

Then finally the integral

$$f(x,t) = \int_{\Omega} S_\omega(x)\, T_\omega(t)\, d\omega$$

Can be replace by a simple sum.

The simulation is fast and efficient – see the example of a simple case.

```
MATLAB CODE EXAMPLE:

for t=0:0.00001:0.01
    f=7*X(:,M)*cos(omega(M)*t)
      +3*X(:,M-1)*cos(omega(M-1)*t)
      +1*X(:,M-2)*cos(omega(M-3)*t);
    plot(x,f)
    ylim([-3 3])
    drawnow
end
```

Helmholtz equation in one dimension

Problem 8.1: Use the finite difference method to find the solutions of Helmholtz equation for a string of length $L = 1$ m. Assume, that the string is fixed at the both ends and the wave velocity is known $v = 100$ m/s.

Solution: Since the eigenvectors and the eigenvalues are known the simulation of the string vibrations can be performed without solving the wave equation. The angular frequency can take only discrete values and the time dependence is known (amplitudes depends on initial conditions)

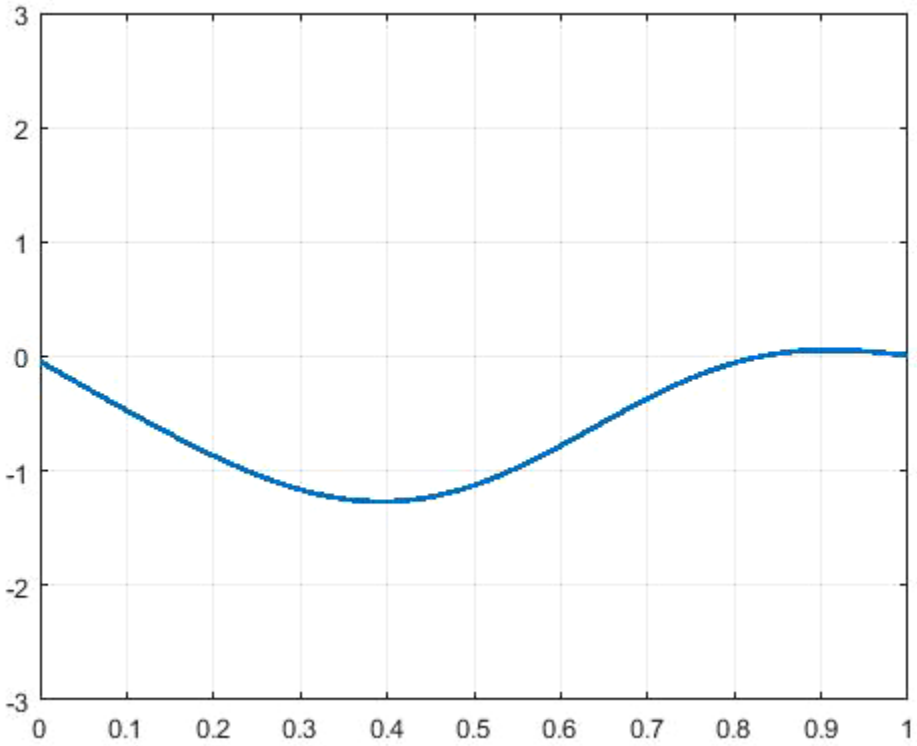
$$T(t) = C_1 \cos(\omega t) + C_2 \sin(\omega t)$$

Then finally the integral

$$f(x,t) = \int_{\Omega} S_\omega(x)\, T_\omega(t)\, d\omega$$

Can be replace by a simple sum.

The simulation is fast and efficient – see the example of a simple case.



Helmholtz equation in two dimensions

Using the Helmholtz equation, we can analyze vibrations of string, membrane or any different periodic movements **without time!**

Let us consider once again the wave equation, this time in two dimensions

$$\frac{\partial^2 f}{\partial t^2}(x,y,t) = v^2 \frac{\partial^2 f}{\partial x^2}(x,y,t) + v^2 \frac{\partial^2 f}{\partial x^2}(x,y,t)$$

As in the one dimension, the unknown function can be expressed as a product of two functions

$$f(x,y,t) = S(x,y)T(t)$$

Then,

$$S(x,y)\frac{\partial^2 T}{\partial t^2}(t) = v^2 T(t)\frac{\partial^2 S}{\partial x^2}(x,y) + v^2 T(t)\frac{\partial^2 S}{\partial y^2}(x,y)$$

Next,

$$\frac{1}{T(t)}\frac{\partial^2 T}{\partial t^2}(t) = v^2 \frac{1}{S(x,y)}\frac{\partial^2 S}{\partial x^2}(x,y) + v^2 \frac{1}{S(x,y)}\frac{\partial^2 S}{\partial y^2}(x,y) \ = -\omega^2 = const$$

Finally, we get the Helmholtz equation in two dimensions

$$\frac{\partial^2 S}{\partial x^2}(x,y) + \frac{\partial^2 S}{\partial y^2}(x,y) + \frac{\omega^2}{v^2}S(x,y) = 0$$

Helmholtz equation in two dimensions

Problem 8.2: Use the finite difference method to find solutions of the Helmholtz equation for a rectangular membrane. Assume, that the membrane is fixed at the boundaries and the wave velocity $v = 100$ m/s.

Solution: The Helmholtz equation can be written in the following form

$$\frac{\partial^2 S}{\partial x^2}(x,y) + \frac{\partial^2 S}{\partial y^2}(x,y) + k^2 S(x,y) = 0$$

where $k = \frac{\omega}{v}$.

The finite difference method requires discretization of the domain (the same as for wave equation)

$$x_m = x_1 + (m-1)\Delta x$$
$$y_n = y_1 + (n-1)\Delta y$$

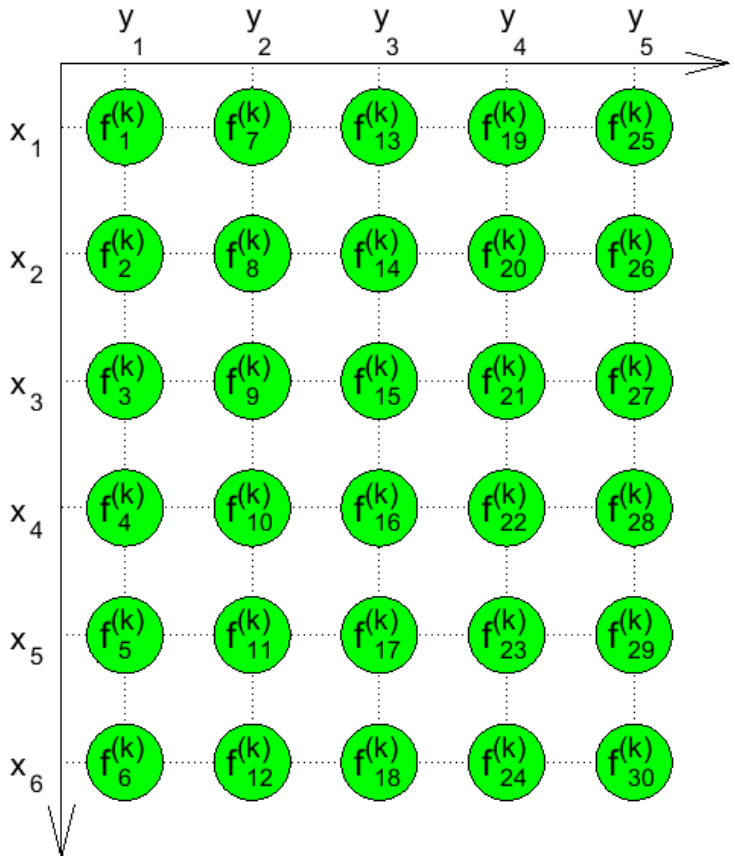
Then

$$S(x,y) \to S(x_m, y_n)$$

or with the one index

$$S_i = S(x_m, y_n)$$

where $i = m + (n-1)M$.



Helmholtz equation in two dimensions

Problem 8.2: Use the finite difference method to find solutions of the Helmholtz equation for a rectangular membrane. Assume, that the membrane is fixed at the boundaries and the wave velocity $v = 100$ m/s.

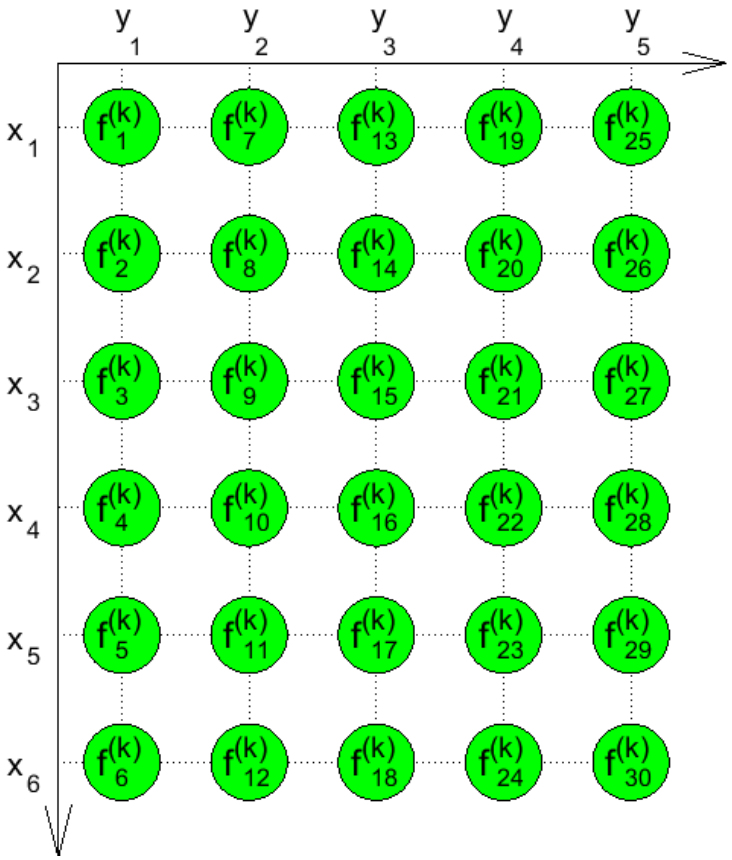
Solution: So, the Helmholtz equation can be written in the discrete form, as follows

$$\frac{S_{i+1} - 2S_i + S_{i+1}}{\Delta x^2} + \frac{S_{i+M} - 2S_i + S_{i-M}}{\Delta y^2} + k^2 S_i = 0.$$

Assuming zero at the boundaries we get the following eigenvalue problem

$$\begin{bmatrix} c & c_x & 0 & 0 & 0 & c_y & 0 & 0 & 0 & 0 & 0 & 0 \\ c_x & c & c_x & 0 & 0 & 0 & c_y & 0 & 0 & 0 & 0 & 0 \\ 0 & c_x & c & c_x & 0 & 0 & 0 & c_y & 0 & 0 & 0 & 0 \\ 0 & 0 & c_x & c & c_x & 0 & 0 & 0 & c_y & 0 & 0 & 0 \\ 0 & 0 & 0 & c_x & c & c_x & 0 & 0 & 0 & c_y & 0 & 0 \\ c_y & 0 & 0 & 0 & c_x & c & c_x & 0 & 0 & 0 & c_y & 0 \\ 0 & c_y & 0 & 0 & 0 & c_x & c & c_x & 0 & 0 & 0 & c_y \\ 0 & 0 & c_y & 0 & 0 & 0 & c_x & c & c_x & 0 & 0 & 0 \\ 0 & 0 & 0 & c_y & 0 & 0 & 0 & c_x & c & c_x & 0 & 0 \\ 0 & 0 & 0 & 0 & c_y & 0 & 0 & 0 & c_x & c & c_x & 0 \\ 0 & 0 & 0 & 0 & 0 & c_y & 0 & 0 & 0 & c_x & c & c_x \\ 0 & 0 & 0 & 0 & 0 & 0 & c_y & 0 & 0 & 0 & c_x & c \end{bmatrix} \begin{bmatrix} f_8 \\ f_9 \\ f_{10} \\ f_{11} \\ f_{14} \\ f_{15} \\ f_{16} \\ f_{17} \\ f_{20} \\ f_{21} \\ f_{22} \\ f_{23} \end{bmatrix} = -k^2 \begin{bmatrix} f_8 \\ f_9 \\ f_{10} \\ f_{11} \\ f_{14} \\ f_{15} \\ f_{16} \\ f_{17} \\ f_{20} \\ f_{21} \\ f_{22} \\ f_{23} \end{bmatrix}$$

where $c_x = \frac{1}{\Delta x^2}$, $c_y = \frac{1}{\Delta y^2}$, $c = -\left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)$



Helmholtz equation in two dimensions

Problem 8.2: Use the finite difference method to find solutions of the Helmholtz equation for a rectangular membrane. Assume, that the membrane is fixed at the boundaries and the wave velocity $v = 100$ m/s.

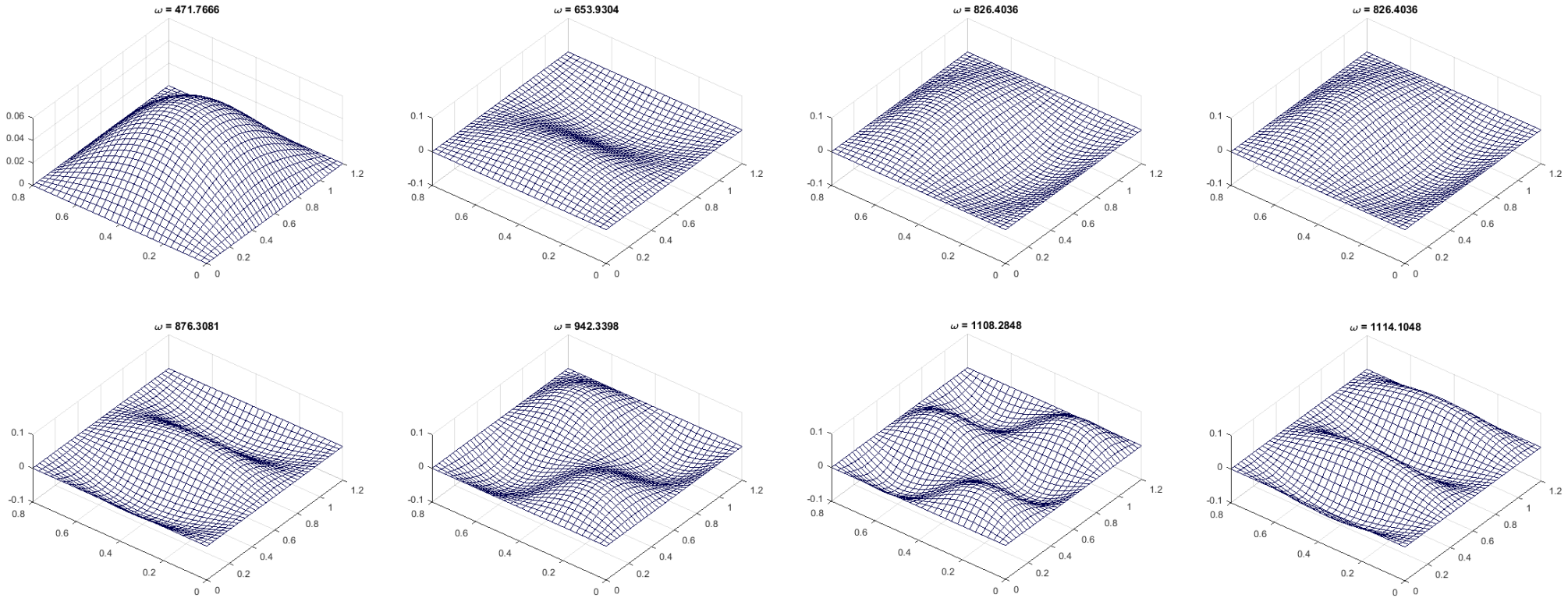
Solution: The implementation is also quite easy and similar to the Laplace equation considered in the previous lecture.

```
MATLAB CODE EXAMPLE:

M=40; N=30; a=1.2; b=0.8; v=100; x=linspace(0,a,M); y=linspace(0,b,N); dx=x(2)-x(1);
dy=y(2)-y(1); [X,Y]=meshgrid(x,y); X=reshape(X',M*N,1); Y=reshape(Y',M*N,1); Pout=[];
Pbnd=sort([1:M 1+M:M:1+(N-2)*M 2*M:M:(N-1)*M 1+(N-1)*M:N*M]); Pin=[1:M*N]; Pin(Pbnd)=[];
Bnd=Pbnd*0;

cx=(1/dx)^2;cy=(1/dy).^2;c=-2/dx^2-2/dy^2;
tmp=ones(1,M*N); tmp1=ones(1,M*N-1);tmpM=ones(1,M*N-M);
A=c*diag(tmp)+cx*diag(tmp1,-1)+cx*diag(tmp1,1)+cy*diag(tmpM,-M)+cy*diag(tmpM,M);
A(Pbnd,:)=[]; A(:,Pbnd)=[];
[V,L]=eig(A);

for p=size(A,1):-1:1
    Vp(Pin')=V(:,p);
    Vp(Pbnd)=Bnd;
    mesh(reshape(X,M,N),reshape(Y,M,N),reshape(Vp,M,N))
    lambda(p)=L(p,p);
    omega(p)=v*sqrt(-lambda(p));
    title(['\omega = ',num2str(omega(p))])
end
```

Helmholtz equation in two dimensions

Problem 8.2: Use the finite difference method to find solutions of the Helmholtz equation for a rectangular membrane. Assume, that the membrane is fixed at the boundaries and the wave velocity $v = 100$ m/s.

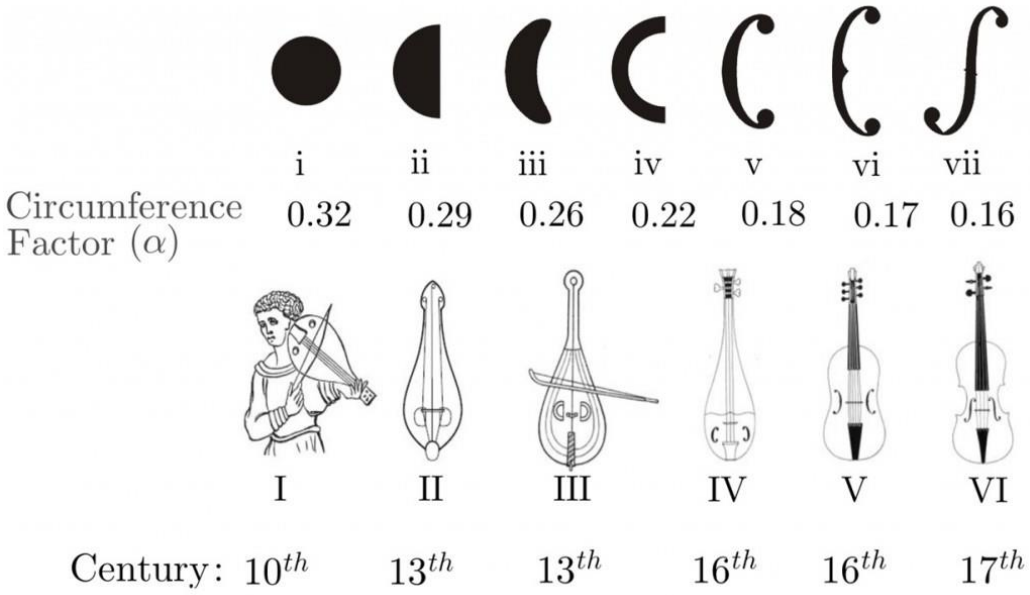
Solution: The eigenvectors (and corresponding eigenvalues) obtained from the analysis

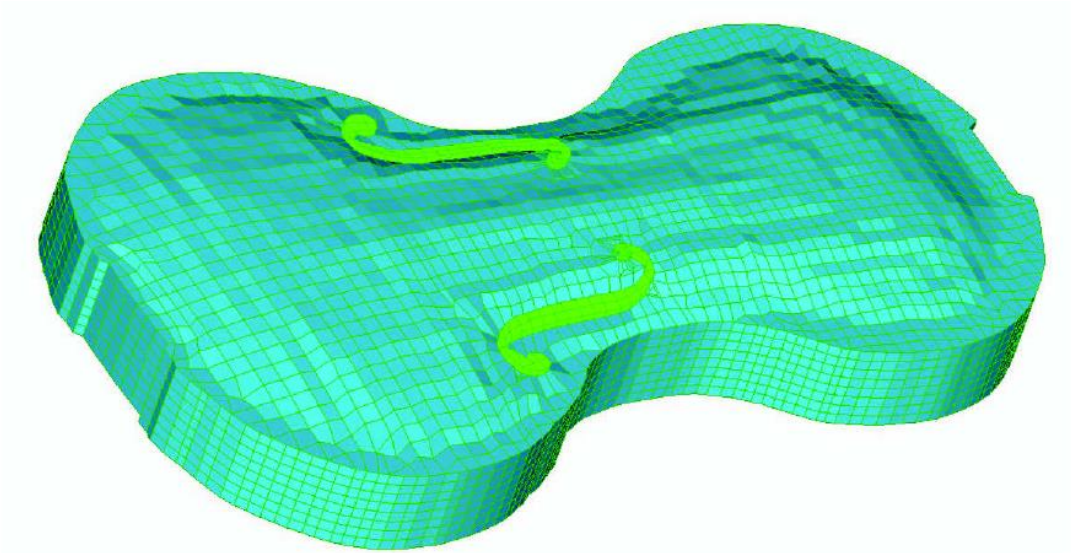


Helmholtz equation in three dimensions

The shape of acoustic chamber and sound-holes is optimized for tuning the resonant frequencies.

The line of development of chamber and sound-holes in violin





Source: http://taflab.berkeley.edu/acoustics-of-sound-holes-in-musical-instruments/

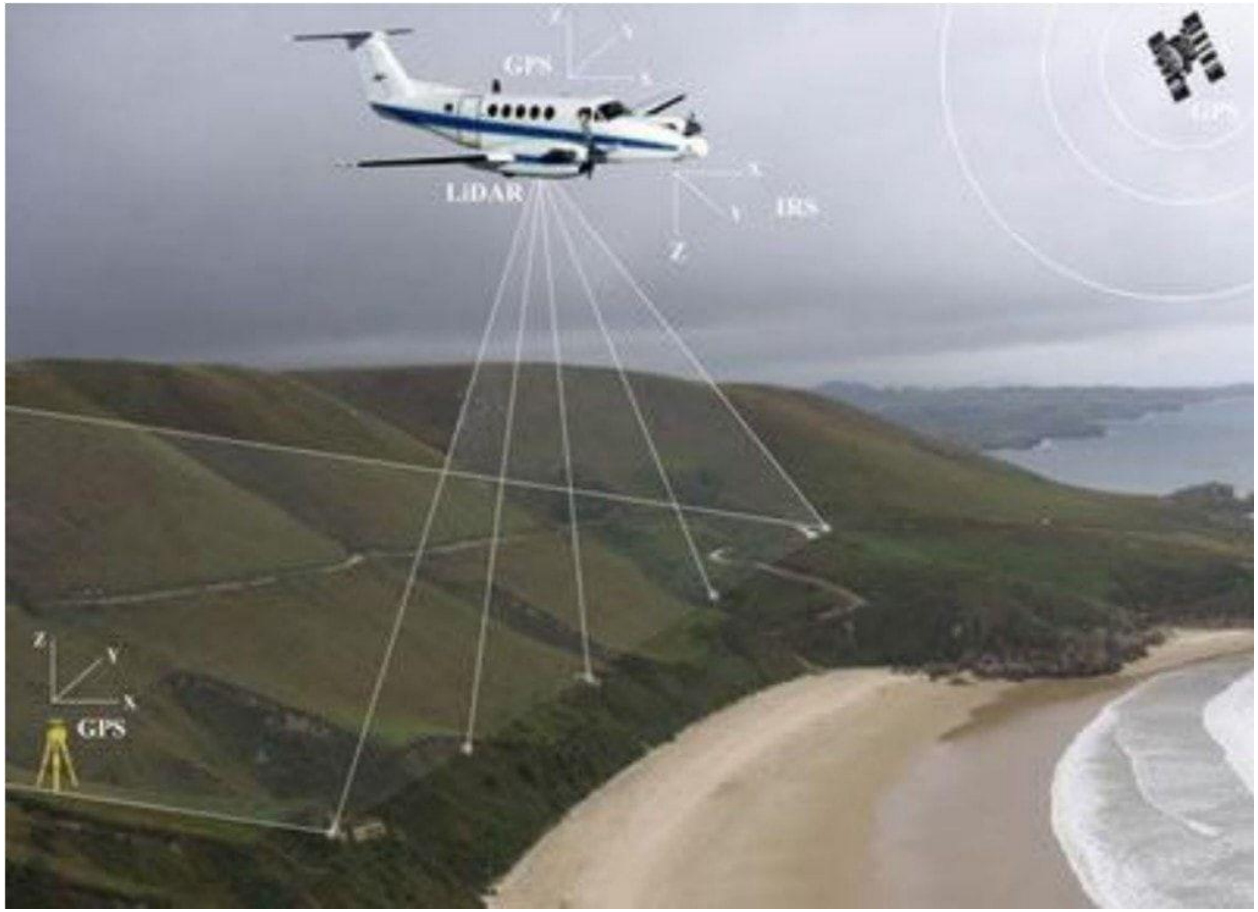
Source: http://lisafea.com/pdf/JMC-Helmholtz_Resonance--Re-ct2.pdf

Modeling and simulation of systems

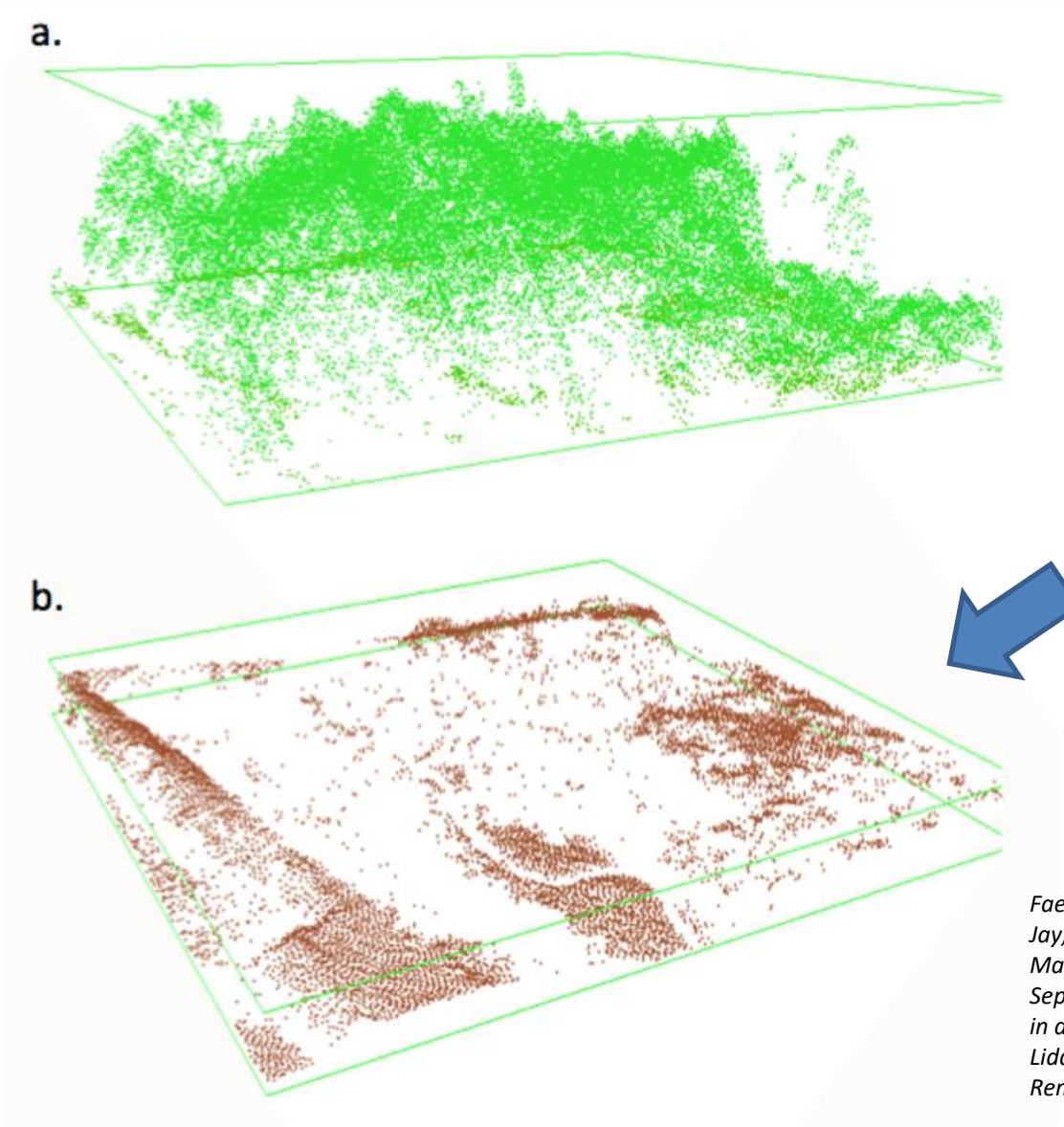
Lecture 9: Radial basis functions & applications in data modeling

Adam Lamęcki
WETI PG, room 708

LIDAR terrain mapping



LIDAR terrain mapping



Our task: create a smooth model of the terrain

Faelga, Regine Anne & Paringit, Enrico & Gay, Jane & Perez, & Jay, Reginald & Argamosa, L & Carlyn, Ann & Nez, Ibã & Anthony, Mark & Posilero, V & Fe, Andrea & Tandoc, & Zaragosa, Gio. Separability and variability of Rhizophocraceae and Avicenniaceae in a Natural Mangrove Forest Using Point Density Disribution from Lidar Data. Journal of the Philippine Society of Geosciences and Remote Sensing, 2016

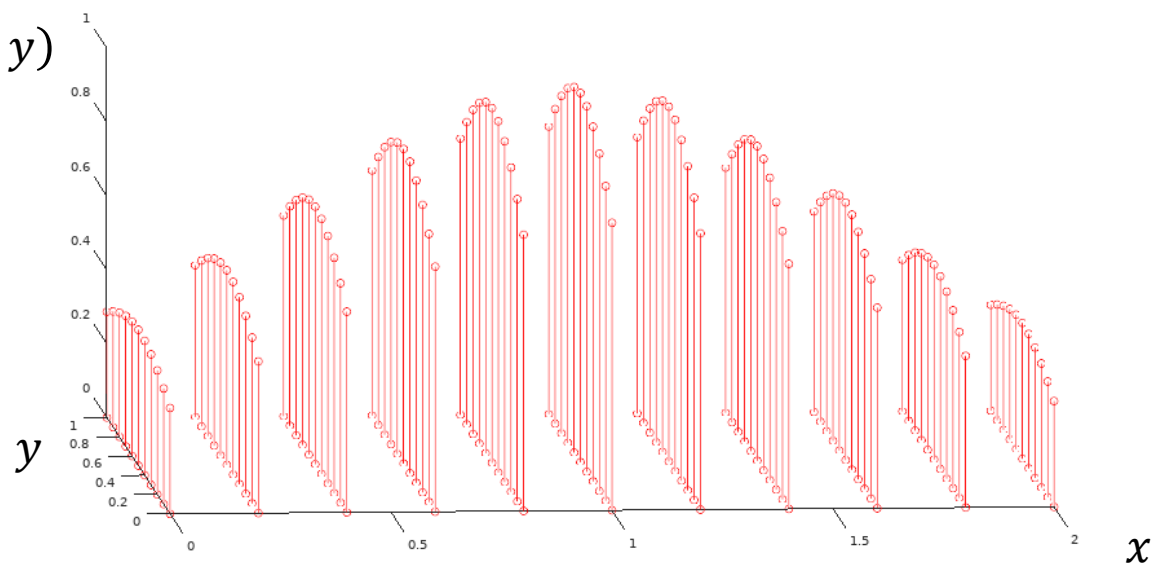
Data modelling

Let's try to solve the problem:

Given a set of K discrete data points x_i, y_i *and values* $f_i = (x_i, y_i)$ *on XY plane try to restore the* ***smooth*** *surface* $f(x, y)$.

Initially we assume that points form a regular rectangular grid:

$z = f(x, y)$



A: Use a simple 2D interpolation scheme: e.g. multivariate polynomial

Data interpolation

- To solve this we need to construct a bi-variate polynomial $p(x, y)$, for example as:

$$p(x, y) = \sum_{p=0}^{N_x} \sum_{q=0}^{N_y} a_{pq} x^p \, y^q$$

Nx, Ny determine funtion order

One can write it directly as:

$$p(x, y) = a_{00} + a_{10}x + a_{11}xy + a_{01}y + \cdots$$
$$\ldots + a_{20}x^2 + a_{21}x^2y + a_{22}x^2y^2 + \cdots$$

- Our **unknowns**: coefficients $\boldsymbol{a_{ij}}$ $\rightarrow$ how to find them?

Data interpolation

- A: write a set of equations: for each sample we have

$$f_i = f(x_i, y_i) = a_{00} + a_{10}x_i + a_{11}x_i y_i + a_{01}y_i + \cdots$$
$$\ldots + a_{20}x_i^2 + a_{22}x_i^2 y_i^2 + a_{02}y_i^2 + \cdots$$

- We get a system of *K* equations with $(Nx + 1) \cdot (Ny + 1)$ unknowns

$$\boldsymbol{f} = \boldsymbol{M} \cdot \boldsymbol{a},$$

($\boldsymbol{M}$ is matrix size of $[K \times (Nx + 1) \cdot (Ny + 1)]$)

where:

$\boldsymbol{a}$ – vector of unknowns coefficients a_{ij}

$\boldsymbol{f}$ – vector of sampled function values f_i

Data interpolation

- Main problem matrix $\boldsymbol{M}$ is matrix size of $[K \times (Nx+1) \cdot (Ny+1)])$:

$$\mathbf{M} = \begin{bmatrix} \mathbf{1} & \boldsymbol{x_1} & \boldsymbol{x_1 y_1} & \boldsymbol{y_1} & \boldsymbol{x_1^2} & \dots & \boldsymbol{x_1^{N_x} y_1^{N_y}} \\ \mathbf{1} & \boldsymbol{x_2} & \boldsymbol{x_2 y_2} & \boldsymbol{y_2} & \dots & \dots & \\ \mathbf{1} & \boldsymbol{x_3} & \dots & \dots & \dots & \dots & \\ \dots & \dots & \dots & \dots & \dots & \dots & \\ \mathbf{1} & \boldsymbol{x_K} & \dots & \dots & \dots & \dots & \end{bmatrix}$$

Matrix elements depend on distribution of samples

Data interpolation

- If $K = (N_x+1) \cdot (N_y+1)$ the system has a unique solution

$$\boldsymbol{a} = \boldsymbol{M}^{-1}\boldsymbol{f}$$

- If $K > (N_x+1) \cdot (N_y+1)$we can find a solution as:

$$\boldsymbol{a} = \left(\boldsymbol{M}^{\boldsymbol{T}}\boldsymbol{M}\right)^{-1}\boldsymbol{M}^{\boldsymbol{T}} \cdot \boldsymbol{f}$$

(with this we minimize the residual error $\boldsymbol{r} = (\boldsymbol{f} - \boldsymbol{M} \cdot \boldsymbol{a})$ in a least square sense))

- It seems we solved the problem!!!
- **BUT: The problem is extremely bad conditioned**
  - **In our series we have powers of $x^p y^q$**
  - **Oscilatory behavior (Runge's phenomenon)**

Data interpolation

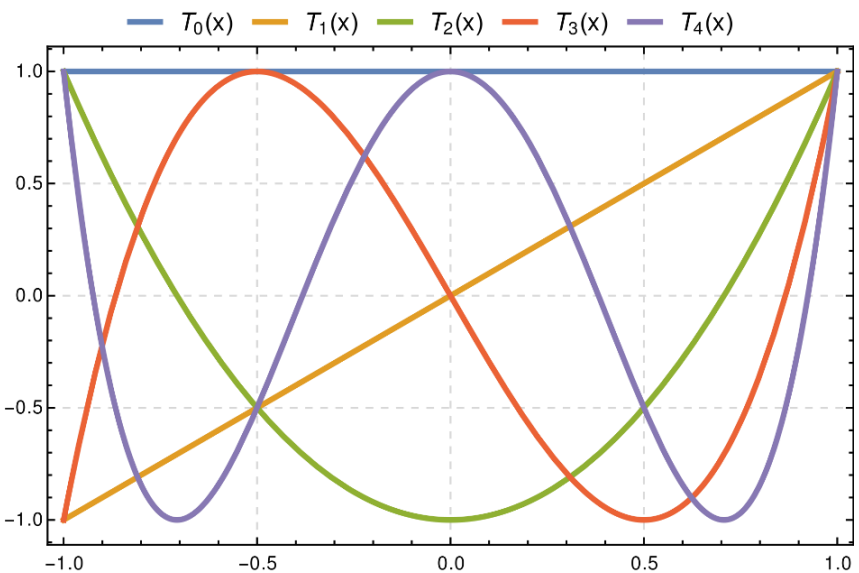
- Q1: What about conditioning if we move to higher dimensions:
  - 3D space?
  - 3D+time = 4D?
- Q2: How we can improve the scheme?
- **A: Change a set of basis functions in our series[1]!**
  - In fact, monomial basis is the worst choice!

[1] *Every continuous function in the function space can be represented as a linear combination of basis functions, just as every vector in a vector space can be represented as a linear combination of basis vectors*

Alternative basis

- Chebyshev polynomials

$$T_0(x) = 1$$
$$T_1(x) = x$$
$$T_2(x) = 2x^2 - 1$$
$$\ldots.$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$



- Function on $-1 \leq x \leq 1$ can be expressed as:

$$f(x) = \sum_{n=0}^{\infty} a_n T_n(x)$$

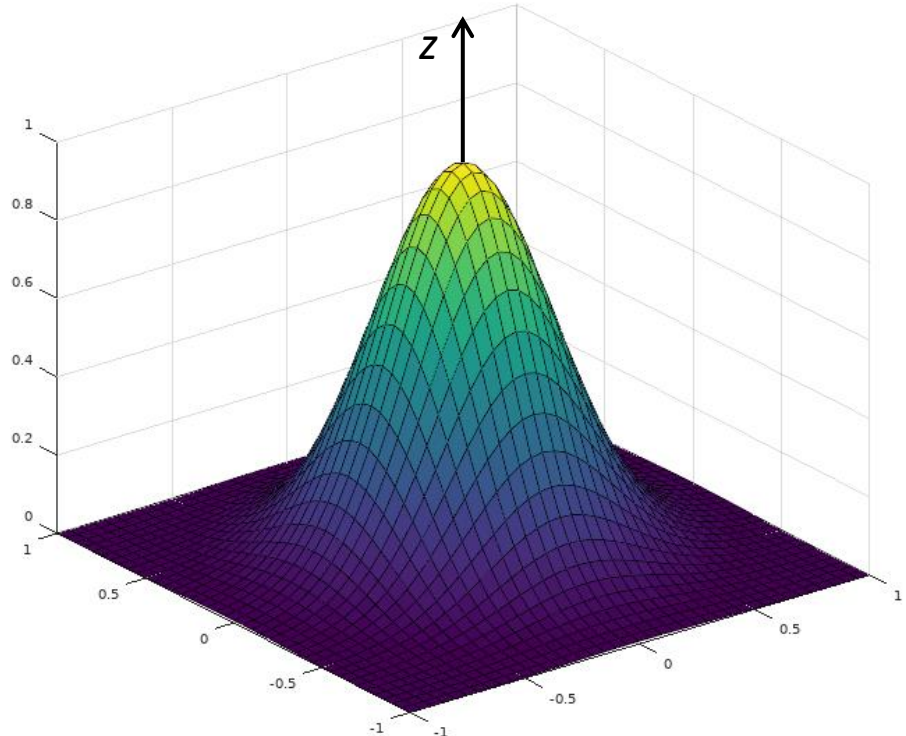
Apply linear mapping of arguments to <-1,1> segment

Radial basis functions (RBF)

Let's imagine a function $\Phi(\underline{x})$, $\underline{x} \in R^N$ whose value depends only on the distance from the origin:

$$\Phi(\underline{x}) = \Phi(\|\underline{x}\|)$$

- In 2D the function would be symmetric along z-axis



$$\Phi(\underline{x}) = e^{-5\cdot\|\underline{x}\|}$$

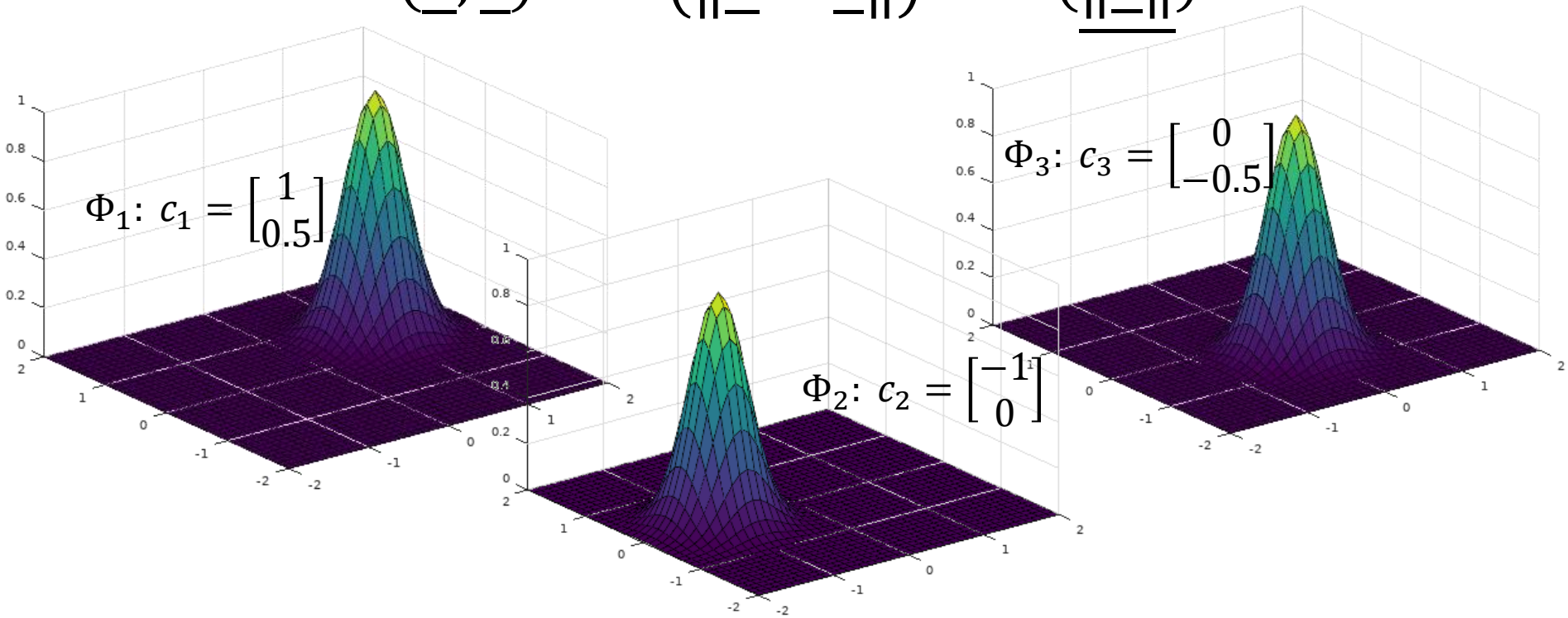
The function has a center at $\|x\| = 0$

Q: how we can move the center?

Radial basis functions

- Let's define center point as $\underline{c} \in R^N$ and distance vector $\underline{r} = (\underline{x} - \underline{c})$, then:

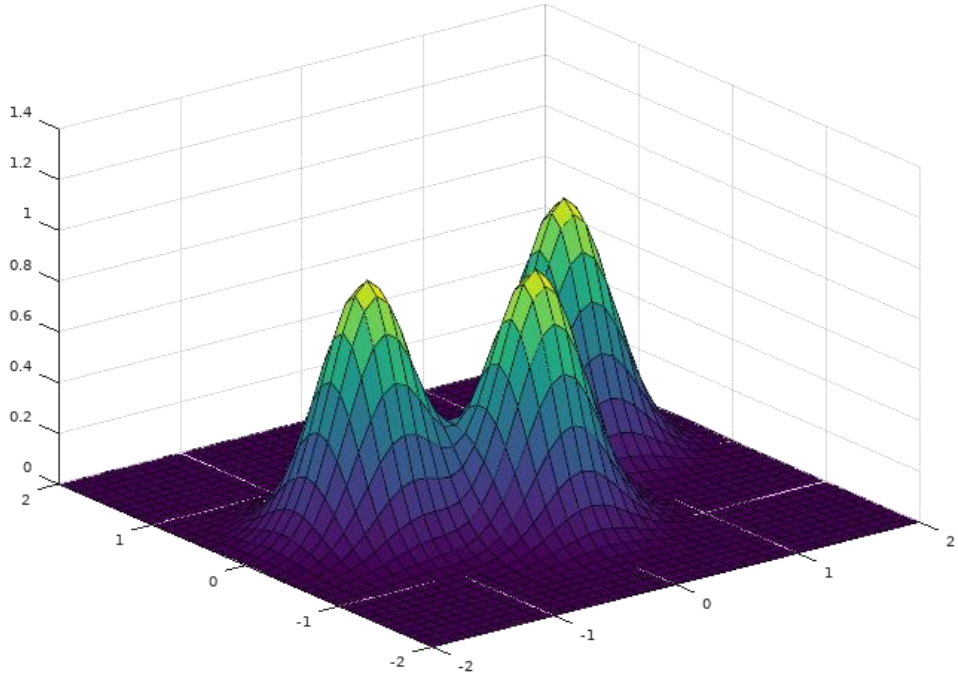
$$\Phi(\underline{x}, \underline{c}) = \Phi(\|\underline{x} - \underline{c}\|) = \Phi(\underline{\|\underline{r}\|})$$



Value of the function depends on the distance from its center (radius)

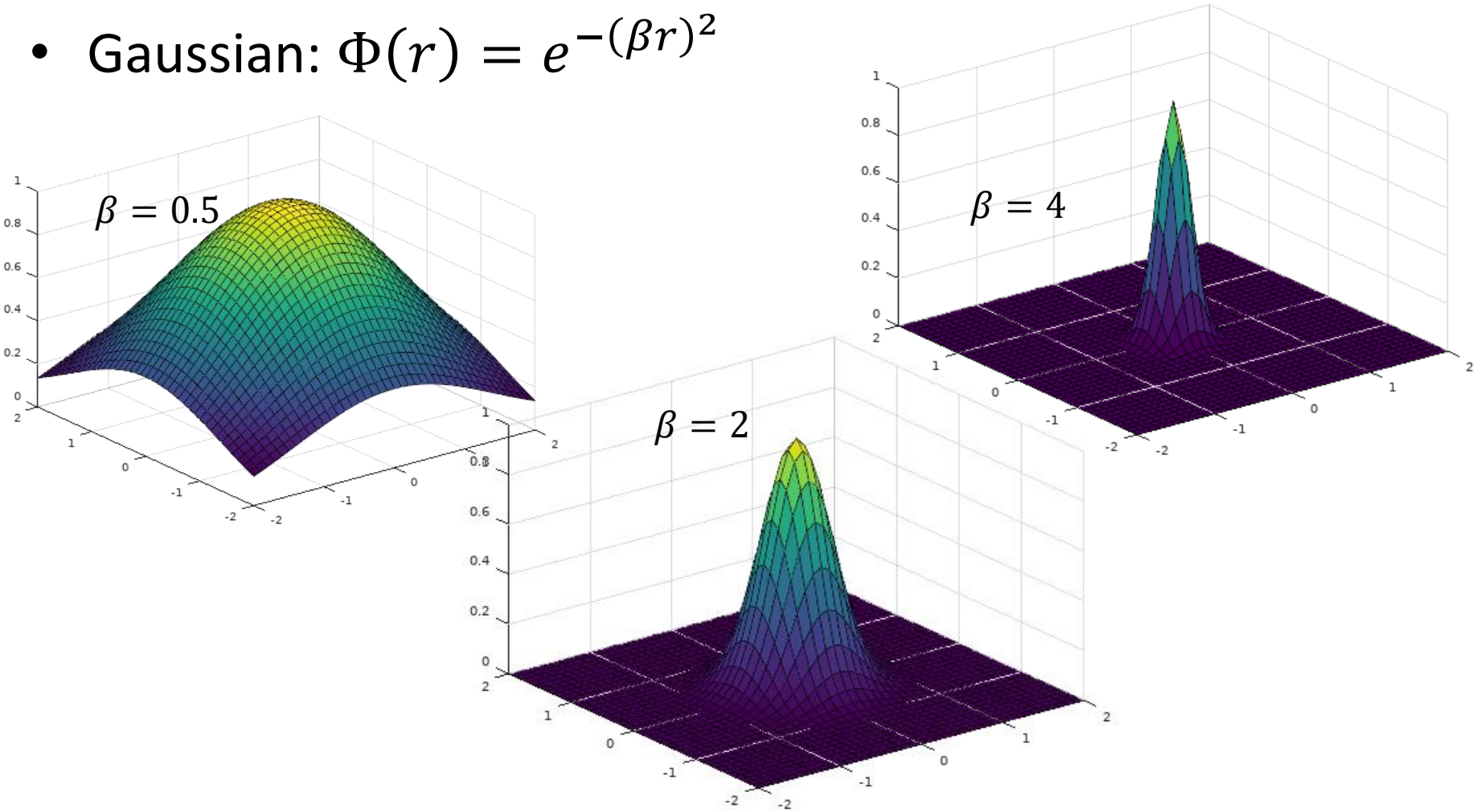
Radial basis functions

- We can add those $\Phi_1 + \Phi_2 + \Phi_3$:



Radial basis functions

- How parameter β influcences $\Phi(x)$?
- Gaussian: $\Phi(r) = e^{-(\beta r)^2}$



Radial basis functions

- Examples of commonly used RBF's:
  - Gaussian: $\Phi(r) = e^{-(\beta r)^2}$
  - Multiquadric: $\Phi(r) = \sqrt{1 + (\beta r)^2}$
  - Inverse quadric: $\Phi(r) = \frac{1}{1+(\beta r)^2}$
  - Polyharmonic: $\Phi(r) = r^k, \;\; k = 1,3,5 \ldots$
  - Polynomial: $\Phi(r) = r^2 + \beta^2$
  - Thin plate spline: $\Phi(r) = r^2 \log(r)$
  - …

Interpolation with RBF's

- Let's redefine the interpolation problem to use RBF's:
  - Set of *K* discrete samples:

$$f_i = f\left(\underline{\boldsymbol{x_i}}\right) = f\left(x_1^i, x_2^i, \dots, x_N^i\right), \quad i = 1 \dots K$$

  - Given radial basis functions with **centers at** $\underline{\boldsymbol{x_i}}$:

$$\Phi_j\left(\underline{\boldsymbol{x}}\right) = \Phi\left(x_j, \underline{\boldsymbol{x}}\right), \qquad j = 1 \dots K$$

  - For each i-th point $\underline{\boldsymbol{x_i}} \in \boldsymbol{R^N}$ we write an equation:

$$f_i = \sum_{j=1}^{K} a_j \Phi_j(\underline{\boldsymbol{r_{i,j}}}) = \sum_{j=1}^{K} a_j \Phi(\left\|\underline{\boldsymbol{x_j}} - \underline{\boldsymbol{x_i}}\right\|)$$

Interpolation with RBF's

- We have a set of K equations:

$$f_1 = \sum_{j=1}^{K} a_j \Phi(\|\underline{\boldsymbol{x_j}} - \underline{\boldsymbol{x_1}}\|)$$

$$f_2 = \sum_{j=1}^{K} a_j \Phi(\|\underline{\boldsymbol{x_j}} - \underline{\boldsymbol{x_2}}\|)$$

$$\dots$$

$$f_K = \sum_{j=1}^{K} a_j \Phi(\|\underline{\boldsymbol{x_j}} - \underline{\boldsymbol{x_K}}\|)$$

NOTE: here $\underline{\boldsymbol{x_i}}, \underline{\boldsymbol{x_j}}$ are vectors in R^N!

- It can we written in matrix form as:

$$\boldsymbol{f} = \boldsymbol{M} \cdot a$$

where: $\mathrm{m_{i,j}} = \Phi(\|\underline{\boldsymbol{x_j}} - \underline{\boldsymbol{x_i}}\|), \quad \boldsymbol{f} = [\mathrm{f}_1, f_2, \dots, f_k]^T$

$$\boldsymbol{a} = (\boldsymbol{M})^{-1} \cdot \boldsymbol{f}$$

Solution: weights of base functions

Interpolation with RBF's

- Once we know the coefficients a_j we can evaluate the function at arbitrary point $\underline{\boldsymbol{x}} = (x_1, x_2, \dots, x_N)$:

$$f(\underline{\boldsymbol{x}}) = \sum_{j=1}^{K} a_j \Phi_j(\underline{\boldsymbol{x}}) = \sum_{j=1}^{K} a_j \, \Phi\left(\left\|\underline{\boldsymbol{x}_j} - \underline{\boldsymbol{x}}\right\|\right)$$

- Each evaluation requires a computation of a series over all basis functions.

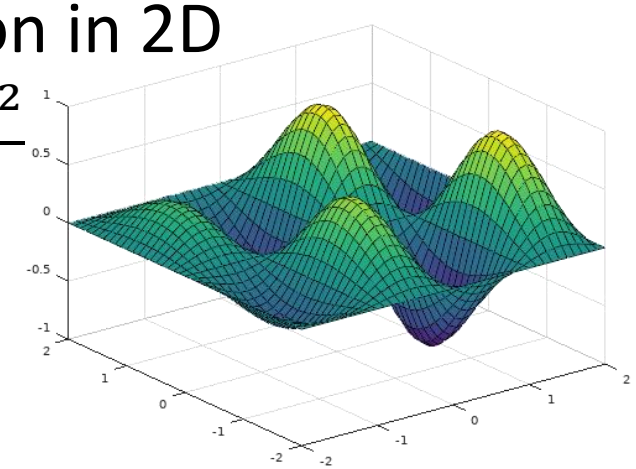
Interpolation with RBF's

- Solution of linear problem:
  - Matrix ***M*** is symmetric, real valued:
    - Store half (upper triangular) of the matrix
      - Dedicated direct solvers for dense symmetric matrices
    - Problem is positive-definite
      - Guaranteed cenvergence with iterative solvers
  - Conditioning of the matrix depends on the sampling points – it detoriates if points are very close each other
  - We can influence the conditioning & accuracy with manipulating parametr β (in selected RBF's)

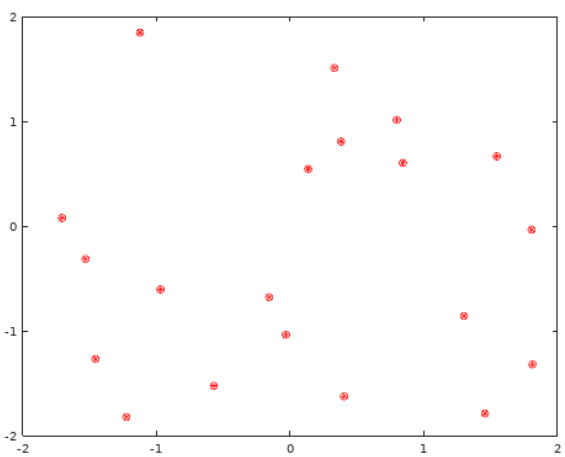
Interpolation in R^2

- Example: scattered samples of function in 2D

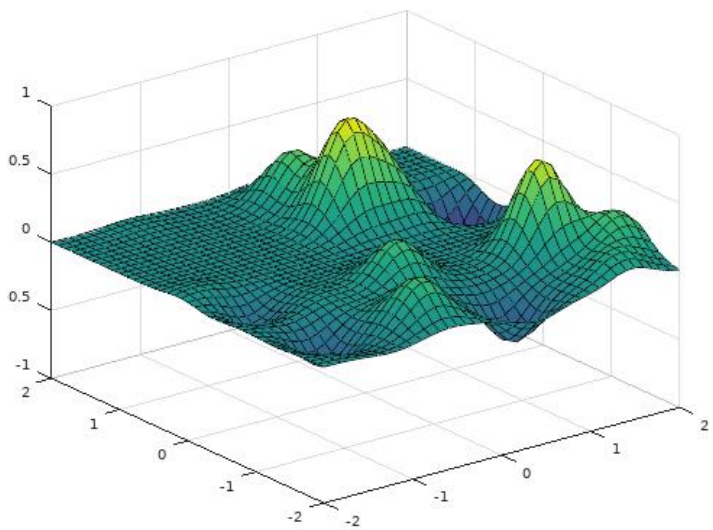
$$f(x,y) = \sin(\pi x) \cdot \sin\left(\frac{\pi}{2} y\right) \cdot e^{-\frac{x^2+y^2}{6}}$$



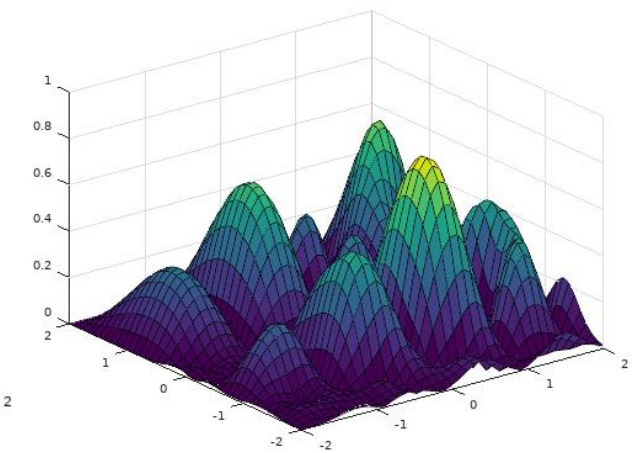
- Gaussian, $\beta = 3$
- K = 20



nodes



interpolation

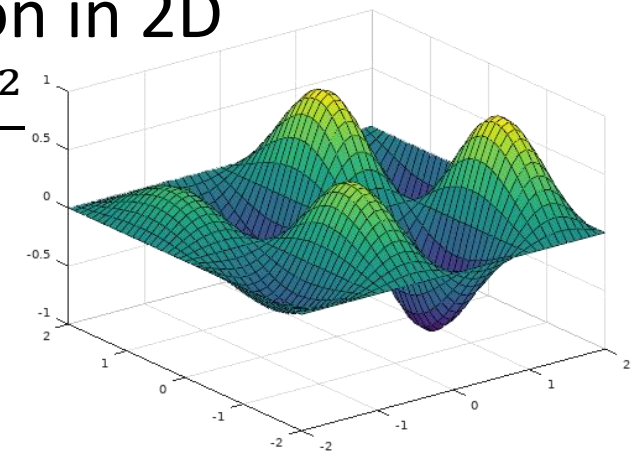


error

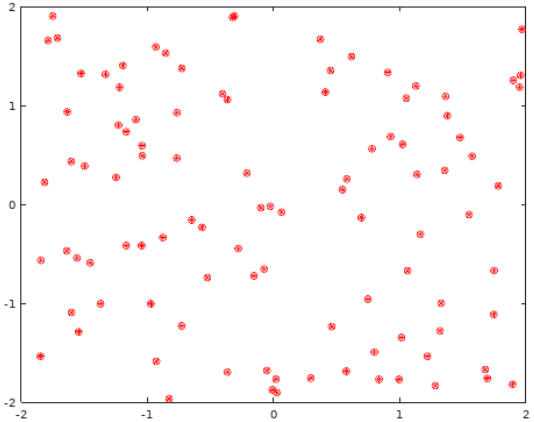
Interpolation in R^2

- Example: scattered samples of function in 2D

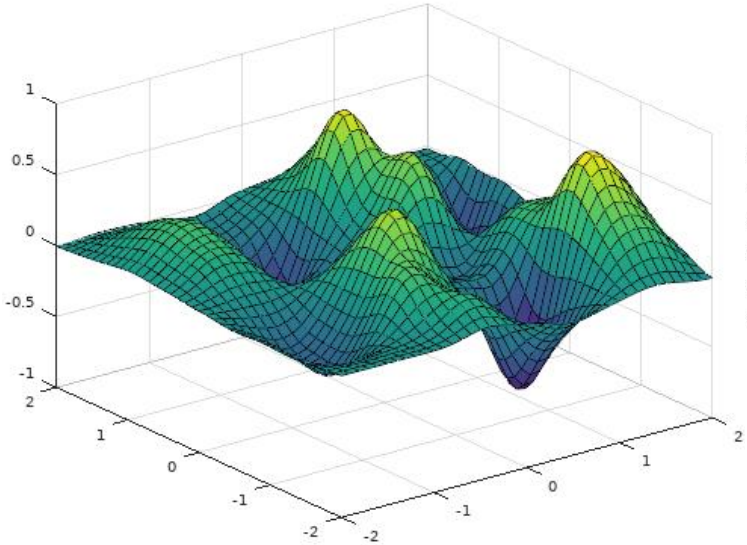
$$f(x,y) = \sin(\pi x) \cdot \sin\left(\frac{\pi}{2} y\right) \cdot e^{-\frac{x^2+y^2}{6}}$$



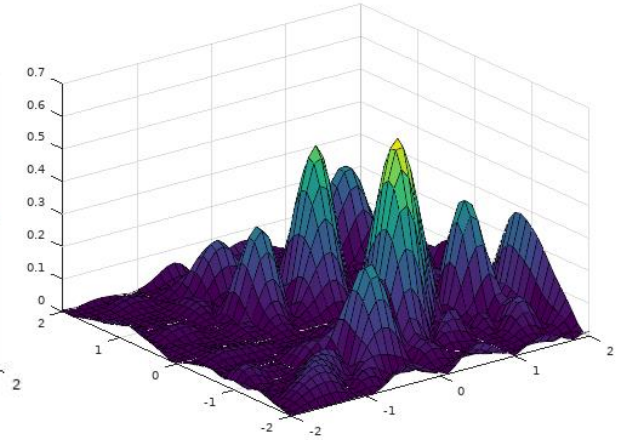
- Gaussian, $\beta = 3$
- K = 100



nodes



interpolation

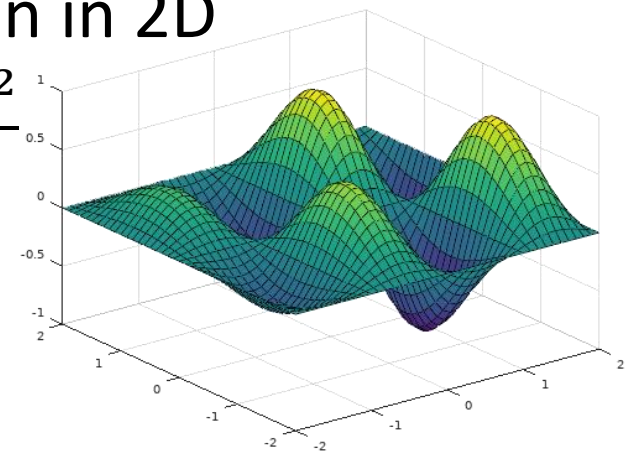


error max = 0.58

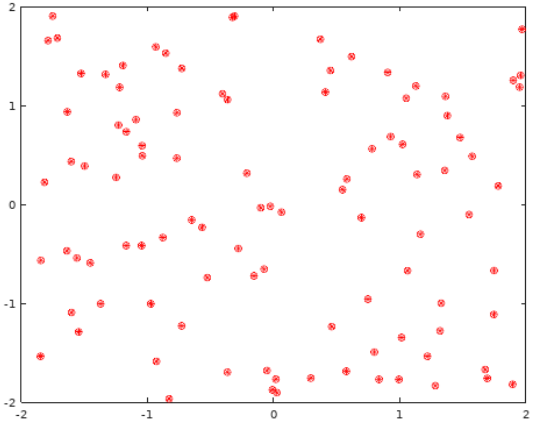
Interpolation in R^2

- Example: scattered samples of function in 2D

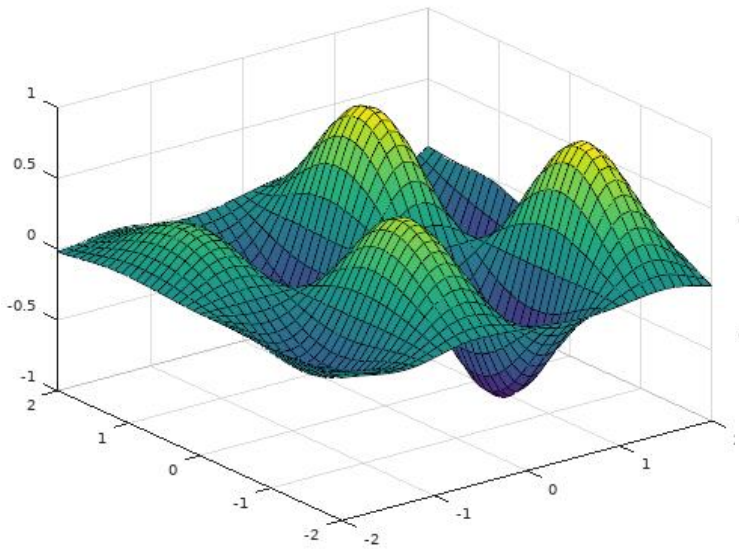
$$f(x,y) = \sin(\pi x) \cdot \sin\left(\frac{\pi}{2} y\right) \cdot e^{-\frac{x^2+y^2}{6}}$$



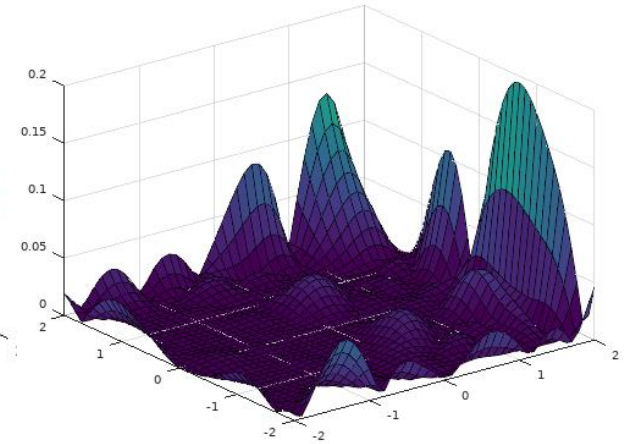
- Gaussian, $\beta = 1.5$
- K = 100



nodes



interpolation

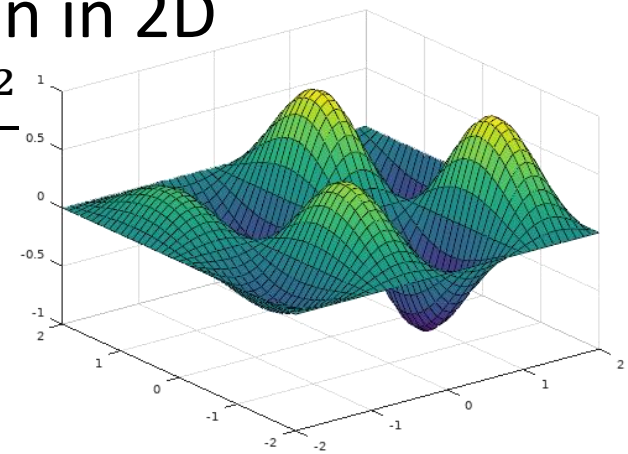


error max = 0.19

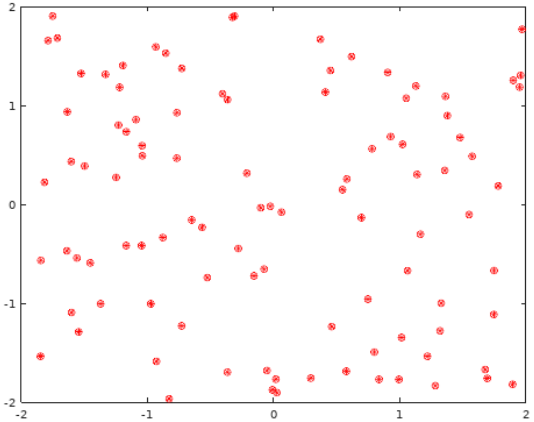
Interpolation in R^2

- Example: scattered samples of function in 2D

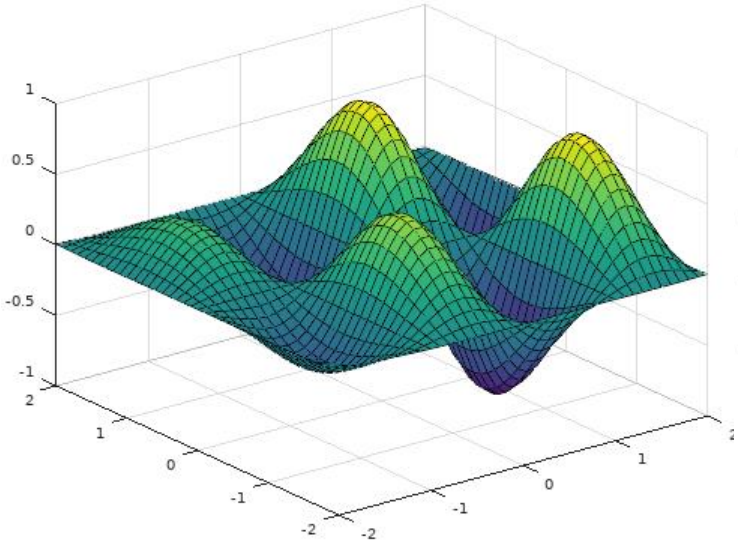
$$f(x,y) = \sin(\pi x) \cdot \sin\left(\frac{\pi}{2} y\right) \cdot e^{-\frac{x^2+y^2}{6}}$$



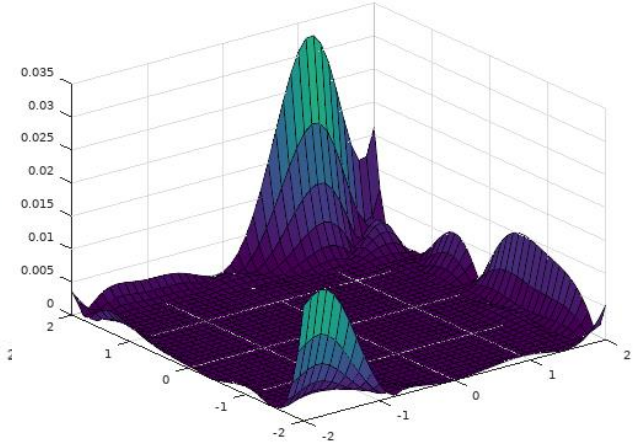
- Gaussian, $\beta = 0.75$
- K = 100



nodes



interpolation



error max = 0.03

Interpolation in R^N

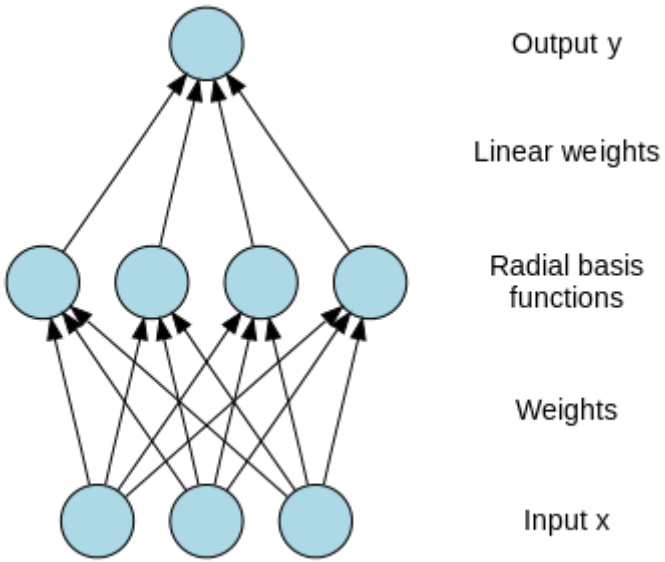
- We can easily increase the number of dimensions
- BUT:
  - As the number of samples grows, the size of the matrix grows
    - If K = 1e5 -> Memory to store matrix = K*K*8B =~ 75GB, not solveable (!)
  - To evalute the value of the function we compute sum of a series

$$f(x) = \sum_{j=1}^{K} a_j \Phi\left(\left\|\underline{x_j} - \underline{x}\right\|\right) \quad \rightarrow \quad \text{MANY TERMS IN THIS SERIES !}$$

- How to improve the scheme?
  - Use a special basis functions that are local, i.e. are equal zero for $r > r_0$
    - Then we limit the number of base functions in summation
    - The matrix ***M*** becomes sparse matrix – special fast linear solvers (Intel MKL PARDISO)
  - Use GPU to evaluate the series

Relation with RBF neural networks

- If we look artificial neural network that uses radial basis functions as activation functions:



$$\varphi(\mathbf{x}) = \sum_{i=1}^{N} a_i \rho(||\mathbf{x} - \mathbf{c}_i||)$$

Exactly the same form as our interpolation scheme!

Interpolation with RBF's

- In order to improve the quality of the intepolation & minimize the qurvature we can add an additional component:

$$f_i = f(\underline{x_i}) = \sum_{j=1}^{K} a_j \Phi\left(\left\|\underline{x_j} - \underline{x_i}\right\|\right) + P(\underline{x_i})$$

For example → $\underline{x_i} = [x_{i,1} \quad x_{i,2} \quad x_{i,3}]^T$

Low order polynomial

$$P(\underline{x_1}) = b_0 + b_1 \cdot x_{1,1} + b_2 \cdot x_{1,2} + b_3 \cdot x_{1,3}$$

3D biharmonic: adds 4 unknowns (b_1, b_2, b_3, b_4) to our system

In R^3 we can have $\underline{x} = [x\ y\ z]^T$

Combining polynomial P(x)

- To restore the symmetry of the system when we add P(x) an additional condition is introduced (in R^3):

$$\sum_{i=1}^{K} a_j = \sum_{i=1}^{K} a_j x_{j,1} = \sum_{i=1}^{K} a_j x_{j,2} = \sum_{i=1}^{K} a_j x_{j,3} = 0$$

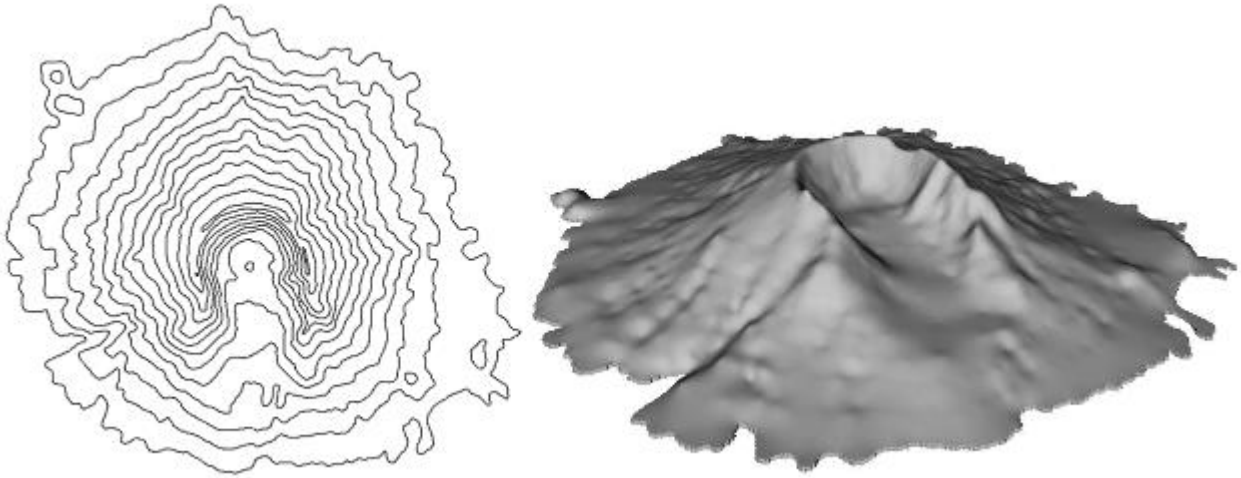
4 additional equations

- Final system to solve:

$$\begin{bmatrix} \overline{\mathrm{m}_{1,1}} & \dots & \overline{\mathrm{m}_{1,K}} & 1 & x_{1,1} & x_{1,2} & x_{1,3} \\ & \ddots & & & & & \\ \overline{\mathrm{m}_{K,1}} & \dots & \overline{\mathrm{m}_{K,K}} & 1 & x_{K,1} & x_{K,2} & x_{K,3} \\ 1 & \dots & 1 & 0 & 0 & 0 & 0 \\ x_{1,1} & \dots & x_{K,1} & 0 & 0 & 0 & 0 \\ x_{1,2} & \dots & x_{K,2} & 0 & 0 & 0 & 0 \\ x_{1,3} & \dots & x_{K,3} & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ \vdots \\ a_K \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_K \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Applications

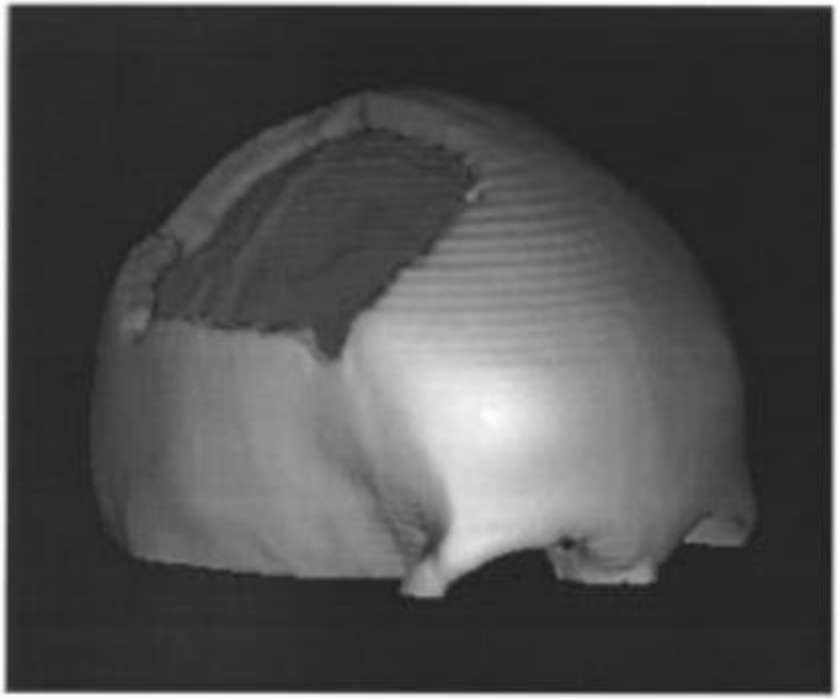
- Digital terrain modelling

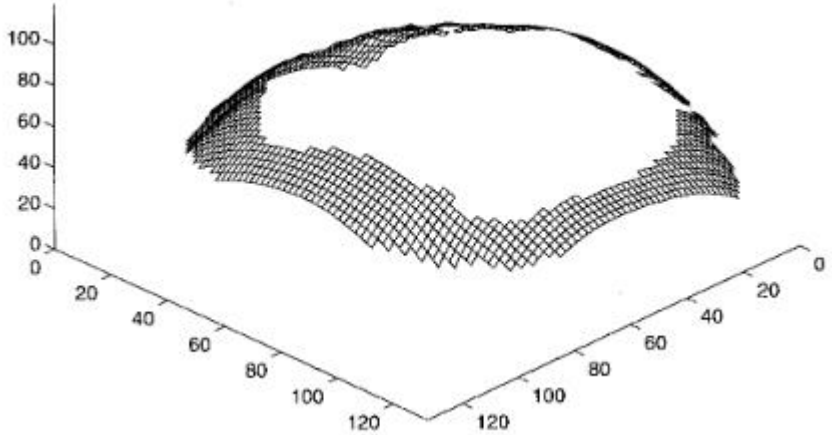


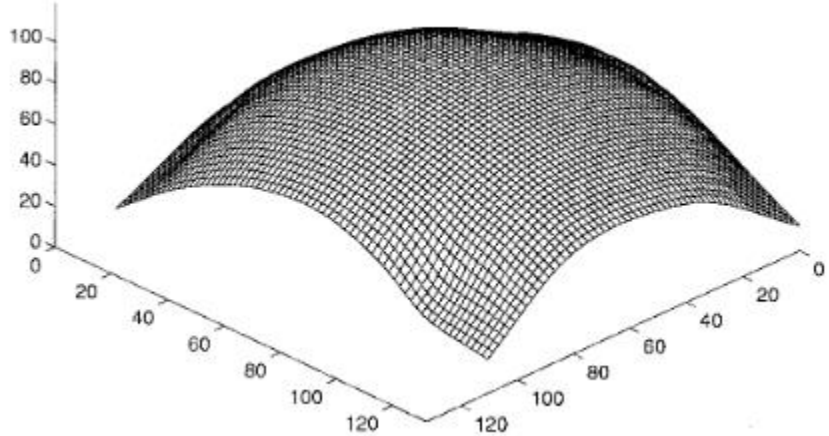
Joachim Pouderoux, Ireneusz Tobor, Jean-Christophe Gonzato, Pascal Guitton. Adaptive Hierarchical RBF Interpolation for Creating Smooth Digital Elevation Models. Proceedings of the Twelfth ACM International Symposium on Advances in Geographical Information System 2004, 2004, United States. pp.232–240, 2004. <hal-00308008>

Applications

- Medical data extrapolation







Surface Interpolation with Radial Basis Functions for Medical Imaging

Jonathan C. Carr,* W. Richard Fright, *Member, IEEE*, and Richard K. Beatson

Applications

- Image inpainting / restoration

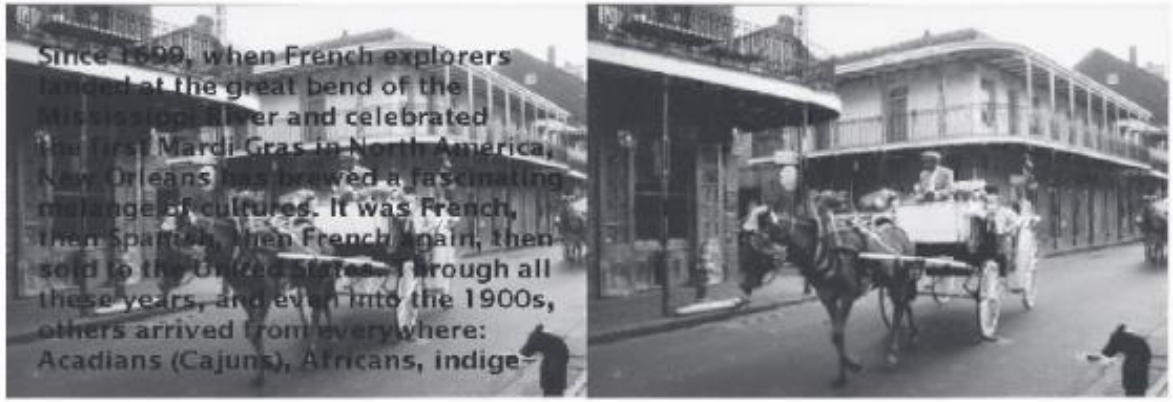


Figure 3. Inpainted image and its reconstruction.

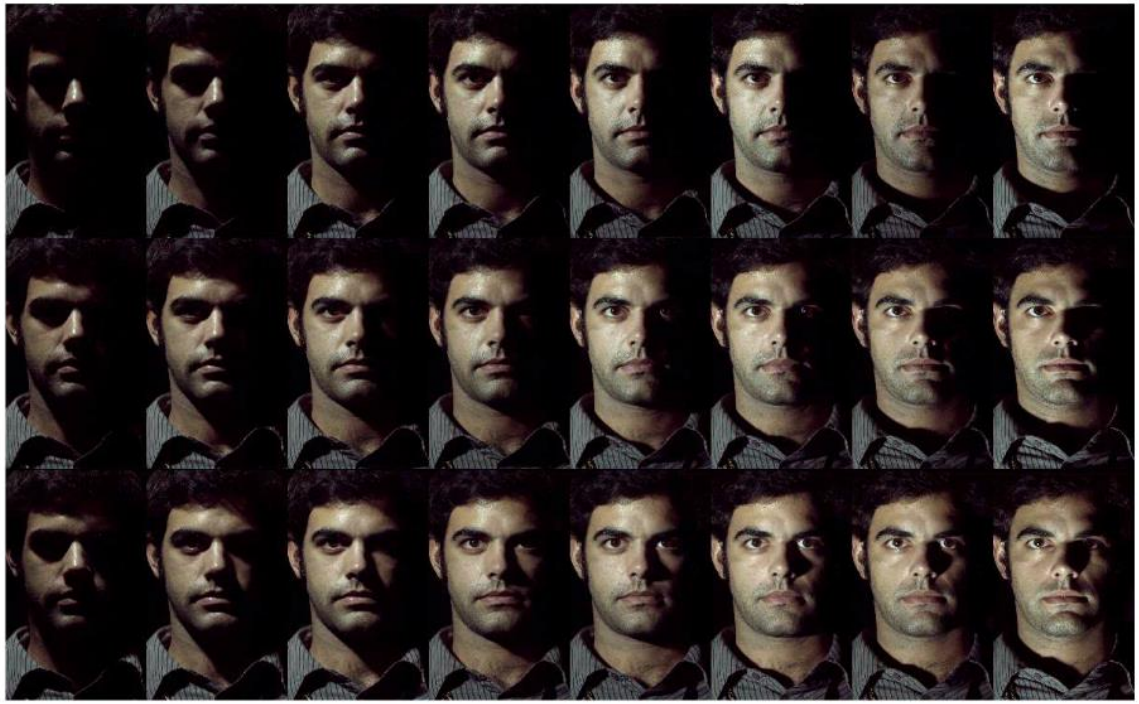
Figure 4. Scaled parts of images.

RADIAL BASIS FUNCTION USE FOR THE RESTORATION OF DAMAGED IMAGES

Karel Uhlir, Vaclav Skala
University of West Bohemia, Univerzitni 8, 30614 Plzen, Czech Republic*

Applications

- Image editing – lightning manipulation



Interactive Lighting Manipulation Application on GPU

Borom Tunwattanapong Paul Debevec
University of Southern California Institute for Creative Technologies

Applications

- Reconstruction of 3D models from scattered datapoints

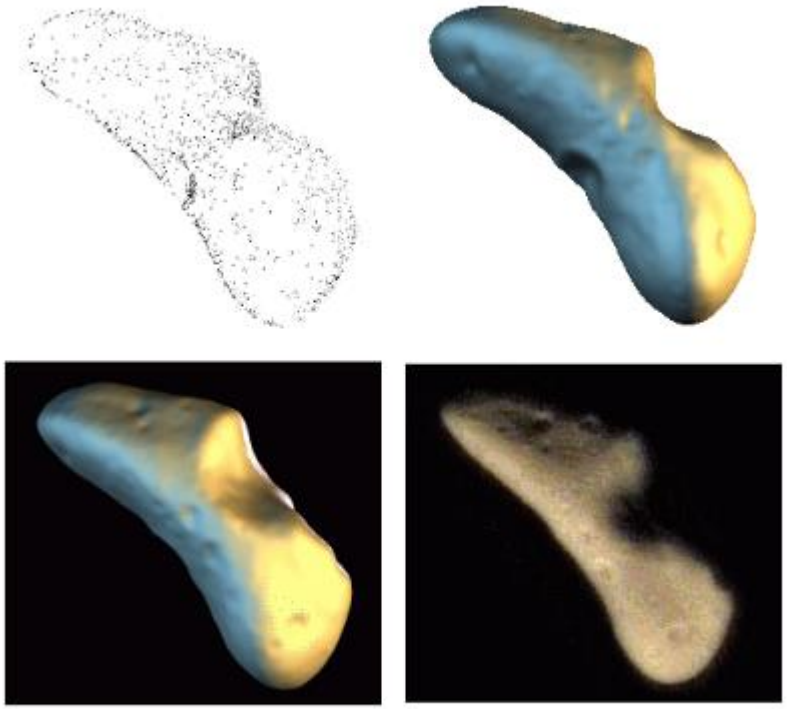
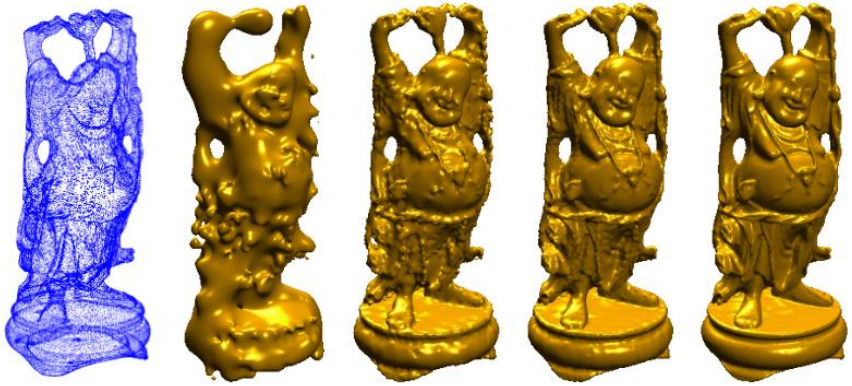


Figure 13: RBF reconstruction of the asteroid Eros from non-uniformly distributed range data (top). Photograph and model from a similar view (bottom).



Q: How it is done?

A:

- Create a RBF model of data

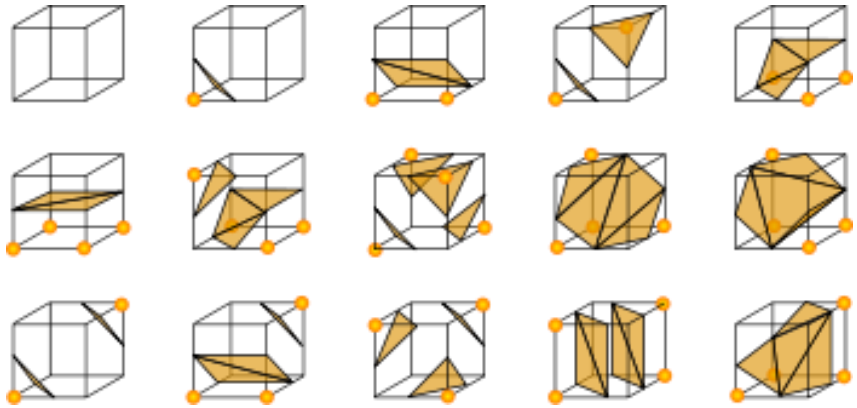
$$
\begin{aligned}
f(x_i,y_i,z_i) &= 0, & i &= 1,\ldots,n & &(\textit{on}\text{-surface points}),\\
f(x_i,y_i,z_i) &= d_i \neq 0, & i &= n+1,\ldots,N & &(\textit{off}\text{-surface points}).
\end{aligned}
$$

- Search for isosurfaces

Jonathan C. Carr, Richard K. Beatson, Jon B. Cherrie, Tim J. Mitchell, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and representation of 3d objects with radial basis functions. *Proceedings of SIGGRAPH 2001*, pages 67–76, August 2001.

Marching cubes

- Reconstruction of 3D models from scattered datapoints
  - How can we search for isosurfaces?
- Marching cubes algorithm
  - 3D cube – 8 nodes – 256 combinations of positive/negative value at each node
  - Only 15 are unique:



  - Create cartesian grid that encloses our object / (scalar field)
  - We check for values at cubes corners and choose proper set of triangles to render
  - More advanced: adaptive marching cubes

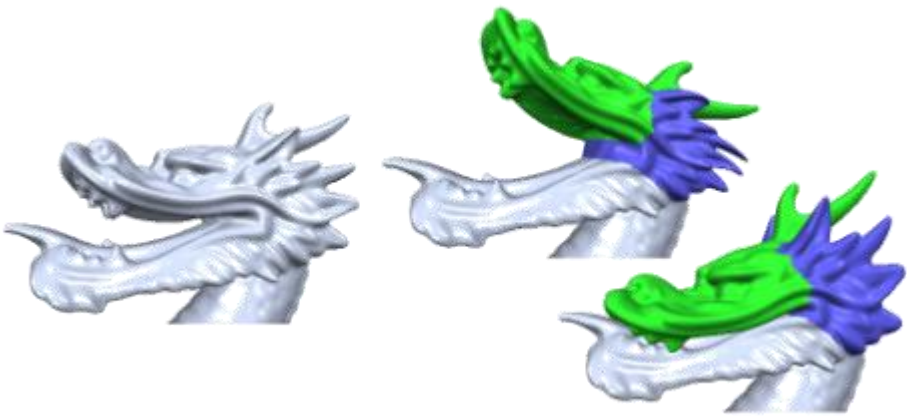
Applications of RBF's

- Computer graphics: mesh deformation techniques
- Interpolation function represent the deformation of the mesh nodes (not the mesh itself!)
  - Imagine we have an aribrary mesh
  - Modify placement of some (selected) mesh nodes (control points) to create a displacement function **d(<u>x</u>)**
  - Displace all the mesh nodes according to displacement function **d(<u>x</u>)**

Applications

- Mesh deformation techniques

Figure 5: *A bowing deformation of a scanned Chinese statue by bending its back and neck.*



Real-Time Shape Editing using Radial Basis Functions

Mario Botsch Leif Kobbelt

Computer Graphics Group
RWTH Aachen University

Applications

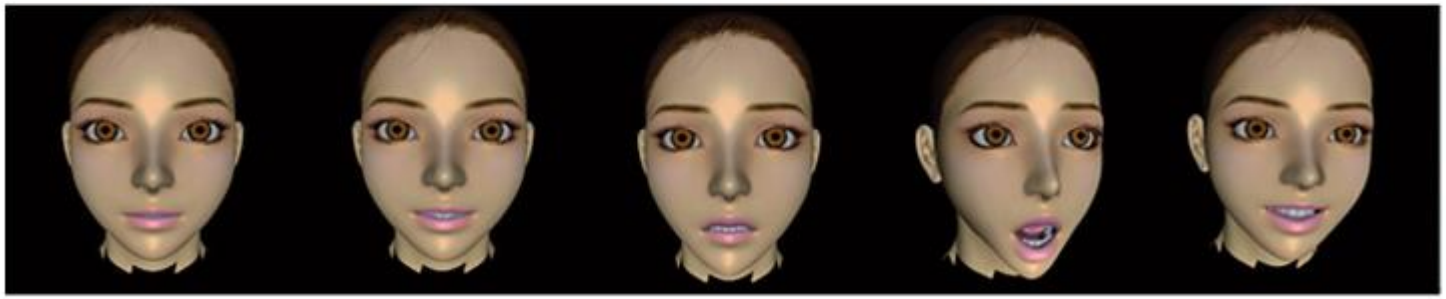
- Mesh deformation techniques

https://www.youtube.com/watch?v=L5rUF17G-_Y

Applications

- Face animations

(a)



(b)

Geodesic Distance-Based Realistic Facial Animation Using RBF Interpolation

Xianmei Wan; Shengjun Liu; Jim X. Chen; Xiaogang Jin

Computing in Science & Engineering

Year: 2012, Volume: 14, Issue: 5

Applications

- Engineering – CAD - shape optimization
  - Volumetric mesh deformation in numerical simulations

  - Radial neural networks (ANN) with radial activity functions….

That’s all!

Possible exercises at home:

- Try to implement a simple RBF interpolation scheme in 2D and apply it to image deformation

- Try to implement a simple marching squares algorithm (2D version of marching cubes) to find isobars for arbitrary function $z = f(x, y)$

Modeling and simulation of systems

Stochastic methods

Adam Lamecki, WETI PG, Dec. 2023

Introduction

- In previous lectures we shown how to solve partial differential equations with finite differences (FD) and finite element methods (FEM)

- In this lecture we will introduce an alternative technique that is based on statistical approach to solve complex problems:

Monte Carlo Methods

Basic Monte Carlo

- The basic idea behind Monte-Carlo Methods is to propose a stochastic process, for which its statistics (like mean value of random variable) gives one requested solution.
- Example: Hit & miss

Basic Monte Carlo : Integration

- The concept of Monte-Carlo metod is often introduced using example of evaluation of a 1D integral:

$$c = \int_a^b f(x)dx$$

- Instead of direct (analytical) solution we can:
  - Generate a random variable $x_i \sim U[a, b]$
  - Compute the values of function $f(x_i)$
  - Then we can estimate the value of *c* as:

$$\tilde{c} = \frac{(b-a)}{N} \sum_{i=1}^{N} f(x_i)$$

Basic Monte Carlo : Integration

- The same concept can be generalized to multidimensional integrals:

$$c = \int_{\Omega} f(\boldsymbol{x}) d\Omega$$

- We can estimate the value of c taking random point inside of Ω and compute estimate as $\hat{c} = \hat{f} \cdot \Omega = \hat{f} \rightarrow$ huge error
- We can repeat the experiment N-times and compute mean:

$$\hat{c} = \left(\hat{f}_1 + \hat{f}_2 + \hat{f}_3 + \cdots + \hat{f}_N\right) \cdot \frac{\Omega}{N}$$

- If we use the set of points that are uniformely scattered inside Ω, then we get good estimation

Monte Carlo Integration

- What is the error?
- The mean of the function $f(x)$ can be estimated as

$$\bar{f}_{est} = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

- Then the variance of the mean is:

$$Var(\bar{f}_{est}) = \frac{\sigma^2}{N}$$

where σ^2 is the variance of $f(x)$

- We can estimate the error as the standard deviation:

$$e = \frac{\sigma}{\sqrt{N}}$$

- In order to reduce the error by factor of 10 we need to increase N by factor of 100 !!!

Monte Carlo Integration

- Finally, the erorr of the Monte Carlo integral can be estimated

$$e = \sigma_c = \frac{\hat{\sigma}_f}{\sqrt{N}}$$

- In order to reduce the error by factor of 10 we need to increase N by factor of 100 !!!
- Example: $\int_0^1 e^x dx$:

N	$\hat{c}$	σ_c
10	1.726607081406647	0.097088755845900
100	1.712461840129966	0.052104648661501
1 000	1.694539476115200	0.015495203418794
10 000	1.717033546757652	0.004915856081510
100 000	1.716746281429447	0.001551825593091
1 000 000	1.717840368230083	0.000492290558784
10 000 000	1.718293866461716	0.000155571844221

Abs(True error)
0.008325252947601
0.005819988329079
0.023742352343846
0.001248281701393
0.001535547029599
0.000441460228963
0.000012038002670

Monte Carlo & Antithetic sampling

- A huge number of function evaluations is needed to achieve low error of the estimated integral – NAIVE APPROACH (!)
- We can modify the scheme to improve the prerformance applying the **<u>variance reduction techniques</u>**:

Antithetic sampling:

- If c_1 and c_2 are random variables with the same distribution then

$$Var(c) = Var\left(\frac{c_1 + c_2}{2}\right)$$

$$Var(c) = \frac{Var(c_1) + Var(c_2) + 2Cov(c_1, c_2)}{4}$$

- If $\mathrm{C}ov(c_1, c_2) < 0$, then we reduce variance of c (!)

- Notice that if random variable $x_i \sim U[0,1]$ then also $x_i' = 1 - x_i$ belongs to $U[0,1]$

Monte Carlo & Antithetic sampling

- The antithetic sampling estimate of c is:

$$\hat{c}_{anti} = \frac{1}{N}\sum_{i=1}^{N/2}(f(x_i) + f(x_i'))$$

- The same example of $\int_0^1 e^x dx$, but now use antithetic sampling:

N	$\hat{c}$	σ_c	True error (abs)	Prev.
10	1.734097248189966	0.036232402252012	0.015815419730920	0.008325
100	1.713841590386055	0.008237499878296	0.004440238072990	0.005819
1 000	1.719533774343124	0.002903936118719	0.001251945884079	0.023742
10 000	1.718845839324546	0.000886509899895	0.000564010865501	0.001248
100 000	1.717939226621344	0.000280679976106	0.000342601837702	0.001535
1 000 000	1.718189879489733	0.000088407209923	0.000091948969313	0.000441
10 000 000	1.718297286794053	0.000027973650102	0.000015458335007	0.000012

Monte Carlo & Stratified sampling

- Alternative technique for
- We can divide the domain intovariance reduction *M non-overlapping strata (subgroups)* (*pl.: warstwy)* and take L samples from each one
- Compute the average in each strata
- Estimate the result from the averages
- With this:
  - Prevent from samples „clustering”
  - Can apply adaptive sampling – monitoring the variance in each subinterval

Advanced Monte Carlo

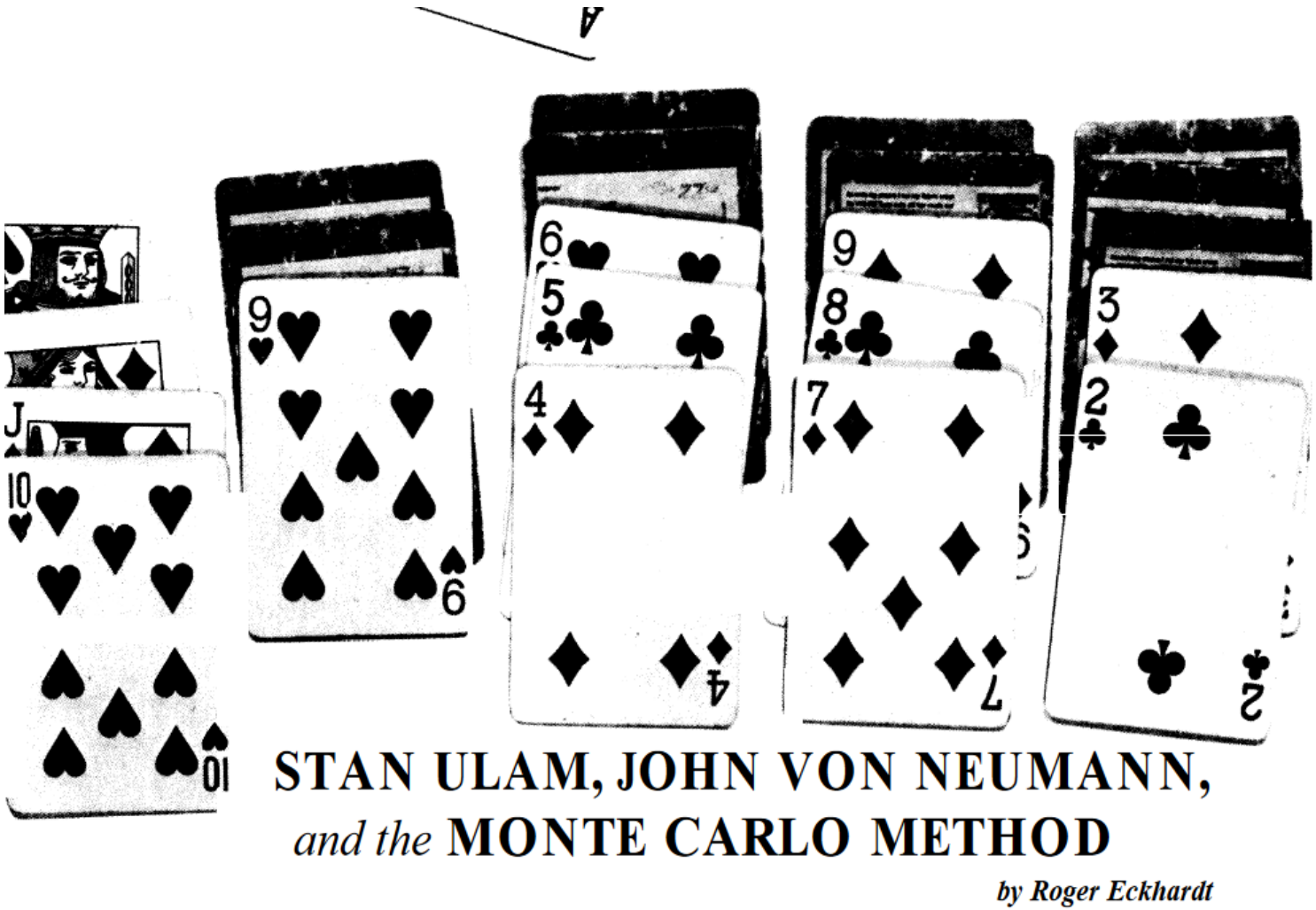
- It is not a simple integration rule (even multidimensional one) that made a Monte-Carlo method famous since 1940's

- In fact, the breakthrough applications of MC method was made by Polish/American scientist **Stanisław (Stan) Ulam** who was working in Los Alamos laboratories and was involved in Manhattan Project

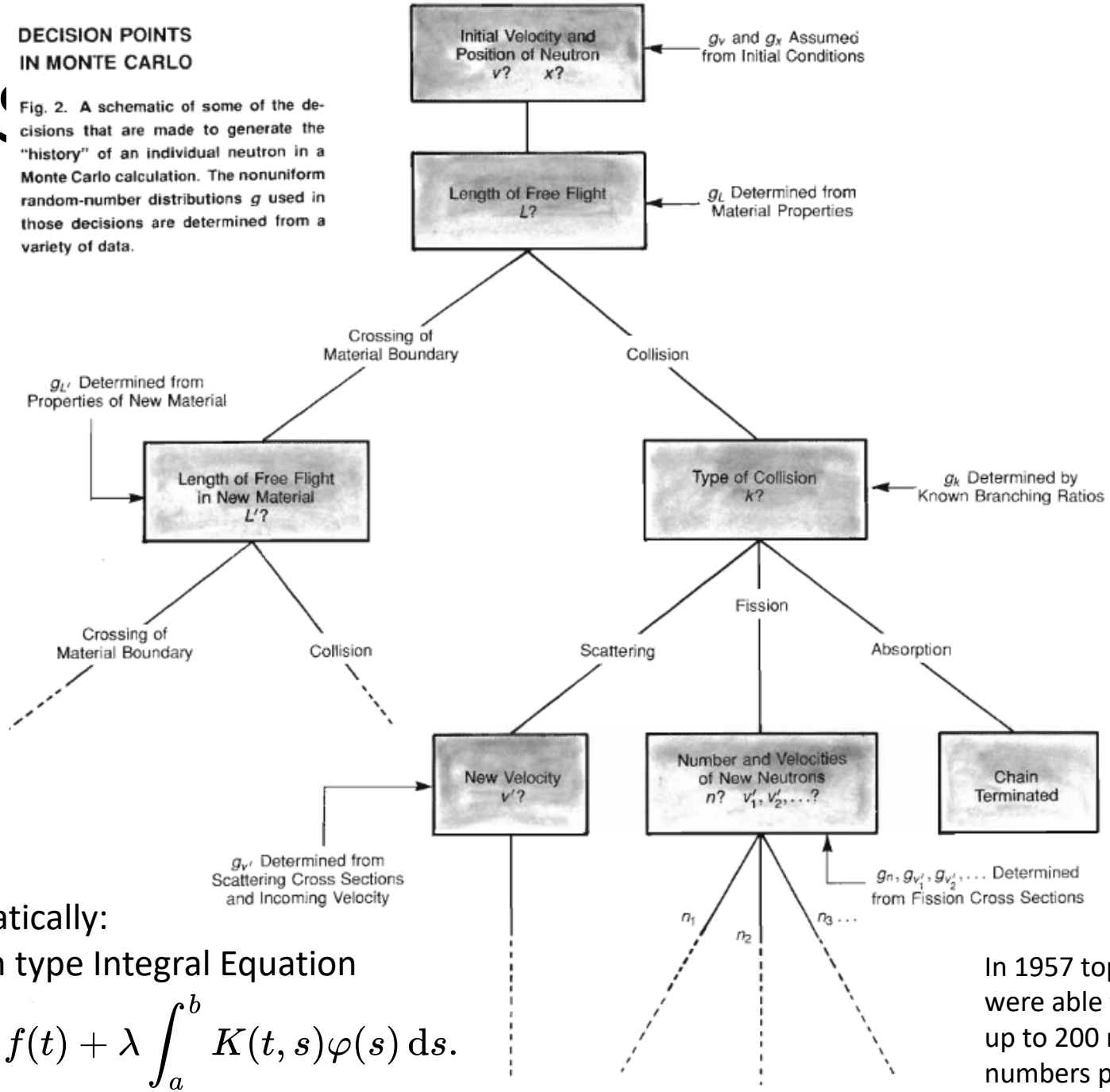
- He, along with **John von Neumann** was one of the first to recognize the possible application of computers to solve complex numeric problems

Advanced Monte Carlo

Roger Eckhardt, “Stan Ulam, John Von Neumann and the Monte Carlo method”, Los Alamos Science, Special issue, 1987, 131-137



n



Mathematically:
Fredholm type Integral Equation

$$\varphi(t) = f(t) + \lambda \int_a^b K(t,s)\varphi(s)\,\mathrm{d}s.$$

In 1957 top machines were able to generante up to 200 random numbers per second (!)

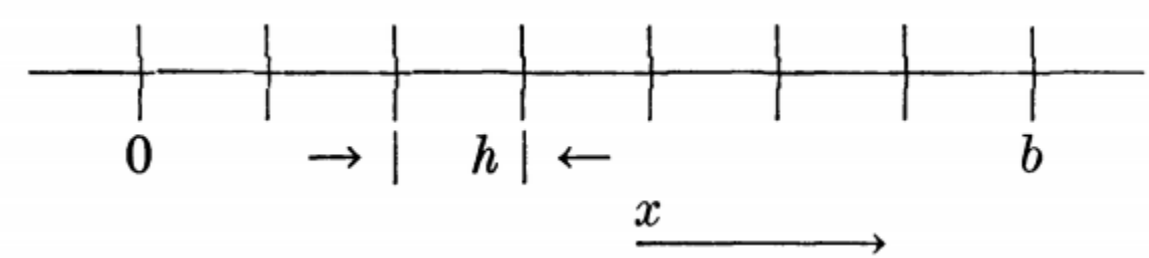
Advanced Monte-Carlo

„…Pomysł ten, nazwany później metodą Monte Carlo, wpadł mi do głowy, kiedy podczas choroby stawiałem pasjanse. Zauważyłem , że znacznie praktyczniejszym sposobem oceniania prawdopodobieństwa ułożenia pasjansa jest wykładanie kart, czyli eksperymentowanie z tym procesem i po prostu zapisywanie procentu wygranych, niż próba obliczenia wszystkich możliwości kombinatorycznych, których liczba rośnie wykładniczo i jest tak wielka, że pominąwszy najprostsze przypadki, jej oszacowanie jest niemożliwe. Jest to zaskakujące z intelektualnego punktu widzenia, i choć może nie całkiem upokarzające, to jednak zmusza do skromności i pokazuje granice tradycyjnego, racjonalnego rozumowania. ***Jeśli problem jest wystarczająco złożony, próbowanie jest lepszym sposobem niż badanie wszystkich łańcuchów możliwości****…”*

Stanisław Ulam

Simple example

- Consider a game for two players (G1,G2), consisting a series of turns
  - G1 has probability to win equal *p*
  - *G2* has probability to win equal *q* = 1 – *p*
  - Stake of each of each turn is $h
  - At the beginning G1 has $x and G2 has $y (in total they have $b = $x + $y



  - The play finishes if one of players reaches $0 (x=0 or x=b)

Based on: W.F. Bauer - The Monte Carlo Method, 1958

Simple example

- The probability *V(x)* that G1 wins if he starts with $x is the solution of the equation written as:

$$\frac{V(x+h)+V(x-h)-2V(x)}{h^2}+\frac{p-q}{qh}\left[\frac{V(x+h)-V(x-h)}{h}\right]=0$$

with boundary conditions V(0)=0, V(b)=1

- We can compare with the approximations of derrivatives:

Three approximations to the derivative $f'(a)$ are

1. the one-sided (forward) difference $\frac{f(a+h)-f(a)}{h}$
2. the one-sided (backward) difference $\frac{f(a)-f(a-h)}{h}$
3. the central difference $\frac{f(a+h)-f(a-h)}{2h}$

Based on: W.F. Bauer - The Monte Carlo Method, 1958

Simple example

- The probability *V(x)* that G1 wins if he starts with $x is the solution of the equation written as:

$$\frac{V(x+h)+V(x-h)-2V(x)}{h^2}+\frac{p-q}{qh}\left[\frac{V(x+h)-V(x-h)}{h}\right]=0$$

with boundary conditions V(0)=0, V(b)=1

- We can compare with the approximations of derrivatives:

Second-order central

$$f''(x)\approx\frac{f(x+h)-2f(x)+f(x-h)}{h^2}$$

Simple example

- The probability *V(x)* that G1 wins if he starts with $x is the solution of the equation written as:

$$\frac{V(x+h)+V(x-h)-2V(x)}{h^2}+\frac{p-q}{qh}\left[\frac{V(x+h)-V(x-h)}{h}\right]=0$$

with boundary conditions V(0)=0, V(b)=1

- Now if $h \to 0$, and $p - q = h \cdot \alpha + 0(h)$ (i.e. *p* and *q* are becoming equal), the finite difference problem transforms to:

$$\frac{d^2u}{dX^2}+2\alpha\frac{du}{dx}=0$$

$$u(0)=0, \qquad u(b)=1.$$

Simple example

- To solve the differential equation for arbitrary x on domain (0,1), we perform a series of experiments (games)

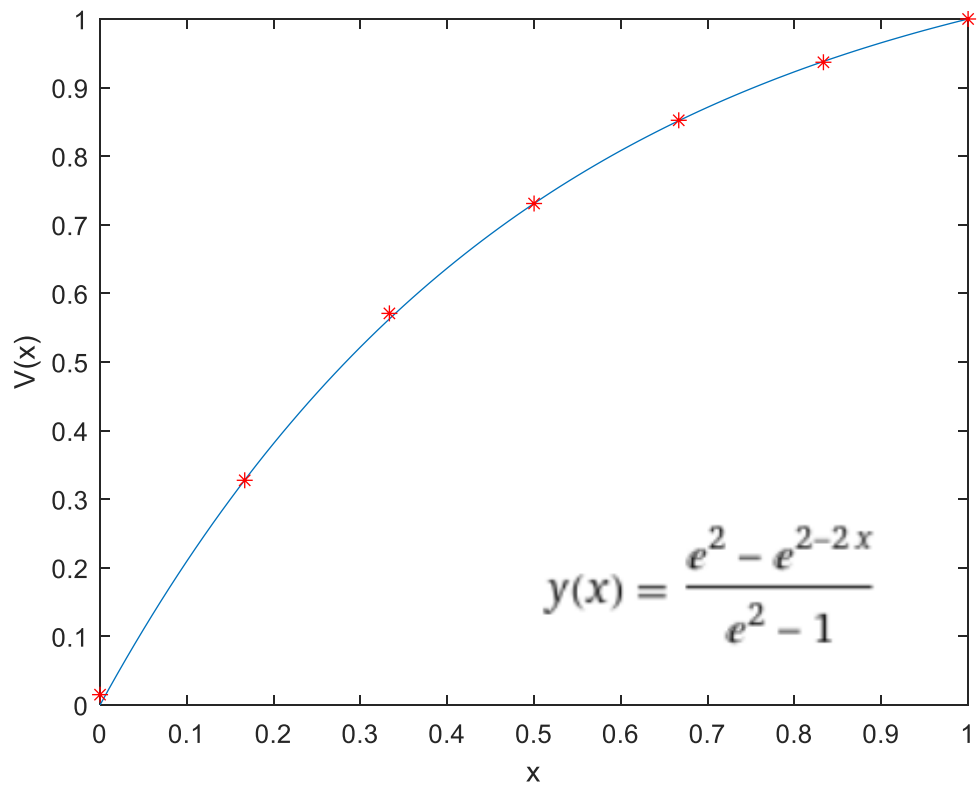
- The expected value is our missing solution u(x)

Simple example

- Matlab code & output

```
% Monte-Carlo solution of equation y"+2*y'=0 with y(0)=0 and y(1)=1
clear all;

N = 10000;  % number of trials in single MC
h = 0.01;   % step

p = 0.5 + h/2;
q = 1 - p;
b = 1;  % max x;

K=7;     % number of x-points in our MC analysis
xx = linspace(0,1,K);
for k=1:K
    x = xx(k);
    G1 = 0; G2 = 0;
    for d=1:N
        while (x>=0) & (x<b)
            r = rand(1);
            if (r<=p)
                x = x + h;
            else
                x = x - h ;
            end
        end
        if (x<=0)
            G1 = G1+1;  % count the failure rate of G1
        else
            G2 = G2+1;  % count the failure rate of G2
        end
        x = xx(k);
    end
    V(k) = G2/N;   % compute probability of success of G1;
end
```

```
% reference solution (analytic) - Wolfram Alpha ;-)
x1 = linspace(0,1,101);
e = exp(1);
s = (e*e - exp( 2 - 2*x1) )/ (e*e-1);
plot(x1,s);
hold on;
plot(xx, V,'r*');
```



Error analysis

- Solve for x = 0.5, h=0.1, K=10 trials
- Exact solution: y(x) = 0.7310585

N	V(0.5)	Rel. Error [%]
10	0.79	8.06
100	0.732	0.129
1000	0.7331	0.279
10000	0.73091	0.0203
100000	0.731121	0.0085384

Advanced Monte Carlo

One can apply the **Feynman-Kac formula** which states that given a general form of parabolic partial differential equation (PDE):

$$\frac{\partial u}{\partial t}(x,t) + \mu(x,t)\frac{\partial u}{\partial x}(x,t) + \frac{1}{2}\sigma^2(x,t)\frac{\partial^2 u}{\partial x^2}(x,t) \ldots$$
$$\ldots - V(x,t)u(x,t) + f(x,t) = 0$$

where $x \in R, t \in [0,T]$ and $\mu, \sigma, V, f, u(x,T) = \phi(x)$ are known, its solution can be written as a conditional expectation:

$$u(x,t) = E^Q\left[\int_t^T e^{-\int_t^r V(X_\tau,\tau)\,d\tau} f(X_r,r)dr + e^{-\int_t^T V(X_\tau,\tau)\,d\tau}\psi(X_T)\middle| X_t = x\right]$$

under the probability measure such that X is an Ito process in form:

$$dX = \mu(X,t)dt + \sigma(X,t)dW^Q$$

with $W^Q(t)$ is a Brownian motion process….

… additional constraints are skipped here…

Advanced Monte Carlo

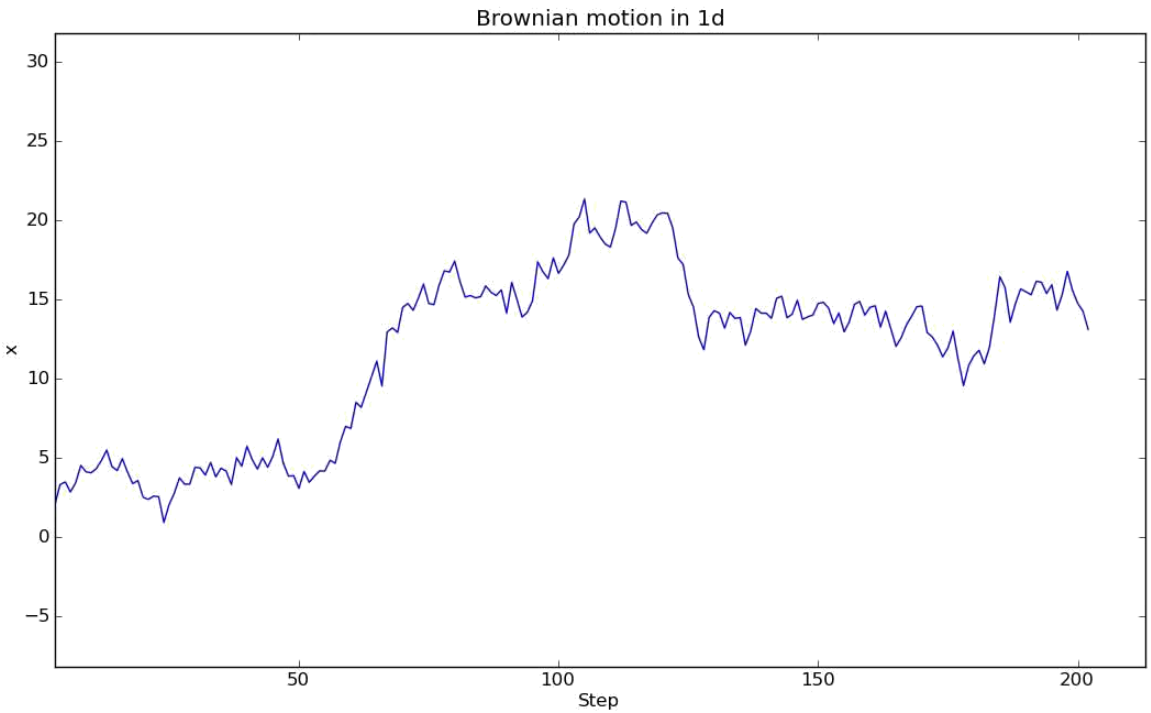
- Some consequences of Feynman-Kac formula:
  - The solution of complex, nonlinear, parabolic PDE can be organized as a stochastic game with expected vaule equal to the solution!

  BUT also:
  - A stochastic process that has a form that appears in Feynman-Kac formula can be solved as an analytic solution of properly defined PDE!

- Modern applications:
  - Solution of the Schrödinger Equation (quantuum physics)
  - A special case of this formula is applied to derive the Black-Scholes formula, the cornerstone of financial mathematics (option pricing)
  - ...

Brownian motion

Take a look at Brownian motion in 1D:



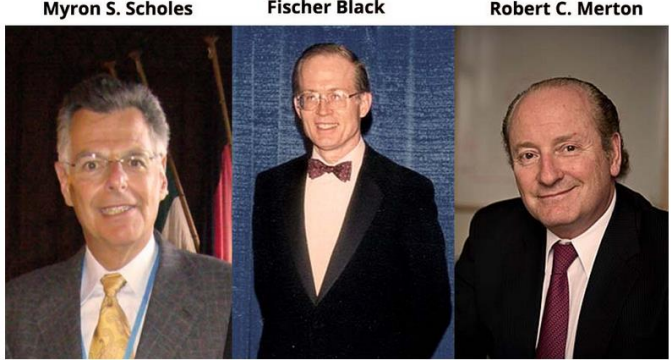
- The displacement at each step has a normal distribution with $\sigma = 1$

Black-Scholes model

- Now take a look at plot of futures of brent crude prices:

- One can notice that it is a stochastics process, similar to diffusion equation with Brownian motion

Black-Scholes model

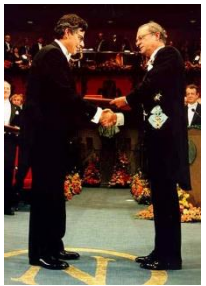


- In 1970's F. Black, M. Scholes and R.C. Merton developed a new model of options pricing that can be shown as a direct application of Feynman-Kac formula to the financial markets
  - Transform the stochastic process into PDE that allows to predict the future prices of options (financial instrument based on equity prices)

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

where: V is the price of the option as a function of stock price S and time t, r is the risk-free interest rate, and σ is the volatility of the stock

- Their solution is up to date treated as one of the most advanced models in area of financial engineering
- In 1997 Scholes and Merton received a Nobel Memorial Prize in Economics



Monte Carlo for Laplace problem

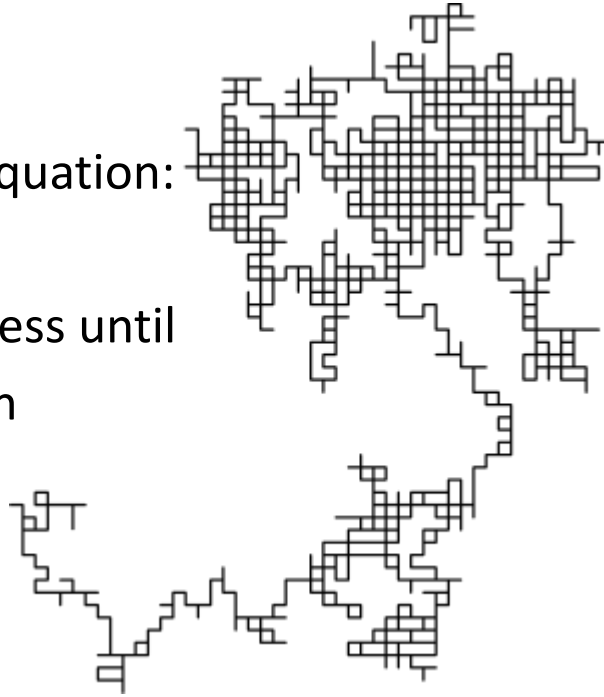
- In 2D we can consider a **random walk** – that is a discrete process
  - Random walk on equidistant 2D grid: probability of P =0.25 to move left/right/up or down
  - Lattice random walk

- For example, we can use the random walk process to solve a 2D Laplace equation using finite difference methods

- Imagine that we want to find a solution of Laplace equation:

$$\nabla^2 u = 0, \quad \Omega \in R^2, \; given \;\; u(x,y) \; on \; \delta\Omega$$

in one point of the grid we start a random walk process until we reach boundary of Ω where $u = u(x,y)$ is known



$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = 0.$$

Monte-Carlo for Laplace problem

- Let's write a central-difference approximation of 2D Laplace equation for $u_{i,j}$, where i,j are nodes on the grid:

$$u_{i,j} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}\right)$$

  - In this discrete form of Laplace equation the value of $u_{i,j}$ is computed as a mean of values at neighboring nodes.
- We start **a random walk** from point $u(i_0, j_0)$, until we reach the boundary:
  - As we move from point to point we know that the next point is also an average of its neighbors
  - We get a chain of steps until we encounter a boundary node „*q*" with known value of $u(q)$, $\rightarrow$ $q = 1 \ldots M$, where *M* is total number of known boundary points
  - Repeat this N-times, then we get an estimate for unknown $u(i_0, j_0)$:

$$u(i_0, j_0) \sim \frac{1}{N}\sum_{k=1}^{M} u(q_k) N_k \,,$$

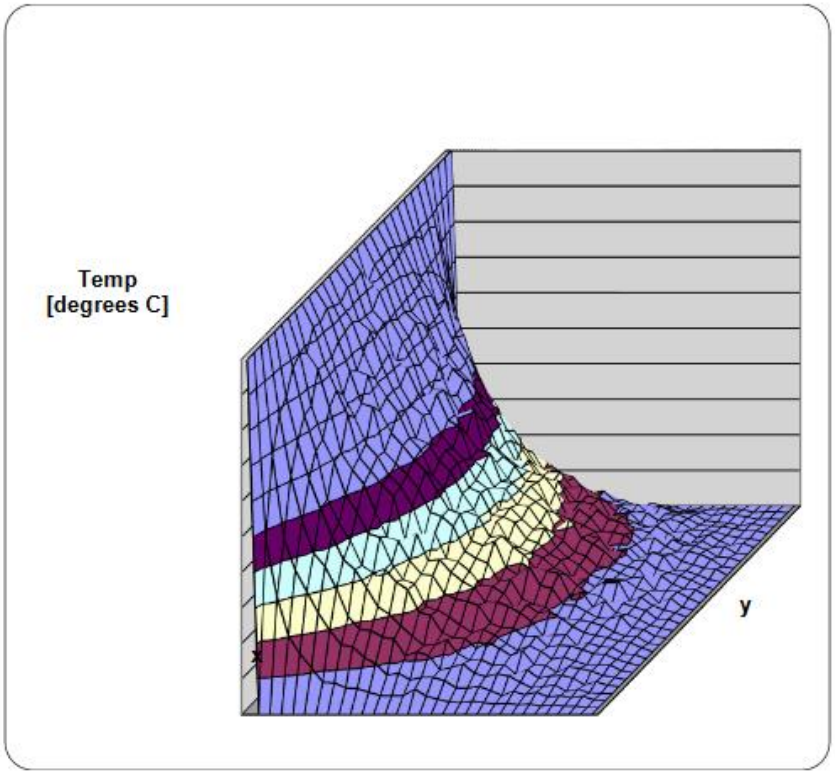
N_k- number of walks that end in point q_k

Monte-Carlo for Laplace problem

- As an example we have a steady-state heat equation

$$-\nabla \cdot (\kappa \, \nabla \, u) = f \quad \text{in } \Omega$$

with $\kappa = 1$, rectangular area and Dirichlet boundary conditions:



The cost of solution for single unknown point can be less comparing to deterministic methods

Chapter 6

POTENTIAL THEORY AND HEAT CONDUCTION DIRICHLET'S PROBLEM

© M. Ragheb
9/19/2013

Other applications

- In engineering Monte-Carlo methods are applied also to estimate design sensitivity
  - To predict/estimate the influence of manufacturing tolerances
  - Evaluate complex model with properly selected distribution of inputs
  - Analyze the statistical properties of outputs

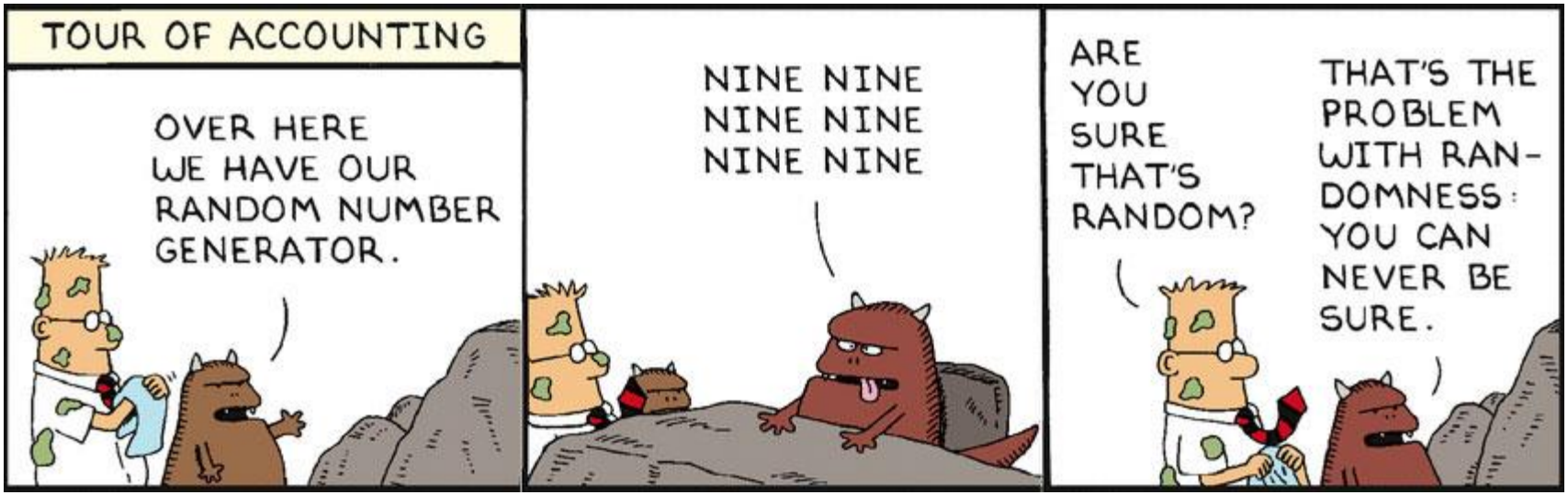
Some practical considerations

- Monte Carlo Methods are naturally parallel
- Many Monte Carlo problems require very large number of random numbers – order of 10^{15} (!)
- The result will be meaningfull only if the random numbers have good quality
- It is not an easy task to generate such a set of random numbers!!!
- Q: How can we get a random numbers?

Modeling and simulation of systems

Lecture 11. Random Number Generators

Random Number Generators



"*Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.*"

John von Neumann

Random number generators

- The only true random source is nature.
- Why not use physics directly? Use can buy a hardware random generator (**true random number generator**, **TRNG)** based on physical process:
  - Photons travelling through semi-transparent windows generate random 0/1
  - Nuclear radiation detection
  - Thermal noise in electronic circuits
  - …
- Very good properties
  - Especially used in cryptographic applications (military)

- There are few of commercial products available today, even as USB-stick (!)
- BUT:
  - Low speed – usually we get a slow data rate – few MB/s
  - Low portability
  - Cost

RDRAND

- Some platforms already have an ***on-board*** **hardware random generators**
- Example: RDRAND
  - It is present in Intel Ivy Bridge processors and AMD64 processors **since 2015**
  - The generator takes pairs of 256-bit raw entropy samples generated by the hardware entropy source and then applies some digital processing (uses this value as a seed)
  - The entropy source: runs asynchronously on a self-timed circuit and uses thermal noise within the silicon to output a random stream of bits at the rate of 3 GHz

RDRAND

- Criticism:
  - „Relying solely on the hardware random number generator which is using an implementation sealed inside a chip which is impossible to audit **is a BAD idea”**
    - Intel was pushing Linux community to use RDRAND as the only RNG in /dev/random
    - In 2013 E. Snowden leaked some details on BULLRUN programme: highly classified program to crack encryption of online communications and data, which is run by the United States National Security Agency (NSA).
    - Similar programme was run in UK
    - Under BULLRUN programme NSA was able to infiltrate VPN & SSL networks
  - It was demonstrated that the Linux random number generator could become insecure if a backdoor is introduced into the RdRand instruction that specifically targets the code using it.
  - Nowadays Linux community is suspicious to use RDRAND
    - FreeBSD removed RDRAND
  - **RDRAND is slower than state-of-the art software-based generators**

Pseudo Random Number Generators

Random number generator (RNG)

- In most cases we want to get a pseudorandom numbers with uniform distribution over unit interval (0,1)
- We get a subset of space R that can be represented in computer and denote this as F
  - Each number is represented as: $\pm(d_1 2^{e-1} + d_2 2^{e-2} + \ \dots d_p 2^{e-p})$ where d is 0 or 1, alternatively:

$$\pm 0.d_1 d_2 \dots d_p \times b^e$$

  - Why not generate random bits in floating point representation?
    - The „density" of real number differs for small and large double numbers!

- In software RNG we usually generate pseudorandom integers and then scale them into interval (0,1)

Congruence of integer numbers

- For a positive integer n, two numbers a and b are said to be congruent modulo *n*, if their difference *a* − *b* is an integer multiple of n:

$$a \equiv b \ \ (mod\ n)$$

- The congruence relation may be rewritten as $a = k \cdot n + b$

(The remainder of division of *a* and *b* with *n* is the same)

Lehmer generator

- In 1948 D. Lehmer proposed a simple *linear congruential generator*:

$$x_i \equiv (ax_{i-1} + c) \ mod\, m, \qquad with\ 0 < x_i < m$$

  - *a* is called „multiplier,"
  - *c* is called „increment,"
  - *m* is called „modulus" of the generator
- In classic Lehmer generator it is assumed that $c = 0$
  - *multiplicative Lehmer generator*
- The seed x_0 is just a single starting value
- A sequence of numbers x_i is called *Lehmer sequence*
- Later on each element x_i is scaled into unit interval:

$$u_i = x_i/m$$

Lehmer generator

- Let's take a look at example sequence where $x_0 = 1, m = 17, a = 3, c = 0$:

  3 9 10 13 5 15 11 16 14 8 7 4 12 2 6

  1 3 9 10 13

- In Lehmer generator x_i is determined by x_{i-1}
  - There are only (*m-1)* unique values of x_i:
  - Therefore the **maximum** theoretical period of this generator is *m-1 !*
  - **The period depends on the smallest positive *k* that** $\boldsymbol{a^k \equiv 1 \; mod \; m}$ (in our example: $3^{16} \equiv 1 \; mod \; 17 \quad \rightarrow \quad k = 16$
  - If we use $m = 2^q$ - fast to compute, but max. period is m/4

Lehmer generator

- Examples:
  - $x_0 = 1, \quad m = 17, a = 4, c = 0$:

    4 16 13 1 4 16 13 …

  - $x_0 = 2, \quad m = 17, a = 4, c = 0$:

    8 15 9 2 8 15 9 …

Lehmer generator

a,b are *relatively prime* if gcd(a,b) = 1

- Theorem: If *a* and *m* are relatively prime, then

$$a^{\phi(m)} \equiv 1 \quad mod\ m$$

where $\phi(m)$ is the **Euler totient** function.

(Euler's totient function counts the positive integers up to a given integer n that are relatively prime to n)

- As result, the period is **upper - limited** with value of $\phi(m)$
- In order to maximize the period, for given *m* we seek *a* that our *k* is equal $\phi(m)$ - we call this number a ***primitive root modulo m***

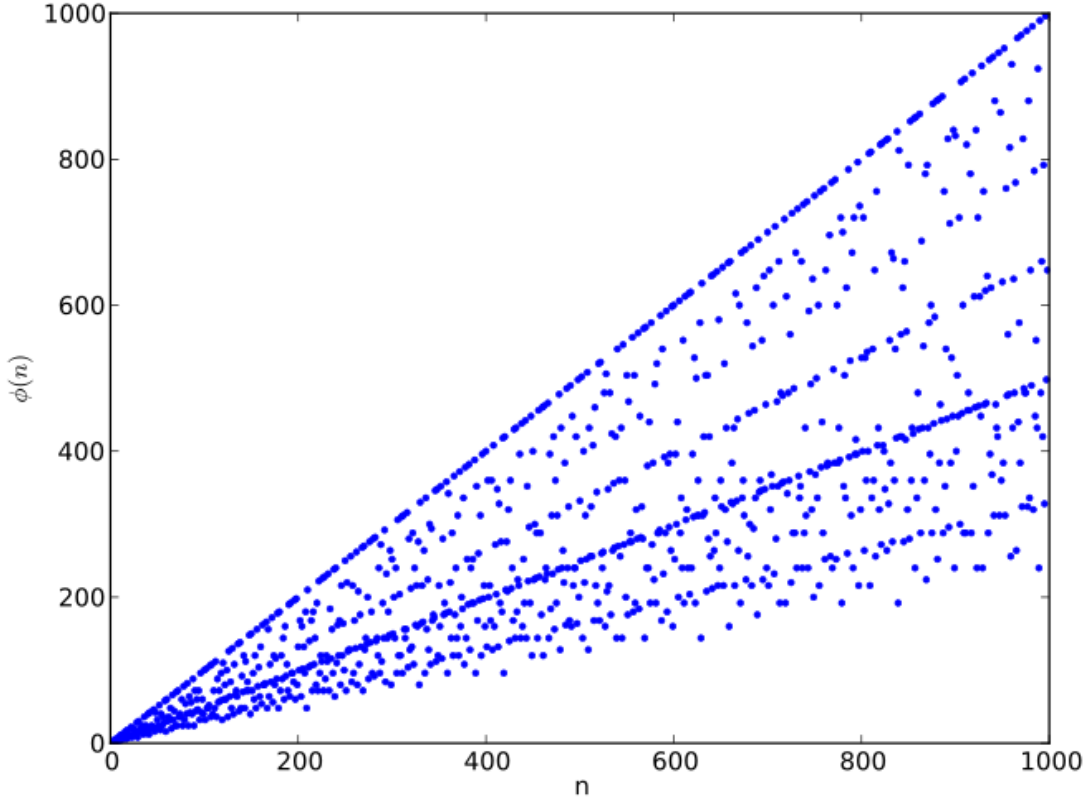
Primitive roots

- For example:
  - assume m = 171
  - Goal: find the best *a,* check if pair (m,a) is good choice…
- Look for relative prime number to *m*
  - Try: a = 3, but gcd(171,3) = 3 !!!
  - Try: a = 5, gcd(171,5) = 1 -> OK

  - $\phi(171) = 108$ – maximum period!
    - test *a* = 5 -> k = 18
    - test *a* = 7 -> k = 4 (!)
    - …
  - BUT: If we change m = 173, then $\phi(m) = 172$

Primitive roots

- First values of $\phi(m)$:



- If m is prime, then $\phi(m) = m - 1$

Practical considerations

- **Need high precision to perform the multiplication!**
  - e.g. use GMP library
- **If** multiplyer is large we can avoid high precision and transform:

$$x_i = ax_{i-1} \ mod \quad m$$

into:

$$x_i = (c(dx_{i-1} \ mod \ \ m) \ + \ e\, x_{i-1} \ mod \ \ m\,)\, mod\, m$$

where $a = cd + e$.

- **Or** we can use multipliers < $\sqrt{m}$ and let $q = \left\lfloor \frac{m}{a} \right\rfloor = floor(\frac{m}{a})$ and $r = m \ mod \ a$. Then:

$$\begin{aligned} ax_{i-1} \bmod m &= (ax_{i-1} - \lfloor x_{i-1}/q \rfloor m) \bmod m \\ &= (ax_{i-1} - \lfloor x_{i-1}/q \rfloor (aq + r)) \bmod m \\ &= (a(x_{i-1} - \lfloor x_{i-1}/q \rfloor)q - \lfloor x_{i-1}/q \rfloor r) \bmod m \\ &= (a(x_{i-1} \bmod q) - \lfloor x_{i-1}/q \rfloor r) \bmod m. \end{aligned}$$

Implementation:

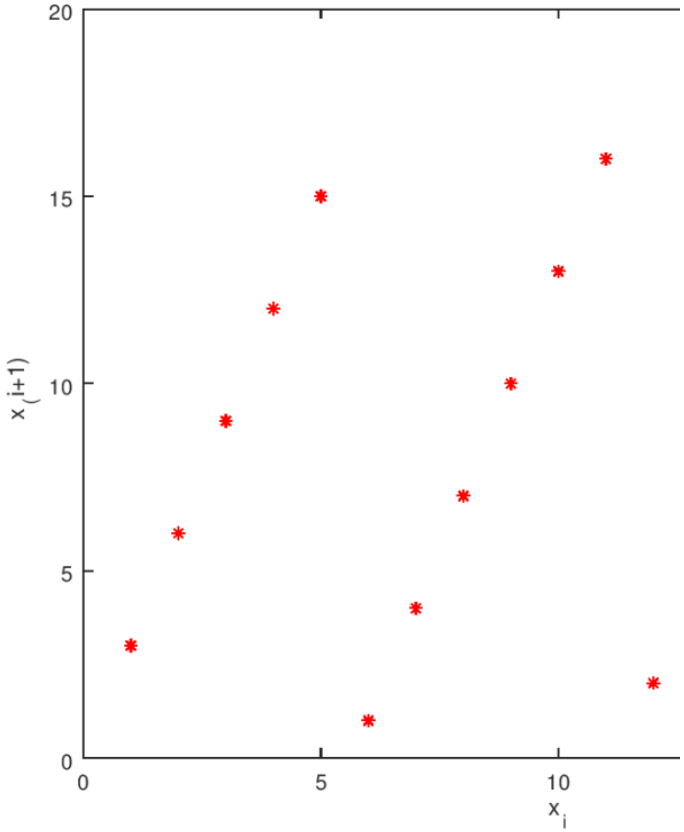
```
k = x/q
x = a*(x - k*q) - k*r
if (x < 0) x = x + m
```

Structure of outputs

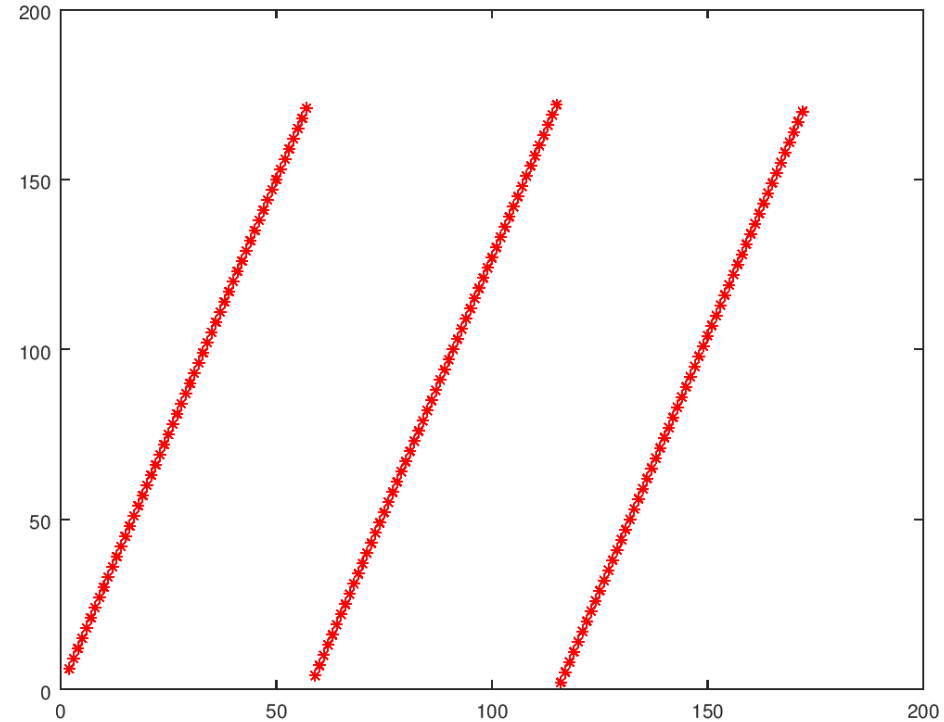
- So far we know that Lehmer generator is periodic
  - If we take a full cycle (*m-1*) then we get a uniform distribution
  - BUT: In a good generator any subsample (subset) of the cycle also should be uniformly distributed
    - The serial corelation should be small(!)
  - In fact: the structure of a sequence from Lehmer generator is very regular

Structure of outputs

- 2D lattice - plot of (x_{i-1}, x_i) on plane



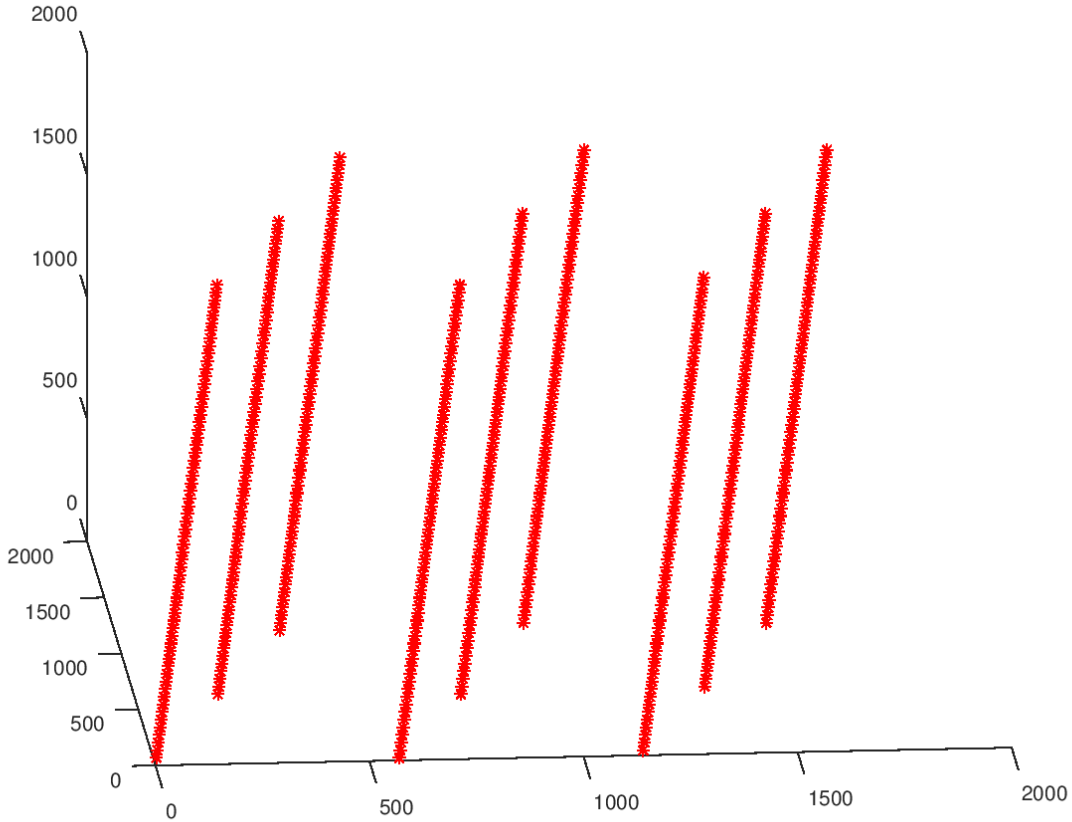
$x_0 = 1, m = 17, a = 3, c = 0$



$x_0 = 1, m = 173, a = 3, c = 0$

Structure of outputs

- 3D Lattice – plot of (x_i, x_{i-1}, x_{i-2})



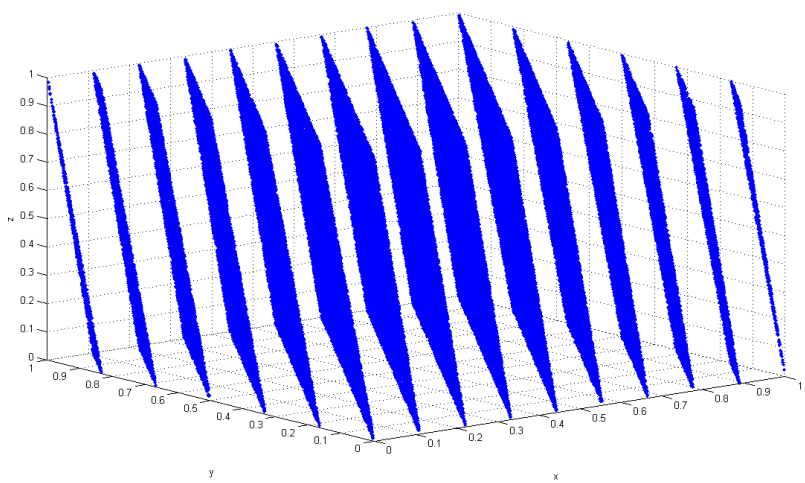
$x_0 = 1, m = 1707, a = 3, c = 0$

RANDU

- For many years since 1960's RANDU generator form IBM was the most popular random number algorithm:

$$x_i = 65539\, x_{i-1} \mod 2^{31}$$

- It has a nice 2D lattice structure, but very poor 3D lattice!
  - Now it is considered as „truly horrible"



Only 15 planes in 3D!

Implementations: rand()

Glibc:

```
#define RAND_MAX 2147483647
```

Visual Studio stdlib.h

```
#define RAND_MAX 0x7fff
```

So, how about code protability?

```c
int
__random_r (buf, result)
     struct random_data *buf;
     int32_t *result;
{
  int32_t *state;

  if (buf == NULL || result == NULL)
    goto fail;

  state = buf->state;

  if (buf->rand_type == TYPE_0)
    {
      int32_t val = state[0];
      val = ((state[0] * 1103515245) + 12345) & 0x7fffffff;
      state[0] = val;
      *result = val;
    }
  else
    {
      int32_t *fptr = buf->fptr;
      int32_t *rptr = buf->rptr;
      int32_t *end_ptr = buf->end_ptr;
      int32_t val;

      val = *fptr += *rptr;
      /* Chucking least random bit.  */
      *result = (val >> 1) & 0x7fffffff;
      ++fptr;
      if (fptr >= end_ptr)
        {
          fptr = state;
          ++rptr;
        }
      else
        {
          ++rptr;
          if (rptr >= end_ptr)
            rptr = state;
        }
      buf->fptr = fptr;
      buf->rptr = rptr;
    }
  return 0;

 fail:
  __set_errno (EINVAL);
  return -1;
}
```

rand() in glibc

```c
/* Linear congruential.  */
#define TYPE_0          0
#define BREAK_0         8
#define DEG_0           0
#define SEP_0           0

/* x**7 + x**3 + 1.  */
#define TYPE_1          1
#define BREAK_1         32
#define DEG_1           7
#define SEP_1           3

/* x**15 + x + 1.  */
#define TYPE_2          2
#define BREAK_2         64
#define DEG_2           15
#define SEP_2           1

/* x**31 + x**3 + 1.  */
#define TYPE_3          3
#define BREAK_3         128
#define DEG_3           31
#define SEP_3           3

/* x**63 + x + 1.  */
#define TYPE_4          4
#define BREAK_4         256
#define DEG_4           63
#define SEP_4           1
```

Nonlinear Congruential Generators

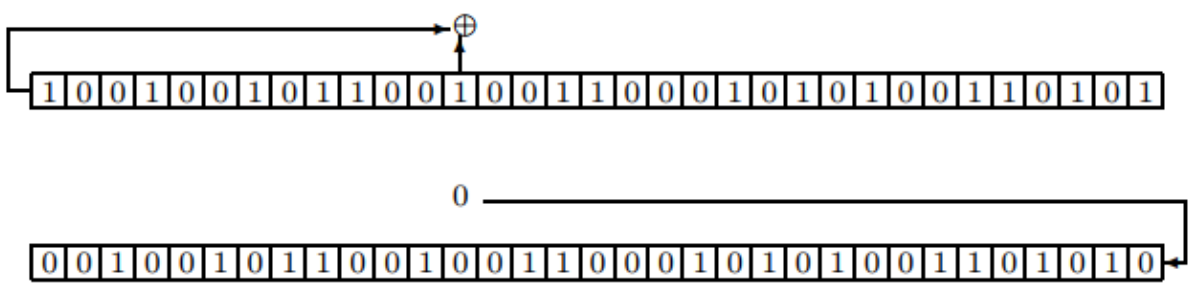
- Polynomial: Knuth (1998):

$$x_i = \left(dx_{i-1}^2 + ax_{i-1} + c\right) \ mod \ \ m$$

  - Higher order polynomials possible
- Feedback Shift Register:
  - Based on sequence of 0s and 1s:
  - Then we use the output bits to form integers

$$b_i \equiv (a_p b_{i-p} + a_{p-1} b_{i-p+1} + \cdots + a_1 b_{i-1}) \bmod 2,$$

  - This recurrence operation can be performed in a *feedback shift register* – a vector of bits that is shifted and combined



Twisting the bits

- Additionally in feedback shift register we can apply the bit twisting
  - This operation can be performed introducing permutation matrix P, size of (L x L), where L is a length of the register
  - There is extensive theory about this type of generators

- One of the state-of-the-art examples is Marsenne Twister generator

Mersenne Twister

- The most widely used general purpose PRNG:
  - Developed in 1997 by Makoto Matsumoto and Takuji Nishimura
  - Based on Mersenne prime $2^{19937} - 1$ (long period)
  - Fast (but complex implementation)
  - Good random properties
  - Patent-free (!)
  - 623-variate uniformity (32bit version MT19937)
    - After observing 624 values we can predict the next ones
    - Poor for cryptographic proposes!

Mersenne Twister

- The Mersenne Twister is the default PRNG for the following software systems (Wikipedia):

- *Microsoft Excel, GAUSS, GLib, GNU Multiple Precision Arithmetic Library, GNU Octave, GNU Scientific Library, gretl, IDL, Julia, CMU Common Lisp, Embeddable Common Lisp, Steel Bank Common Lisp, Maple, MATLAB, Free Pascal, PHP, Python, R, Ruby, SageMath, Scilab, Stata.*

- *It is also available in Apache Commons, in standard C++ (since C++11), and in Mathematica. Add-on implementations are provided in many program libraries, including the Boost C++ Libraries, the CUDA Library, and the NAG Numerical Library.*

Parallelism in PRNGs

- Many of the Monte Carlo methods are parallel
- We need to have independent substreams of random numbers
  - The same generator with:
    - Different seed (inital values)
    - Different multiplyer (same modulus)
  - Different types of generators

Random number generators

- Often we need to generate the numbers with given distribution – how can we do that?
  - Poisson?
  - Normal?
  - …
- Apply **Inverse Transformation Method:**
- *If Y is a continous uniform random variable and X has cumulative distribution function* F_X*, then random variable* $F_X^{-1}(Y)$ *has the same distribution as X*
  - Assumption: cumulative distribution function is invertible!

Modelowanie i Symulacja Systemów

Introduction to Finite Element Methods

Adam Lamecki

Series of lectures on FEM

- Introduction
- Strong vs. weak form of BVP
- Steady-state heat transfer in 1D
- Transient heat transfer in 1D
- Heat transfer in 2D & 3D
- Solution of solid mechanics problems with 3D FEM

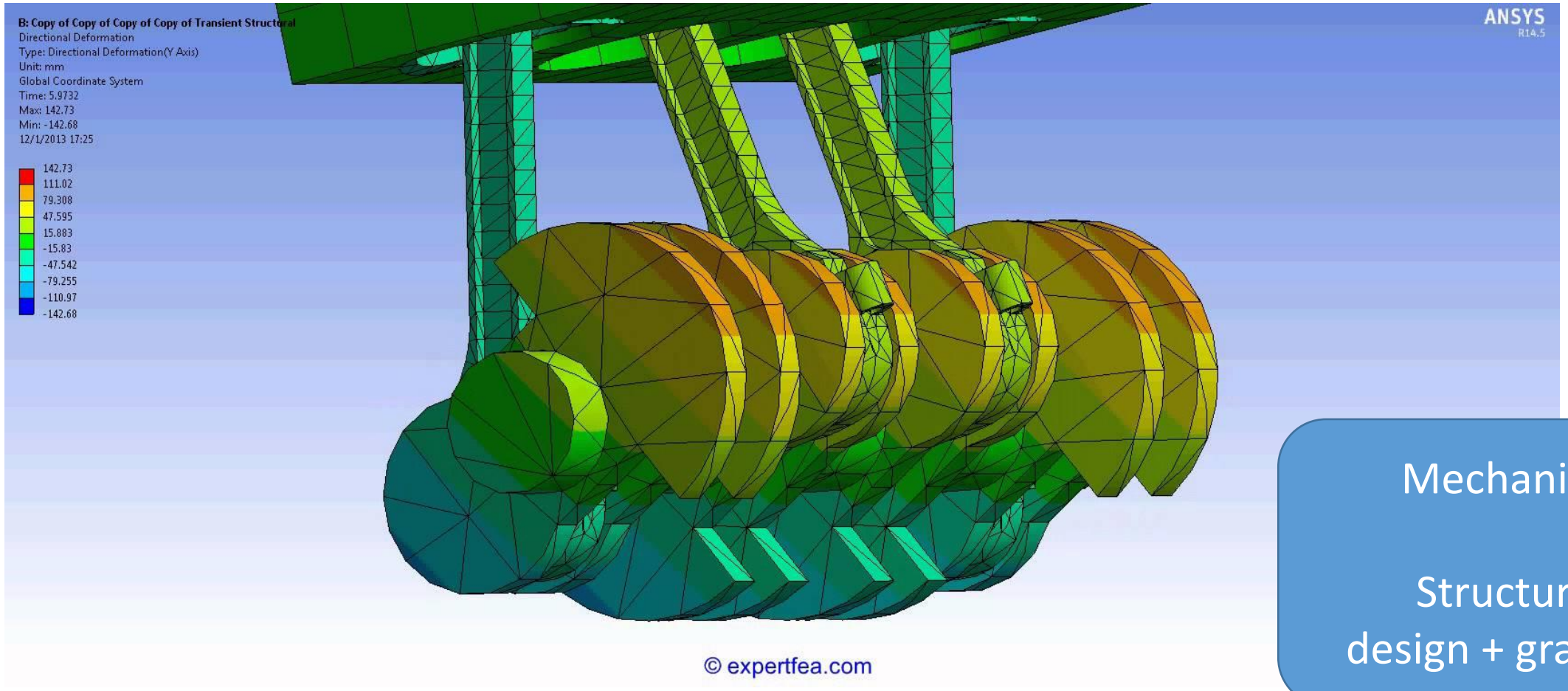
Introduction to FEM

- Finite element method (FEM) has become a key modelling technology that is often applied in design of advanced engineering systems:
  - Heat transfer
  - Fluid flow
  - Computational mechanics (&computer graphics)
  - Electrodynamics
  - Acoustics
  - Coupled simulations (multiphysics)
  - ….

- The great advantage of FEM is **flexibility**

- The disadvantage is complexity – quite difficult math…

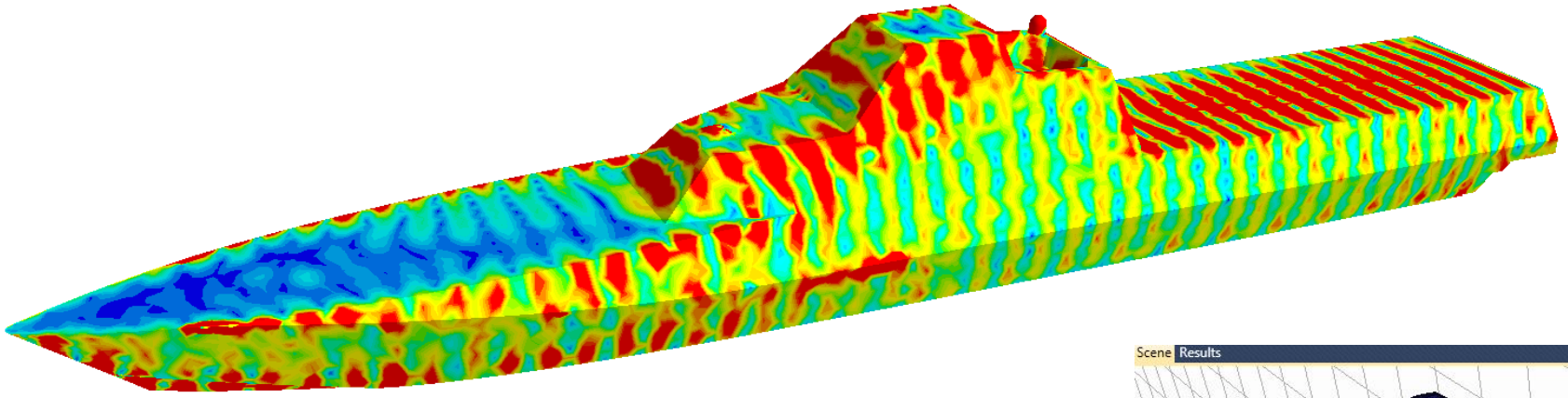
Examples of FEM simulations (1)



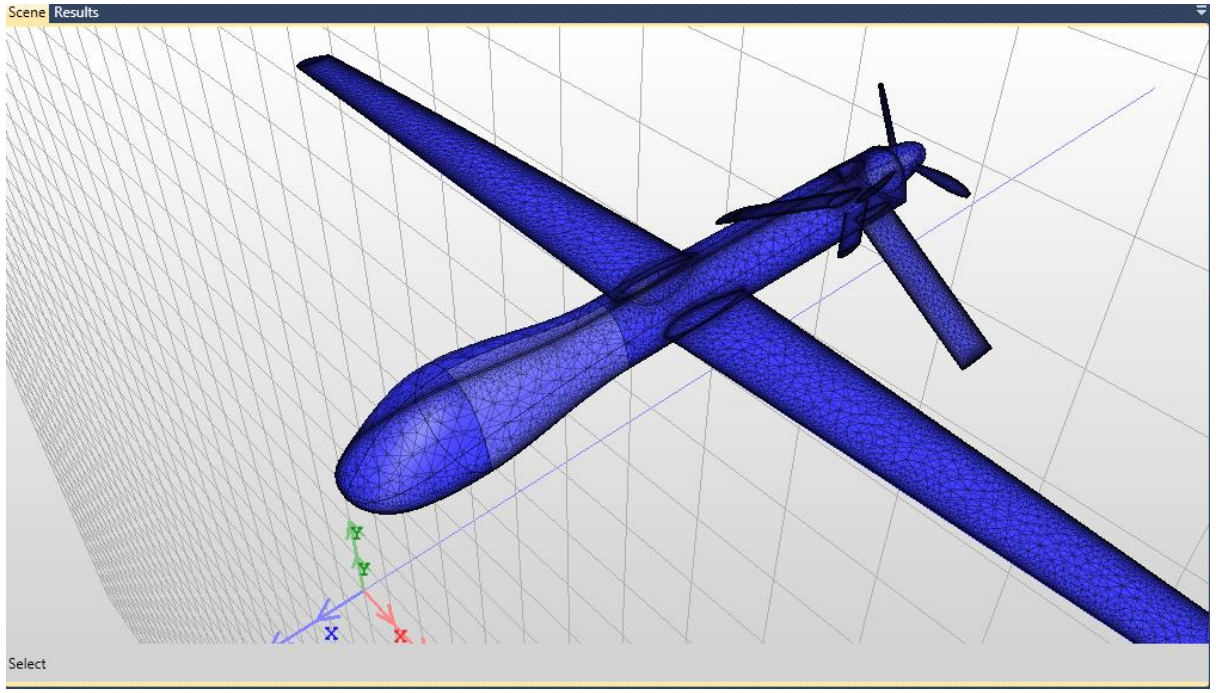
Mechanics:

Structural design + graphics

Examples of FEM simulations (2)

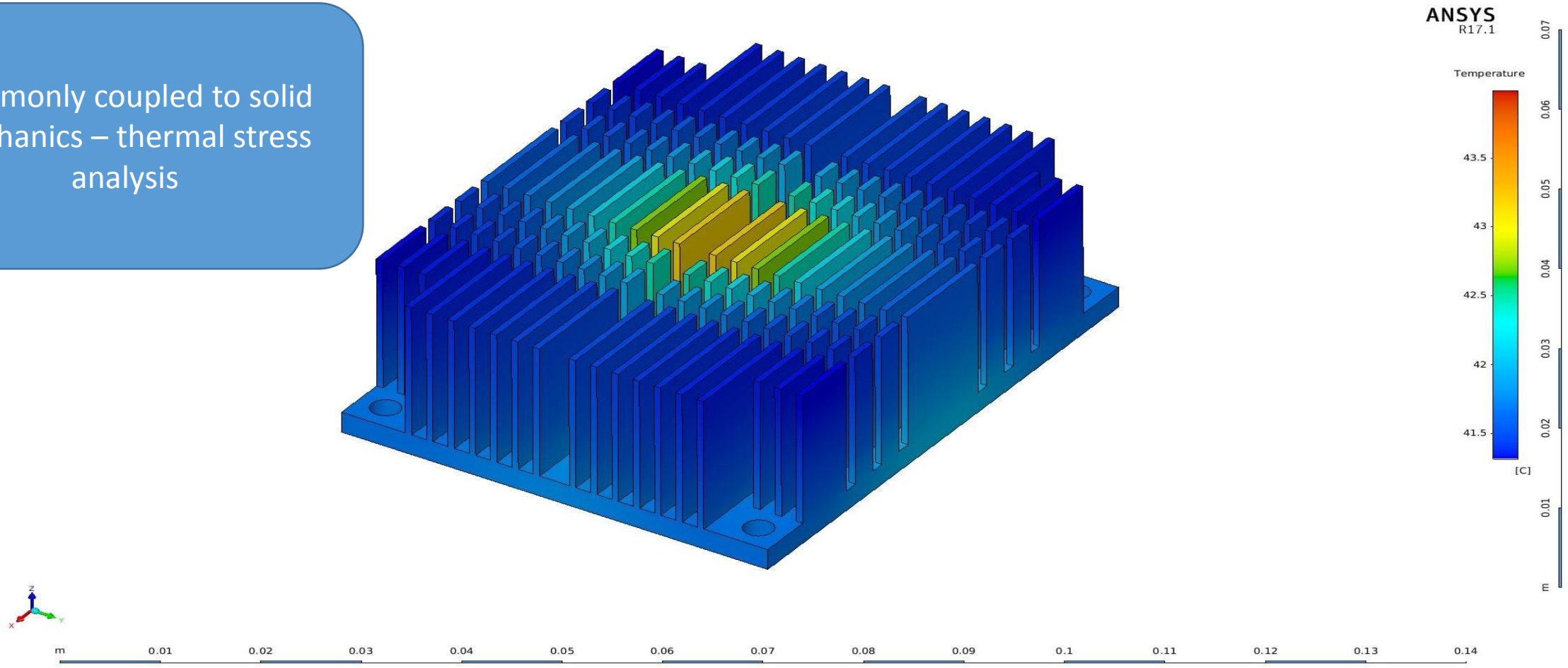


Electromagnetic field scattering

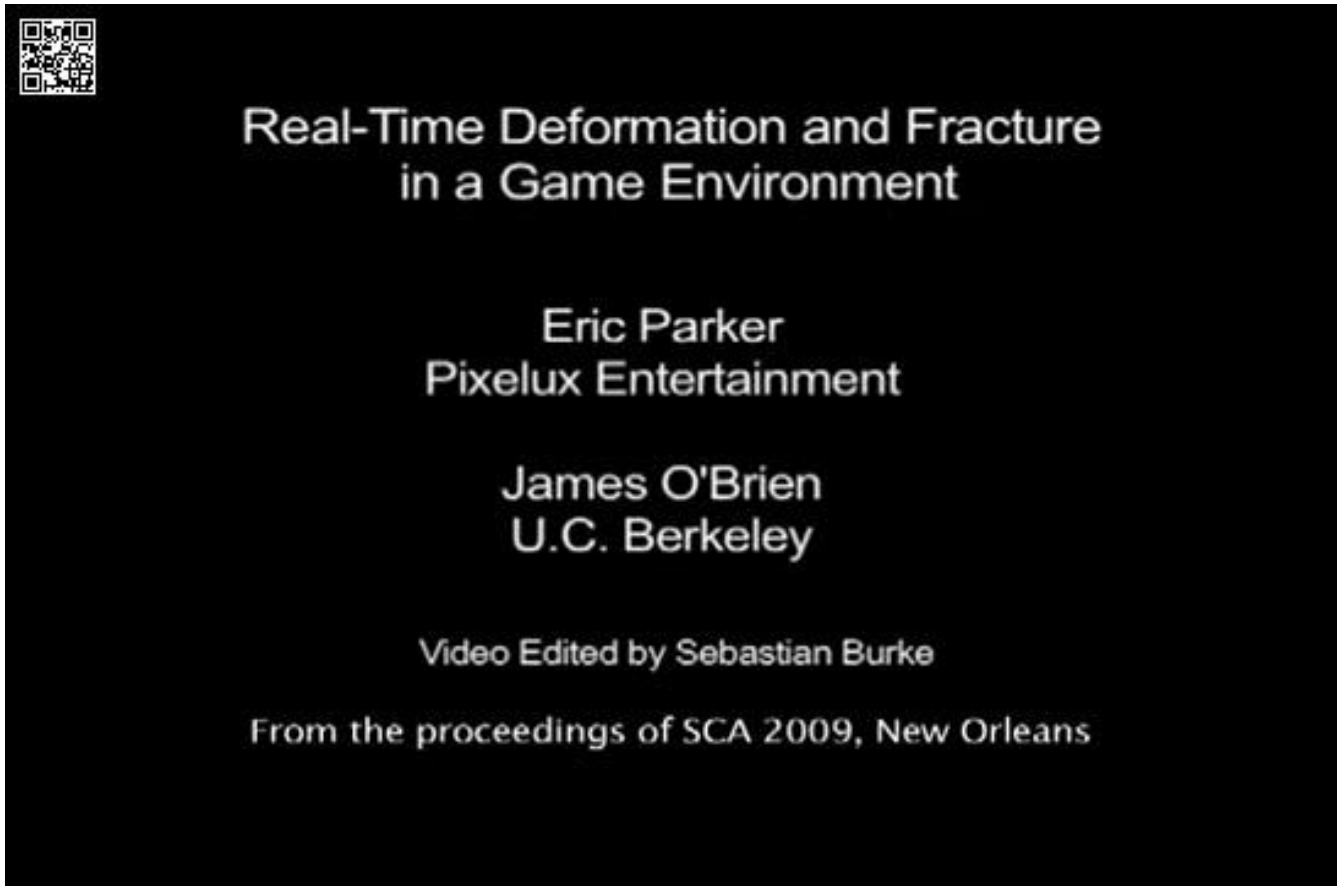


Heat transfer

Commonly coupled to solid mechanics – thermal stress analysis

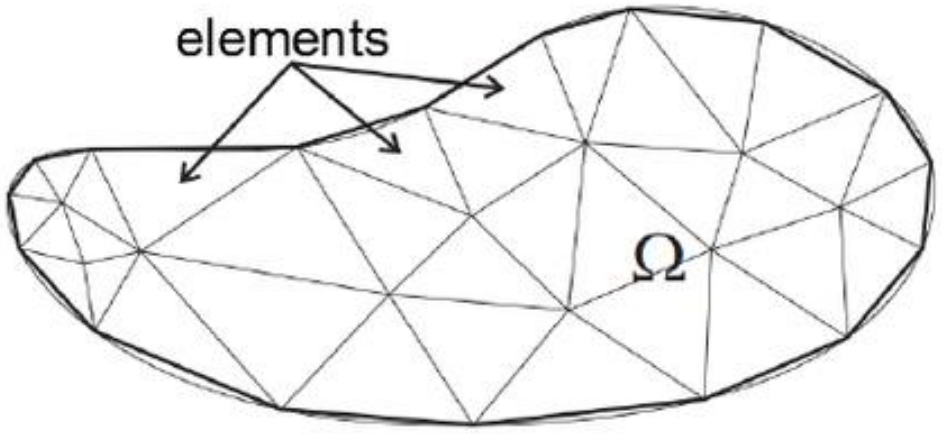


FEM in Entertainment

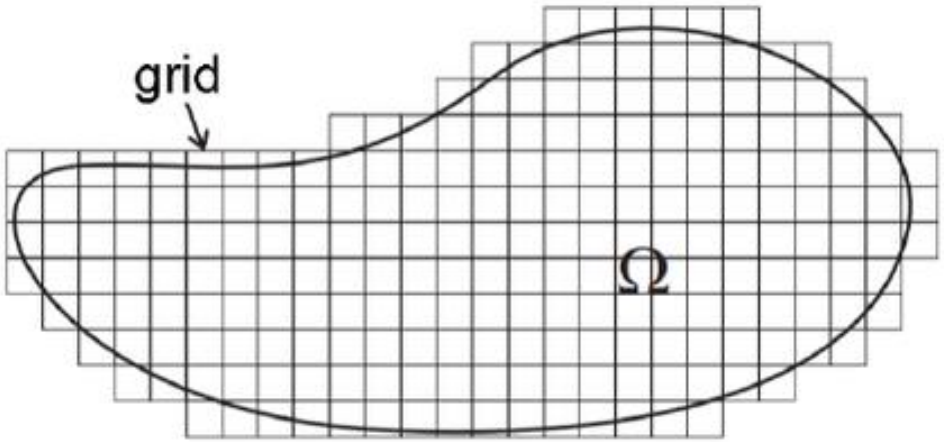


Eric G. Parker, James F. O'Brien, "Real-time deformation and fracture in a game environment," Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, p. 165-175

FEM vs. FD



FEM mesh



FD mesh

Ferreira, Leonardo & Fonseca, A.R. & Zoia Lima, Naisses & Mesquita, Renato. (2015). Graphical Interface for Electromagnetic Problem Solving Using Meshless Methods. Journal of Microwaves, Optoelectronics and Electromagnetic Applications. 14. SI-54.

FEM

- But how it is actually done?
- In finite differences we used the samples of our solution in discrete points on the grid (1D,2D or 3D).
- In finite elements we look for the solution in the form of a continuous function!

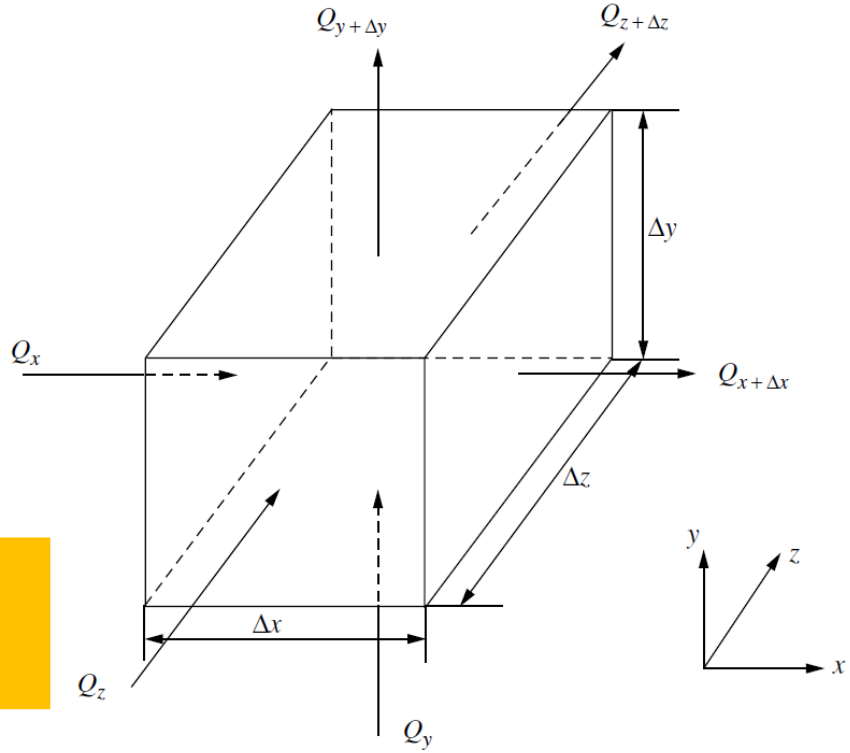
Heat transfer

- Basic govering equations:
  - Temperature $u = u(x, y, z, t)$ - scalar field
  - Heat flux q – vector field
  - Heat flux q through surface A (1D): $Q = -\kappa_x A \cdot \frac{du}{dx}$ (Fourier law of heat conduction)

    where κ – heat conductivity (W/mK)
  - Change of energy stored in volume: $\Delta Q = c_p \rho \frac{\delta u}{\delta t}$
- Use energy conservation condition to derive the heat transfer equation
  - Energy in – Energy out + Energy generation (f) = Energy stored
- We get general heat transfer equation (3D):

$$c_p \rho \frac{\partial u}{\partial t} = \nabla \cdot (\kappa \nabla u) + f$$

where c_p - heat capacity, ρ – mass density



A. Lamecki – Introduction to Finite Element Methods, 2017

Strong form of BVP

- First let's consider the steady-state heat transfer problem:

$$-\nabla \cdot (\kappa \, \nabla \, u) = f \quad \text{in } \Omega$$

where κ is thermal conductivity, u is temperature and f is a heat source (or sink)

- If f is continous and u is a solution, then u and its partial derivatives up to second order are continous on Ω

- Let's define a set of such functions as C^2

Weak form of BVP

- If u is a solution then $-\nabla \cdot (\kappa\, \nabla\, u) = f \quad \text{in } \Omega$
- We multiply both sides with **test function $\boldsymbol{v}$**:

$$-\nabla \cdot (\kappa\, \nabla\, u) v = f v \quad \text{in } \Omega$$

- Also the integral over Ω must agree:

$$\int_\Omega \nabla \cdot (\kappa \nabla u) v \, d\Omega = -\int_\Omega f v \, d\Omega$$

- Apply the Green's identity:

Fundamental step in FEM !!!

$$\boxed{\int_\Omega p(\nabla \cdot \boldsymbol{F})\, d\Omega = -\int_\Omega \mathbf{F} \cdot \nabla p \, d\Omega + \int_{\delta\Omega} p(\boldsymbol{F} \cdot \vec{n}) d\Omega}$$

$$\int_\Omega v\left(\nabla \cdot (\kappa \nabla u)\right) d\Omega = -\int_\Omega (\kappa\, \nabla u) \nabla v \, d\Omega + \int_{\delta\Omega} v(\kappa \nabla u \cdot \vec{n}) d\Omega$$

Weak form of BVP

- With this we obtain a new form of BVP:

Find function $u \in C^2(\Omega)$ such that:

$$\int_\Omega \nabla u \, \kappa \, \nabla v \, \mathrm{d}\Omega - \int_{\delta\Omega} \kappa v (\nabla u \cdot \vec{n}) d\Omega = \int_\Omega f \, v \, \mathrm{d}\Omega$$

for **<u>all</u>** $\mathrm{v} \in \mathrm{C}^2(\Omega)$

- In the original PDE $u \in C^2(\Omega)$
- In the weak form we <u>have only first derivatives</u> of u
- In the weak form $(f \cdot v)$ must be integrable
- Weak form: weakest possible assumptions on the functions involved (relaxed PDE)

Weak form of BVP

- In the weak form of BVP it is not necessary that all functions and derivatives be continuous
- Instead we introduce a set of *weakly differentiable* functions

Weakly differentiable functions

- The intuitive interpretation of weakly-differentiable function is that the function is differentiable except for set of discrete points

  $\rightarrow$ It can the integrated
- Weak derivative of continuous functions is the same as regular (strong) derivative
- Weak derivative use the same notation as strong one $\left(u' = \frac{du}{dx}\right)$
- Weak derivative follows the same rules as „strong” one

  $\rightarrow$ e.g. integration by parts

Weak derivative

- Example: Let Ω be the line segment $(0,1)$ and define:

$$u(x) = \begin{cases} x, & 0 < x \leq \frac{1}{2} \\ 1 - x, & \frac{1}{2} < x < 1 \end{cases}$$

Function $u(x)$ is not differentiable over all of Ω, but has a partial derivative

$$\frac{\partial u}{\partial x} = g(x) = \begin{cases} 1 & 0 < x \leq \frac{1}{2} \\ -1 & \frac{1}{2} < x < 1 \end{cases}$$

- In mathematics a set of weakly derivative functions up to order *n* forms a **<u>Sobolev space H^n</u>**

Weak form of BVP

- Finally, our BVP can be defined in terms of the <u>Sobolev space $H^1(\Omega)$:</u>

Find $u \in H^1(\Omega)$, $\int_\Omega \nabla u \cdot \kappa \cdot \nabla v \, \mathrm{d}\Omega - \int_{\delta\Omega} \kappa v (\nabla u \cdot \vec{n}) d\Omega = \int_\Omega f v \, d\Omega$

for all $v \in H^1(\Omega)$

This is a *weak-form* of our original problem.

- One needs to derive the weak form FIRST!
- In most cases this is already done: mechanics, CFD, etc.

Galerkin method

- How to solve this weak-form equation?
- In general u, v belong to infinite dimensional space
- In order to solve the problem on computer we restrict ourselves to its finite-dimensional approximation W (subspace of H^1) size of *n*
- We want to find a best approximation of our solution using the subspace functions $w_i \ \ (i = 1 \dots n)$

$$u = \sum_i^n u_i \cdot w_i$$

(our solution is a linear combination of basis functions – we only need to find the unknown coefficients u_i)

Galerkin method

- We can trasform infinite-dimensional equation into a finite-dimensional set of equations:

$$\int_\Omega \nabla u \cdot \kappa \cdot \nabla v \, \mathrm{d}\Omega = \int_\Omega f v \, d\Omega$$

$$\int_\Omega \nabla \left(\sum_i^n u_i \cdot w_i \right) \cdot \kappa \cdot \nabla w_1 \, \mathrm{d}\Omega = \int_\Omega f w_1 \, d\Omega$$

$$\int_\Omega \nabla \left(\sum_i^n u_i \cdot w_i \right) \cdot \kappa \cdot \nabla w_2 \, \mathrm{d}\Omega = \int_\Omega f w_2 \, d\Omega$$

$$\ldots$$

$$\int_\Omega \nabla \left(\sum_i^n u_i \cdot w_i \right) \cdot \kappa \cdot \nabla w_n \, \mathrm{d}\Omega = \int_\Omega f w_n \, d\Omega$$

→ For simplicity we skipped term :

$$-\int_{\delta\Omega} \kappa v (\nabla u \cdot \vec{n}) d\Omega$$

→ **<u>we use the same set of base functions for testing and solution approximation</u>**

$$v = \{w_1, w_2, \ldots, w_n\}$$

(*here we skip the analysis of uniqueness of solution*)

Galerkin method

- If we take just one equation:

$$\int_{\Omega} \nabla\left(\sum_{i}^{n} u_i \cdot w_i\right) \cdot \kappa \cdot \nabla w_1 \, \mathrm{d}\Omega = \int_{\Omega} f w_1 \, d\Omega$$

$$\Downarrow$$

$$\sum_{i}^{n} u_i \cdot \int_{\Omega} \nabla w_i \cdot \kappa \cdot \nabla w_1 \, \mathrm{d}\Omega = \int_{\Omega} f w_1 \, d\Omega$$

- Then, if one defines a matrix K as: $k_{ij} = \int_{\Omega} \nabla w_i \cdot \kappa \cdot \nabla w_j d\Omega$, vector $U = [u_1 \quad u_2 \quad \dots \quad u_n]^T$, and vector $l_i = \int_{\Omega} f w_i d\Omega$ we get

$$K \cdot U = L$$

As a result we get a system of *n* equations with *n* unknowns!

Finite element method - summary

Steps:

1) Find a weak form of the problem (hardest!)
2) Define a finite-dimensional set of weakly differentiable base functions
3) Apply Galerkin method
4) Solve arising discrete problem

How to do this in practice?

- Our general problem (3D):

$$c_p \rho \frac{\partial u}{\partial t} = \nabla \cdot (\kappa \, \nabla \, u) + f$$

Assume $\frac{\partial u}{\partial t} = 0$
Steady state (equlibrium)
heat transfer problem

- Let's start with 1D problem:

$$\frac{d}{dx} A\kappa \frac{du}{dx} = f$$

- **Example:** bar segment with L = 3m and cross-section *A*=1, $\kappa = 15 \frac{W}{m \cdot K}, f = 1, u = 0 \text{ for x} = 0 \text{ and x} = 3$:

$\Omega = (0,3) \rightarrow$

1D Heat with FEM

- **First: Drive the *weak form* of original BVP!**
- Multiply with arbitrary function $w(x)$ and integrate over domain:

$$\int_a^b w \frac{d}{dx} A\kappa \frac{du}{dx}\, dx = \int_a^b wf\, dx$$

Notice that:

$$\frac{d}{dx} wf = w\frac{df}{dx} + f\frac{dw}{dx} \quad \rightarrow \quad w\frac{df}{dx} = \frac{d}{dx}(wf) - f\frac{dw}{dx}$$

$$\int_a^b w\frac{df}{dx}dx = \int_a^b \frac{d}{dx}(wf)dx - \int_a^b f\frac{dw}{dx}dx$$

$$= [wf]\Big|_a^b$$

$$\int_a^b w \frac{d}{dx} A\kappa \frac{du}{dx}\, dx = \left[wA\kappa \frac{du}{dx}\right]\Big|_a^b - \int_a^b \frac{dw}{dx} A\kappa \frac{du}{dx} dx = \int_a^b wf\, dx$$

Find $u \in H^1(\Omega)$, that $\int_a^b \frac{dw}{dx} A\kappa \frac{du}{dx} dx = \left[wA\kappa \frac{du}{dx}\right]\Big|_a^b - \int_a^b wf\, dx$ for all $w \in H^1(\Omega)$

1D Heat with FEM

Weak From: Find u ∈ $H_0^1(\Omega)$, that $\int_a^b \frac{dw}{dx} A\kappa \frac{du}{dx} dx = \left[wA\kappa \frac{du}{dx} \right]\Big|_a^b - \int_a^b wf\,dx$

for all w ∈ $H^1(\Omega)$

$$\int_a^b \frac{dw}{dx} A\kappa \frac{du}{dx} dx = \left[wA\kappa \frac{du}{dx} \right]\Big|_a^b - \int_a^b wf\,dx$$

Neumann Boundary conditions Load/Excitation

$$\left[wA\kappa \frac{du}{dx} \right]\Big|_a^b = wA\kappa \frac{du(b)}{dx} - wA\kappa \frac{du(a)}{dx} = wAq_b - wAq_a$$

$$\int_a^b \frac{dw}{dx} A\kappa \frac{du}{dx} dx = wAq_b - wAq_a - \int_a^b wf\,dx$$

q_a, q_b - heat flux at the ends of the bar

1D Heat with FEM

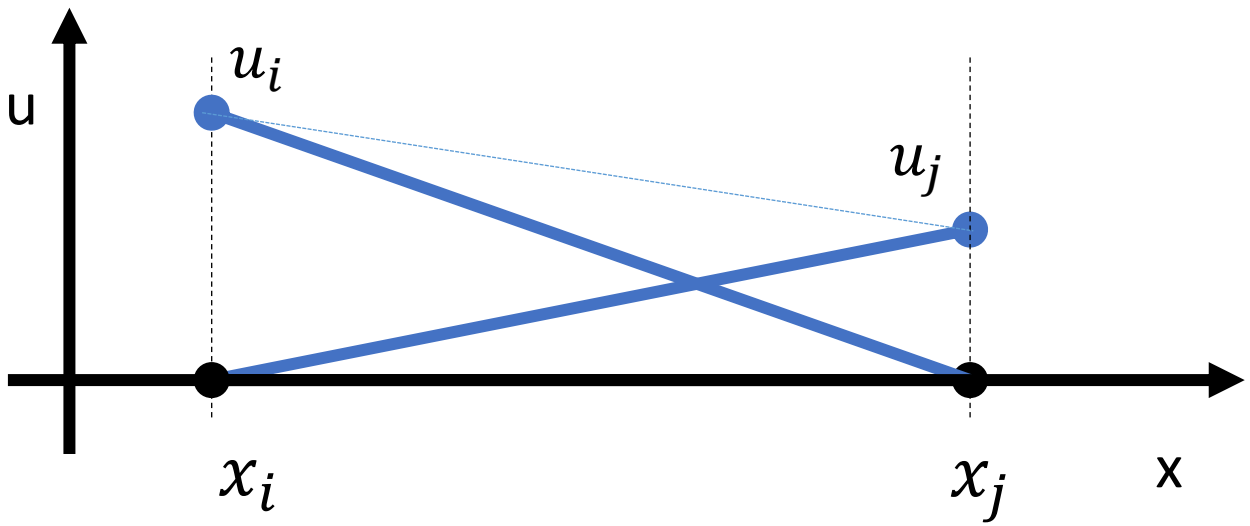
- Now it is time to **define the basis functions $\boldsymbol{w}$**:
- Imagine simple linear function defined on segment x_i to x_j :

$$u(x) = ax + b, \text{ with values at ends equal } u_i, u_j$$

- It can be written as a sum of two linear functions:

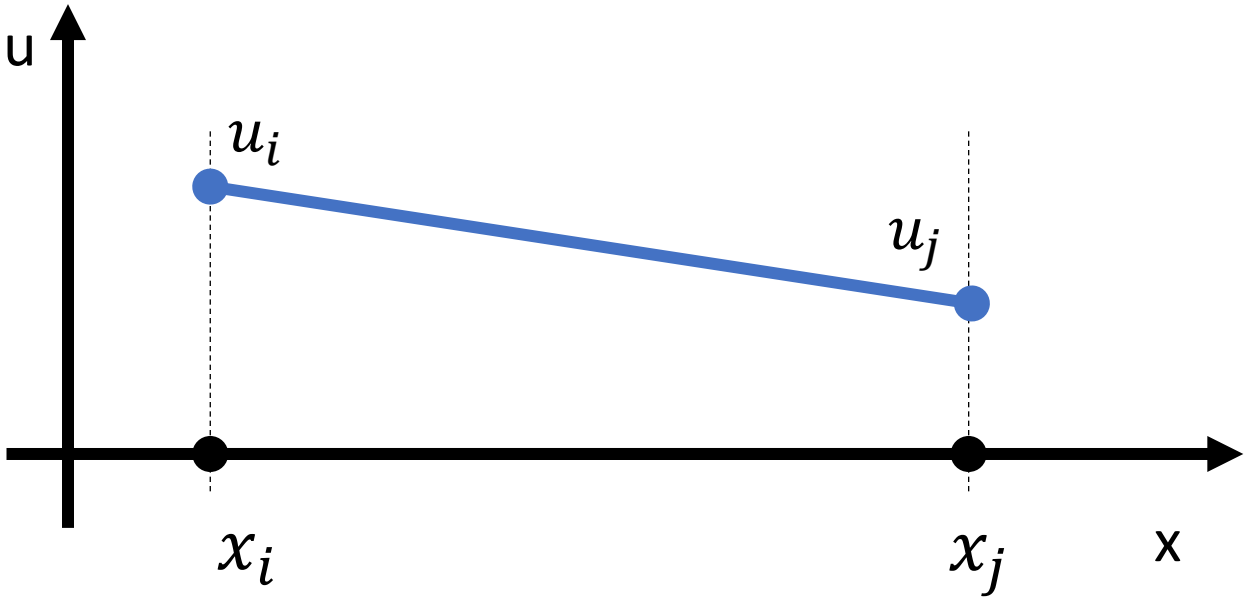
$$u(x) = u_i \left[\frac{x_j - x}{x_j - x_i}\right] + u_j \left[\frac{x - x_i}{x_j - x_i}\right]$$

$$u(x) = u_i \boldsymbol{w_i}(x) + u_j \boldsymbol{w_j}(x)$$



1D Heat with FEM

- Divide the domain into segments, in each segment we have 2 functions: w_i and w_j



1D Heat with FEM

- How about derivative of $u(x)$?

$$\frac{du}{dx} = u_i \frac{dw_i}{dx} + u_j \frac{dw_j}{dx}$$

$$\frac{du}{dx} = -u_i \frac{1}{x_j - x_i} + u_j \frac{1}{x_j - x_i} = const.$$

- We can introduce matrix notation:

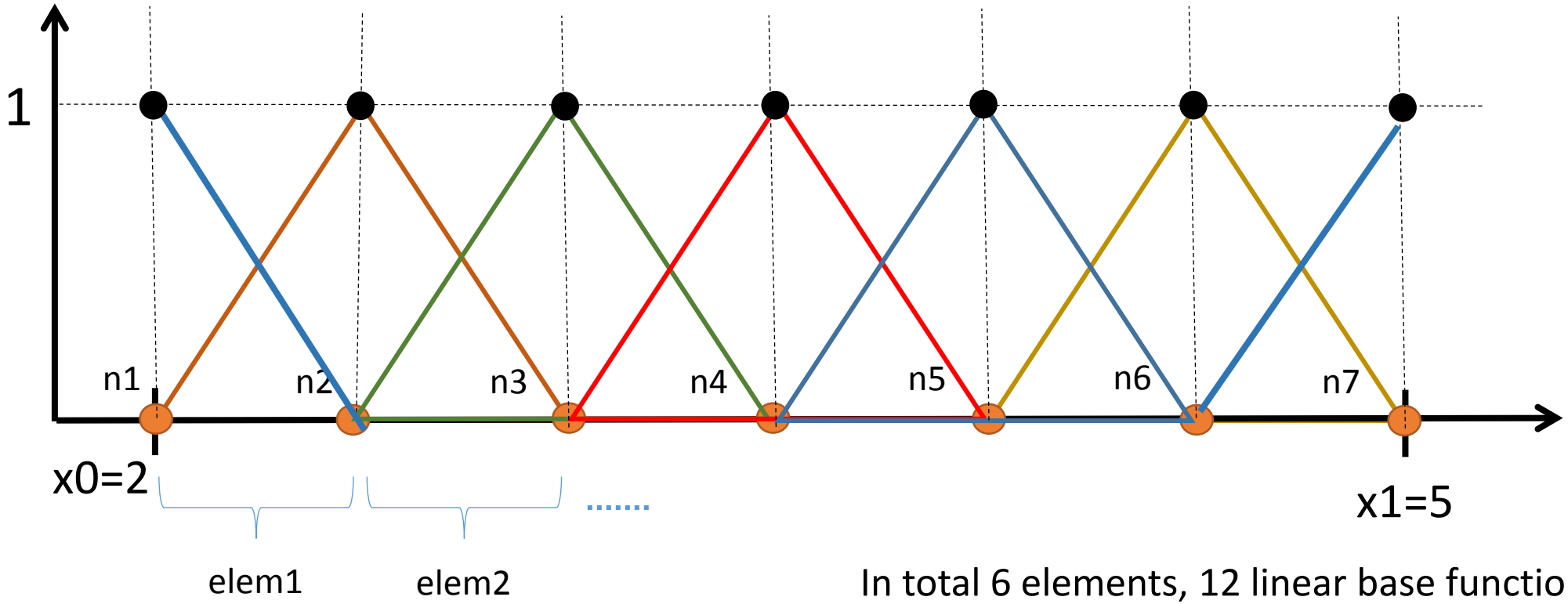
$$u(x) = [N][U] = [w_i \quad w_j] \begin{bmatrix} u_i \\ u_j \end{bmatrix}$$

$$\frac{du}{dx} = [B][U] = \begin{bmatrix} -\frac{1}{L_e} & \frac{1}{L_e} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix},$$

$$L_e = (x_j - x_i) \; - element\ length$$

1D Heat with FEM

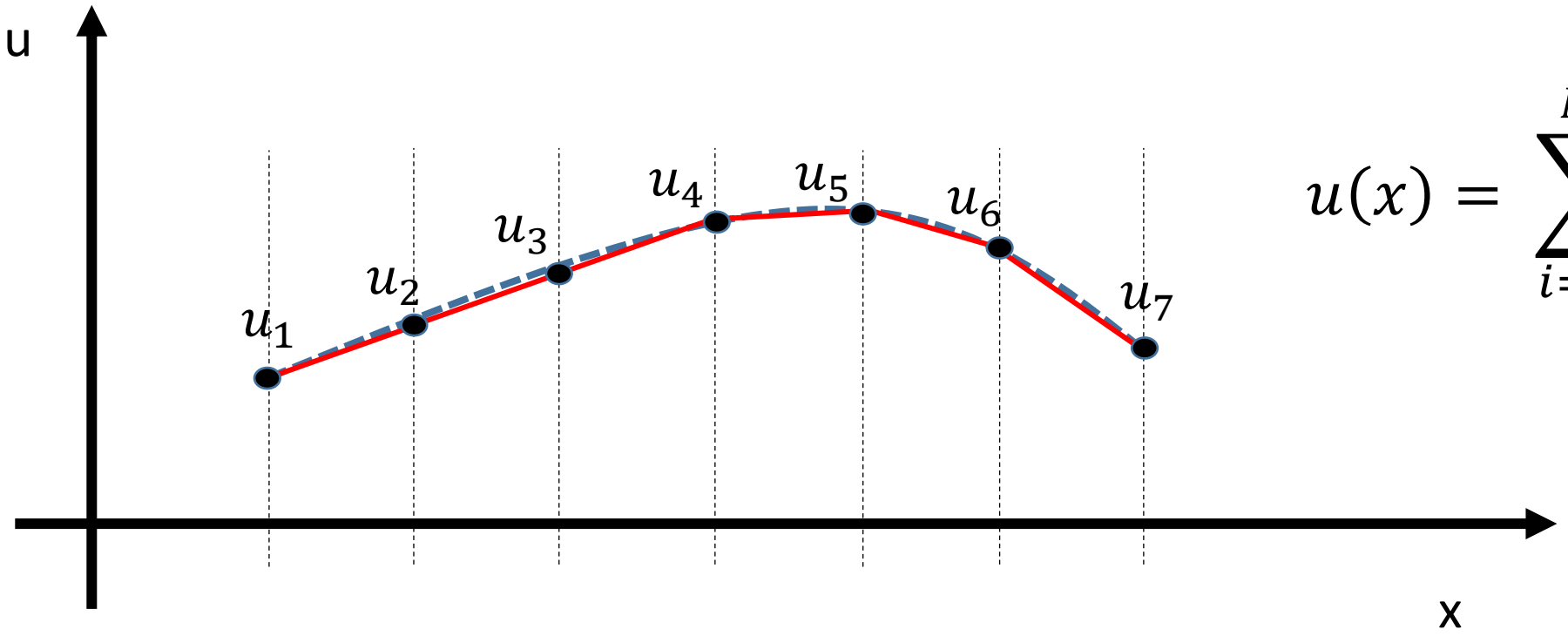
Then, the finite-dimensional space W of functions that we want to approximate the solution will look like this:



In total 6 elements, 12 linear base functions

1D Heat with FEM

- Picewise linear function:



$$u(x) = \sum_{i=1}^{N} u_i w_i$$

1D Heat with FEM

- We can evaluate the matrix K_e in each element (two local nodes: 1,2)

$$K_{11} = a(w_1, w_1) = \int_{x_0}^{x_1} \frac{dw_1}{dx} A\kappa \frac{dw_1}{dx} dx = \int_{x_0}^{x_1} -\frac{1}{L_e} A\kappa \cdot \left(-\frac{1}{L_e}\right) dx = \frac{A\kappa}{L_e}$$

$$K_{22} = a(w_2, w_2) = \int_{x_0}^{x_1} \frac{dw_2}{dx} A\kappa \frac{dw_2}{dx} dx = \int_{x_0}^{x_1} \frac{1}{L_e} A\kappa \cdot \left(\frac{1}{L_e}\right) dx = \frac{A\kappa}{L_e}$$

$$K_{12} = a(w_1, w_2) = \int_{x_0}^{x_1} \frac{dw_1}{dx} A\kappa \frac{dw_2}{dx} dx = \int_{x_0}^{x_1} \frac{1}{L_e} A\kappa \cdot \left(-\frac{1}{L_e}\right) dx = -\frac{A\kappa}{L_e}$$

$$= K_{21}$$

$$K_e = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} = \frac{A\kappa}{L_e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

1D Heat with FEM

- Now compute the right hand side component:
- Element 1:

$$l(w_1) = -w_1 q(x=2) + \int_{x_0}^{x_1} f w_1 dx = 0 + \int_{x_0}^{x_1} 1 \cdot w_1 \, dx = \frac{L}{2}$$

$$l(w_2) = \int_{x_0}^{x_1} f w_1 dx = \int_{x_0}^{x_1} 1 \cdot w_2 dx = \frac{L}{2}$$

- Element 2:

$$l(w_1) = \int_{x_0}^{x_1} f w_1 dx = \int_{x_0}^{x_1} 1 \cdot w_1 \, dx = \frac{L}{2}$$

$$l(w_2) = \int_{x_0}^{x_1} f w_1 dx = \int_{x_0}^{x_1} 1 \cdot w_2 dx = \frac{L}{2}$$

- ……

1D Heat with FEM

- For each element we get an equation $K_e \cdot u_e = l_e$:

$$\begin{bmatrix} K_{1_{11}} & K_{1_{12}} \\ K_{1_{21}} & K_{1_{22}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} l_1 \\ l_2 \end{bmatrix}$$

$$\begin{bmatrix} K_{2_{11}} & K_{2_{12}} \\ K_{2_{21}} & K_{2_{22}} \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} l_2 \\ l_3 \end{bmatrix}$$

$$\begin{bmatrix} K_{3_{11}} & K_{3_{12}} \\ K_{3_{21}} & K_{3_{22}} \end{bmatrix} \begin{bmatrix} u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} l_3 \\ l_4 \end{bmatrix}$$

.....

- Define a global vector of temperatures at the nodes

$$U = [u_1\ u_2\ u_3\ \ \ldots . u_N]^T,$$

1D Heat with FEM

- Re-arrange & assembly the equations to a single matrix, size of NxN:

$$\begin{bmatrix} K_{1_{11}} & K_{1_{12}} & & \\ K_{1_{21}} & K_{1_{22}} + K_{2_{11}} & K_{2_{12}} & \dots \\ & K_{2_{21}} & K_{2_{22}} + K_{3_{11}} & \\ & \dots & & \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \end{bmatrix} = \begin{bmatrix} l_1 \\ l_1 + l_2 \\ l_2 + l_3 \\ \dots \end{bmatrix}$$

Steps:

- Determine the local matrix
- Determine the global DOF (degree of freedom) corresponding to local DOF
- Add the components of local matrix to the rows and columns of global matrix
- Repeat this for all elements

1D Heat with FEM

- In the case of our bar, assuming euqal lengths of all elements & homogeneous $\kappa(x), A(x)$, we get:

$$K = \frac{A\kappa}{L}\begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & \ddots & & \\ & & \ddots & \ddots & -1 & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}, \qquad L = \begin{bmatrix} \frac{L}{2} \\ L \\ \vdots \\ L \\ L \\ \frac{L}{2} \end{bmatrix}$$

- But: $u_a = u_b = 0 \quad \rightarrow \quad$ eliminate two equations (first and last)

1D Heat with FEM

- Example Matlab code

```
% Simulation of heat transfer in 1D bar:

x0 = 2;  % begining of a bar on x-axis
x1 = 5; % end of a bar on x-axis
N =50; % number of elements
k = 15; % heat conductivity (constant)
A = 1; % bar crossection area (constant)

Le = (x1-x0)/N;  % length of single element

% Generate mass matrix:
K = sparse(N+1,N+1);
L = zeros(N+1,1);

for p=1:N
    Ke = (k*A/Le)*[1 -1; -1 1];
    K(p:p+1,p:p+1) = K(p:p+1,p:p+1)+Ke;
    L(p:p+1) = L(p:p+1)+Le/2;
end
```

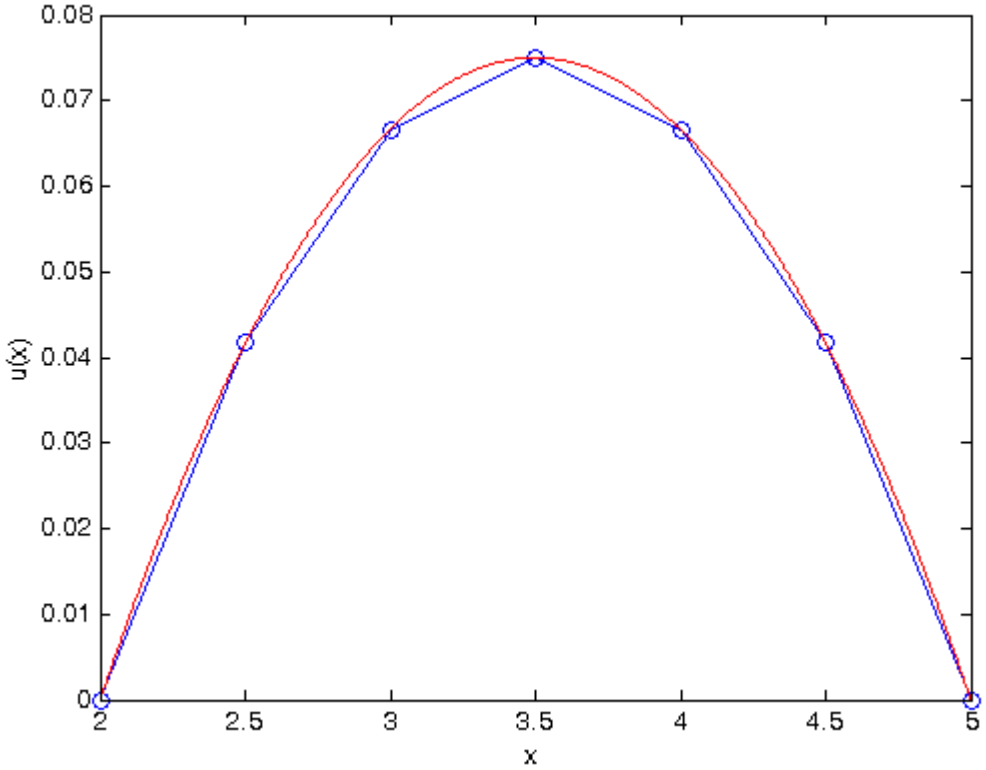
```
% enforce zero on boundaries (u=0)
K = K(2:end-1,2:end-1);
L = L(2:end-1,1);

% solve equation:
U = K\L;
U = [0; U; 0];  % add the zeros on boundaries

x = linspace(x0,x1,N+1);
plot(x,U,'b-o');

% plot analytic solution u(x) = (1/(2*k))*(-x^2+7x-10)
Nt = 1001;
xx = linspace(x0,x1,Nt);
ut = (1/2/k)*(-xx.^2+7*xx-10);
hold on;
plot(xx,ut,'r');
```

1D Heat with FEM

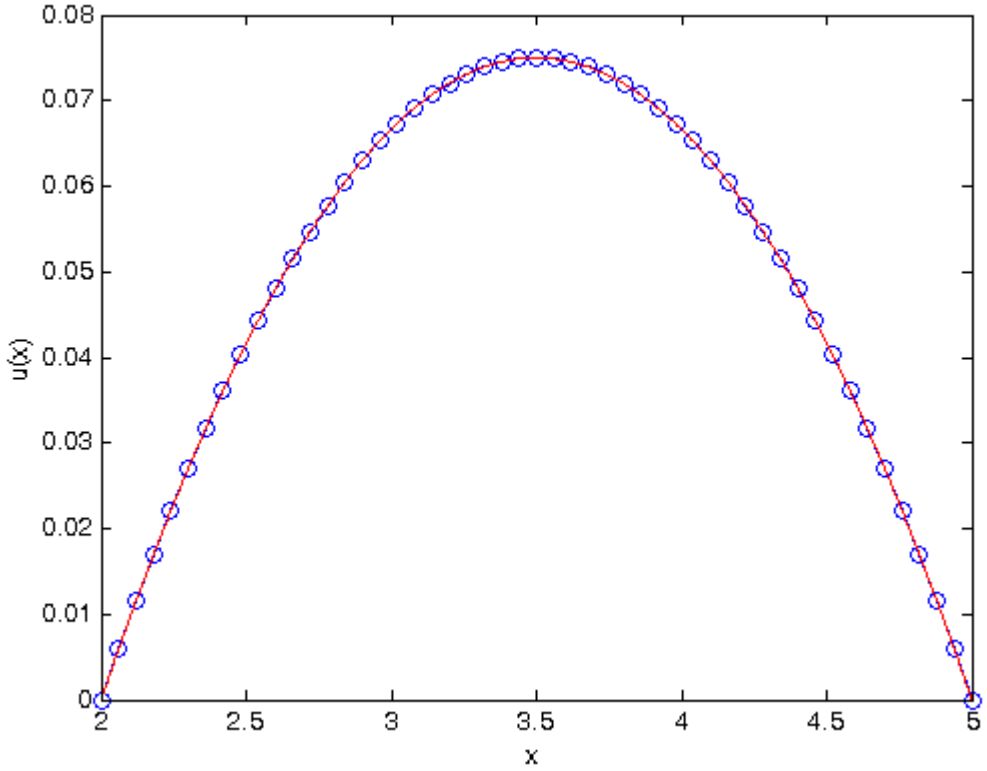
- Results

Analytic solution (WolframAlpha)

$$u(x) = \frac{1}{2\kappa}(x^2 - 7x + 10)$$



N=6



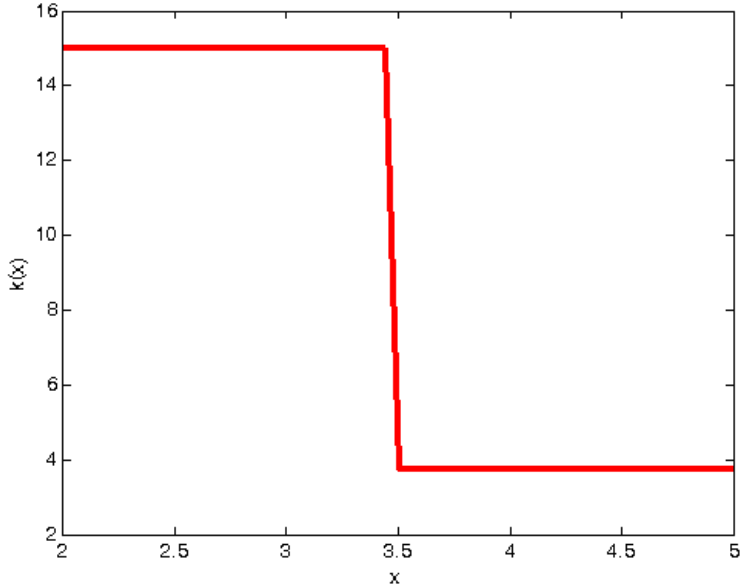
N=51

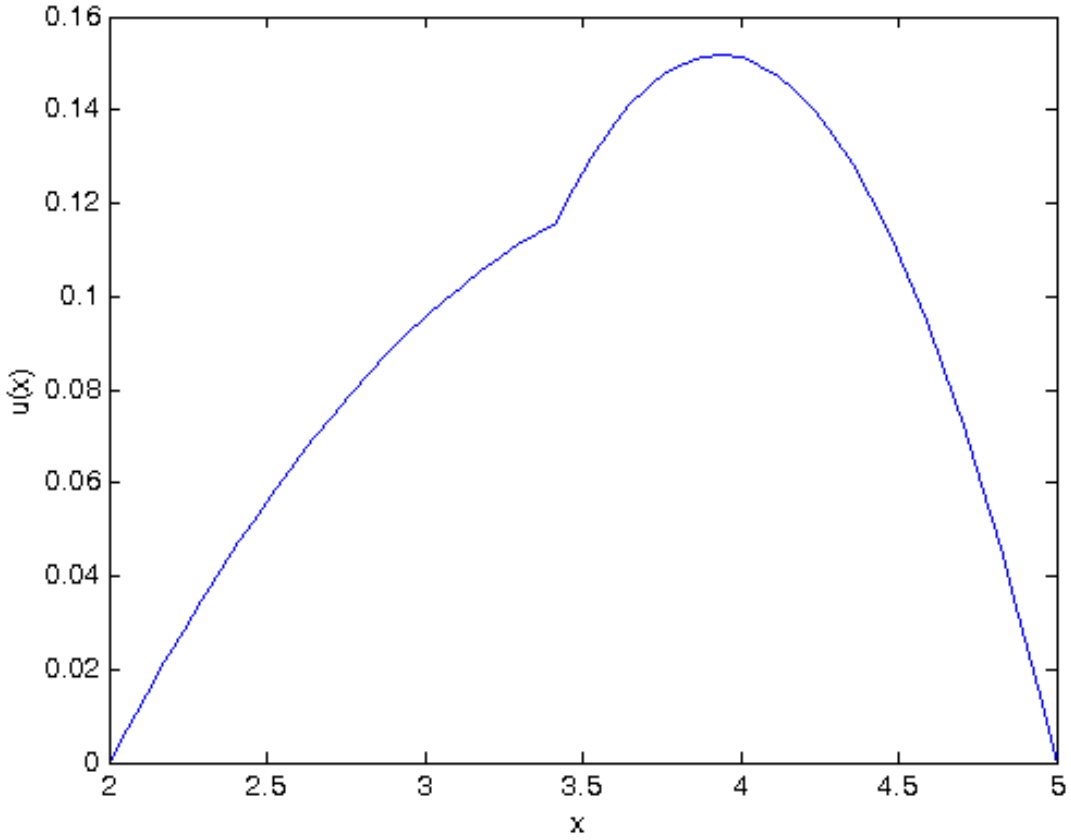
1D Heat with FEM

- Great advatage of FEM solution is flexibility: we can easily find solutions for non-homogeneous conductivity $\kappa(x)$ and/or crossection $A(x)$:

Non-uniform conductivity:

$$\kappa(x) = \begin{cases} 15 & x \in < 2, 3.5) \\ 3.75 & x \in < 3.5, 5 > \end{cases} \quad \left[\frac{W}{mK}\right]$$





1D Heat with FEM

- Matlab code modification:

```
% Simulation of 1D heat transfer in bar:
% with inhomogeneous thermal conductivity
clear all;

x0 = 2;  % begining of a bar on x-axis
x1 = 5; % end of a bar on x-axis
N = 50; % number of elements
k = 15; % heat conductivity (constant)
A = 1; % bar crossection area (constant)

Le = (x1-x0)/N;  % length of single element

% Generate mass matrix:
K = sparse(N+1,N+1);
L = zeros(N+1,1);
ke = zeros(N+1,1);

% define the profile of conductivity:
ke(1:25) = k;
ke(26:end)=k/4;  % ke(i) defines conductivity of i-th element!!!
```

```
for p=1:N
    Ke = (ke(p) *A /Le)*[1 -1; -1 1];
    K(p:p+1,p:p+1) = K(p:p+1,p:p+1)+Ke;
    L(p:p+1) = L(p:p+1)+Le/2;
end

% enforce zero on boundaries (u=0)
K = K(2:end-1,2:end-1);
L = L(2:end-1,1);

% solve equation:
U = K\L;
U = [0; U; 0];  % add the zeros on boundaries

x = linspace(x0,x1,N+1);
plot(x,U,'b');
```

In the same manner one can introduce variable A(x)

1D Heat with FEM

- Task: create a temperature profile calculator for a brick wall with styrofoam insulation
- Assumptions:
  - Inside temperature $u_a = 20°C$
  - Temperature outside $u_b = -5°C$
  - Wall consists of two layers:
    - Layer A (inside) brick wall thickness 0.4m, thermal conductivity $\kappa = \ 1.31 \ \frac{W}{m\,K}$
    - Layer B (outside) insulation 0.1m, thermal conductivity $\kappa = \ 0.036 \ \frac{W}{m\,K}$

- How to do this?
  - Modify our program to support non-zero Dirichlet boundary conditions!!

1D Heat with FEM

- We know the values of temperature at the boundary nodes → we can directly enforce the weights of base functions
  - Base function is local => is zero in all remaining nodes
  - Inside the wall there is no heat source => $f = 0$ => RHS is zero!

$$K \cdot U = 0$$

- Set values of u_a for $x = a = 0$ and u_n for $x = b = 0.5$

$$U = \begin{bmatrix} Uf \\ Uc \end{bmatrix}, \mathrm{K} = \begin{bmatrix} K_{ff} & Kfc \\ Kcf & Kcc \end{bmatrix}, \qquad U_f \ - unknown, Uc \ - prescribed$$

$$Uf = -K_{ff}^{-1} \cdot (K_{fc} U_c)$$

- Solve the system!

1D Heat with FEM

- Imagine we have 5x5 linear system $Ax = b$, $\quad x = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5]$
- Enforce $x_1 = 1, x_4 = 1 \;\rightarrow\; ?$
- Transform the system to 2 blocks of unknowns (change ordering):

$$A' = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix}, \qquad x' = \begin{bmatrix} x_2 \\ x_3 \\ x_5 \\ x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_f \\ x_c \end{bmatrix}, \qquad b = \begin{bmatrix} b_2 \\ b_3 \\ b_5 \\ b_1 \\ b_4 \end{bmatrix} = \begin{bmatrix} b_f \\ b_c \end{bmatrix}$$

$$\begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{bmatrix} x_f \\ x_c \end{bmatrix} = \begin{bmatrix} b_f \\ b_c \end{bmatrix} \quad \rightarrow \quad \begin{aligned} A_{ff}x_f + A_{fc}x_c &= b_f \\ A_{cf}x_f + A_{cc}x_c &= b_c \end{aligned}$$

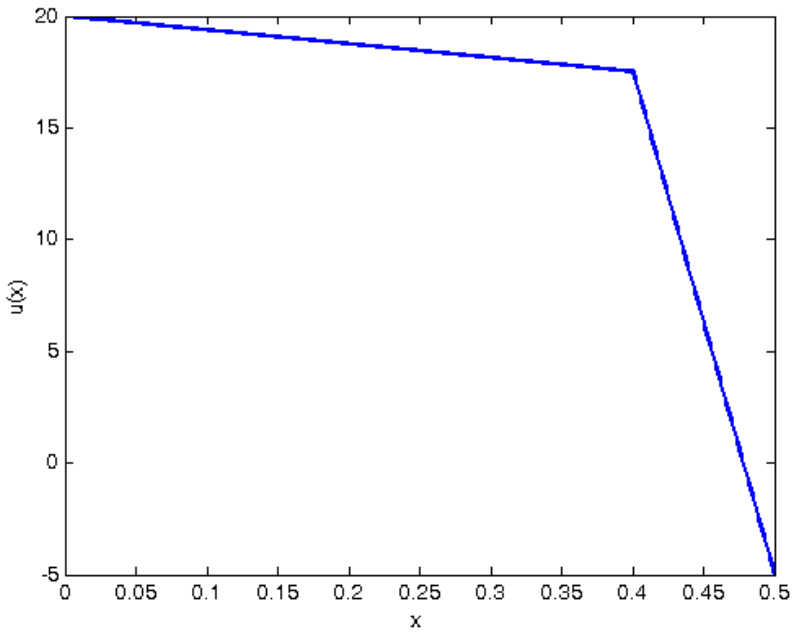
$$x_f = A_{ff}^{-1} \cdot (b_f - A_{fc}x_c)$$

1D Heat with FEM

- Matlab code:

```
% Simulation of 1D heat transfer in insulated wall:
clear all;

x0 = 0;  % begining of a bar on x-axis
x1 = 0.5; % end of a bar on x-axis
N = 101; % number of elements
k1 = 1.31; % heat conductivity - brick (constant)
k2 = 0.036;  % heat conductivity - styrofoam
A = 1; % unitary crossection
u0 = 20; % temperature inside
u1 = -5; % temperature outside
xc = 0.4; % position of border between  brick and insulation (x-axis)

Le = (x1-x0)/N;  % length of single element
x = linspace(x0,x1,N+1); % nodes on x-axis

% Generate mass matrix:
K = sparse(N+1,N+1);
L = zeros(N+1,1);
ke = zeros(N+1,1);

% define the profile of conductivity of our wall:

idx  = find(x>0.4);
ke(:) = k1;
ke(idx) = k2;
```

```
for p=1:N
    Ke = (ke(p) *A /Le)*[1 -1; -1 1];
    K(p:p+1,p:p+1) = K(p:p+1,p:p+1)+Ke;
    L(p:p+1) = L(p:p+1)+Le/2;
end

% create free and fixed nodes sub-matrices
Kff = K(2:end-1,2:end-1);
Uc = [u0 u1]';
idx = [1 , N+1];
Kfc = K(2:end-1,idx);

% solve equation:
U = -Kff\(Kfc*Uc);
U = [u0; U; u1];  % add the zeros on boundaries

plot(x,U,'b');
```



Modelowanie i Symulacja Systemów

Introduction to Finite Element Methods

Adam Lamecki

Series of lectures on FEM

- Introduction
- Strong vs. weak form of BVP
- Steady-state heat transfer in 1D
- **Transient heat transfer in 1D**
- **Heat transfer in 2D & 3D**
- Solution of solid mechanics problems with FEM (?)

Heat transfer

- General heat transfer equation (3D):

$$c_p \rho \frac{\partial u}{\partial t} = \nabla \cdot (\kappa \, \nabla \, u) + f$$

where c_p - heat capacity, ρ – mass density

- Last week it was shown how to solve this equation in 1D with FEM for steady state:

$$\frac{\partial u}{\partial t} = 0$$

- We assumed the solution has a form

$$u(x) = \sum_{i=1}^{N} u_i w_i$$

- We derived the linear problem $K \cdot U = F$, where U is a vector of unknown weigths of base functions u_i.

Unsteady Heat Transfer Problem

- In steady state $\frac{\partial u}{\partial t} = 0$
- In time-dependent problem $\frac{\partial u}{\partial t} \neq 0$
- Need to solve full, time dependent heat equation:

$$c_p \rho \frac{\partial u}{\partial t} = \nabla \cdot (\kappa \nabla u) + f \quad \rightarrow \quad u = u(t, x, y, z), f = f(t, x, y, z)$$

→ need to add time dependence to our solution:

$$u(t, x, y, z) = \sum_i \boxed{u_i(t)} \cdot \boxed{w_i(x, y, z)} = \sum_i u_i(t) \cdot w_i$$

Temperature at mesh nodes depends on time

Unsteady Heat Transfer Problem

- Follow the general rule to get a weak-form:
  - Multiply with w_j, integrate over Ω and apply Green identity to reduce the order of equation:

$$\int_\Omega \left(c_p \rho \frac{\partial u}{\partial t} - \nabla \cdot (\kappa \nabla u) - f \right) w_j d\Omega = 0$$

$$\int_\Omega c_p \rho \frac{\partial u}{\partial t} w_j \mathrm{d}\Omega - \int_\Omega \nabla \cdot (\kappa \nabla u) w_j d\Omega - \int_\Omega f w_j d\Omega = 0$$

$$\underbrace{\int_\Omega c_p \rho \frac{\partial u}{\partial t} w_j \mathrm{d}\Omega}_{\text{New component}} - \underbrace{\int_\Omega \nabla u \cdot \kappa \cdot \nabla w_j \, \mathrm{d}\Omega - \int_\Omega f w_j \, d\Omega - \kappa w_j (\nabla u \cdot \vec{n}) d\Omega}_{\text{The same as in steady-state}} = 0$$

Unsteady Heat Transfer problem

- Let's analyze new component:

$$\int_{\Omega} c_p \rho \frac{\partial u}{\partial t} w_j \mathrm{d}\Omega = \int_{\Omega} c_p \rho \frac{\partial \left(\sum_{i=1}^{n} u_i(t) \cdot w_i\right)}{\partial t} w_j \mathrm{d}\Omega$$

- Define matrix M as:

$$m_{ij} = \int_{\Omega} c_p \rho \, w_i w_j d\Omega$$

- Then we can write:

$$\int_{\Omega} c_p \rho \frac{\partial u}{\partial t} w_j \mathrm{d}\Omega = \sum_{i=1}^{n} m_{ij} \left\{ \frac{\partial u_i}{\partial t} \right\}$$

Do this for j=1..n

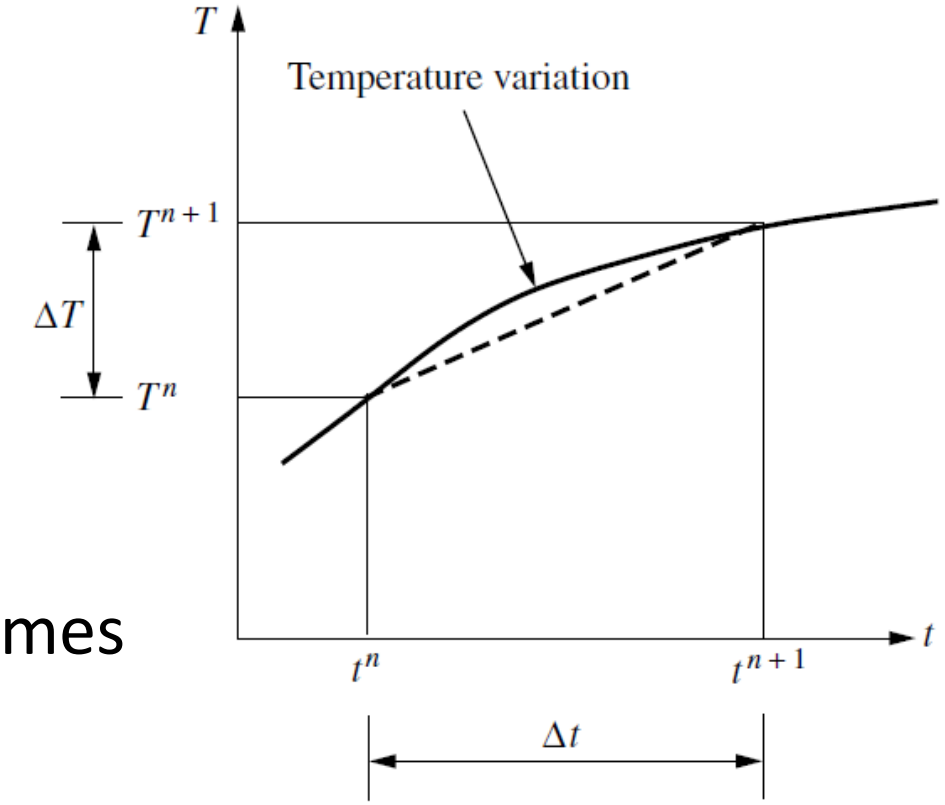
Unsteady Heat Transfer problem

- We can combine this with formerly derived steady-state components, as

$$M\left\{\frac{\partial U}{\partial t}\right\} + K \cdot U = \mathrm{F}$$

- Now it is time to introduce a time discretization:

$$T^{n+1} = T^n + \Delta t \frac{\partial T}{\partial t}^n + \frac{\Delta t^2}{2}\frac{\partial^2 T^n}{\partial t^2} + \cdots$$



- We can use variety of finite-difference schemes to apply time-stepping

Unsteady Heat Transfer problem

- Euler forward scheme:

$$U^n = U^{n-1} + \Delta t^n \, M^{-1}(F^{n-1} - K \, U^{n-1})$$

Conditionally stable

(Δt^n denotes time-step in *n*-th iteration)

- Euler backward scheme:

$$M \, \frac{(U^n - U^{n-1})}{\Delta t^n} + K \, U^n = F^n$$

$$(M + \Delta t^n K)U^n = MU^{n-1} + \Delta t^n F^n$$

Unconditionally stable

Unsteady Heat Transfer problem

- Crank-Nicolson scheme
  - Second order accurate
  - Conditionally stable

$$M\frac{U^n - U^{n-1}}{\Delta t_n} + K\frac{U^n + U^{n-1}}{2} = \frac{F^n - F^{n-1}}{2}$$

- Many more….

Unsteady Heat Transfer problem in 1D

- If we transform our former 1D steady state solution to transient anaysis, we notice that we need to evaluate and assembly matrix M
- In each element

$$m_{11} = \int_{x_0}^{x_1} c_p \rho \, w_1 w_1 d\Omega = \frac{c_p \rho A L}{3} = m_{22}$$

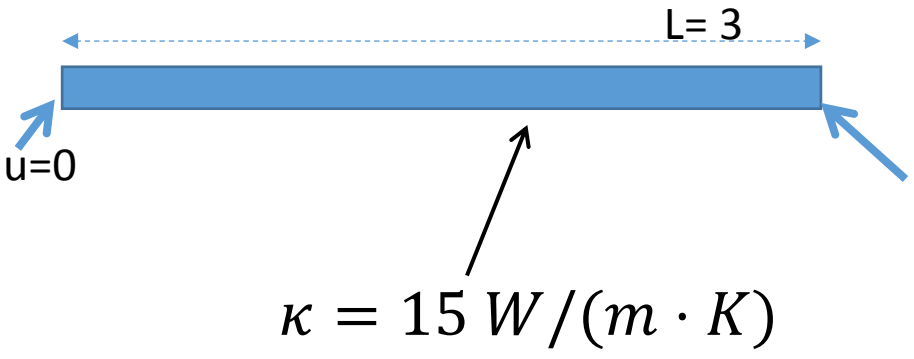
$$m_{12} = \int_{x_0}^{x_1} c_p \rho \, w_1 w_2 d\Omega = \frac{c_p \rho A L}{6} = m_{21}$$

$$M_e = \frac{c_p \rho A L}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Unsteady Heat Transfer in 1D

- Imagine a similar problem as we solved in steady state:

  1D bar of length 3m, with $\kappa = 15\,\frac{W}{mK}$, and initial temperature $u_{(t=0)} = 0°C$, with enforced temperatures at both ends equal $0°C$ and in $t = 0$ we turn on a uniform heat source $f(t) = 1$. Assume $c_p\rho = 1$. Find $u(t)$.



$$\Omega = (2,5)$$

$$\mathrm{f(t)} = \begin{cases} 0 & t < 0 \\ 1 & t \geq 0 \end{cases}$$

$$c_p\rho\frac{\partial u(t)}{\partial t} + \frac{d}{dx}\,A\kappa\frac{du(t)}{dx} = f(t),$$

$$u = \begin{cases} 0 & x = 2 \\ 0 & x = 5 \end{cases}$$

Unsteady Heat Transfer in 1D

- Matlab code:

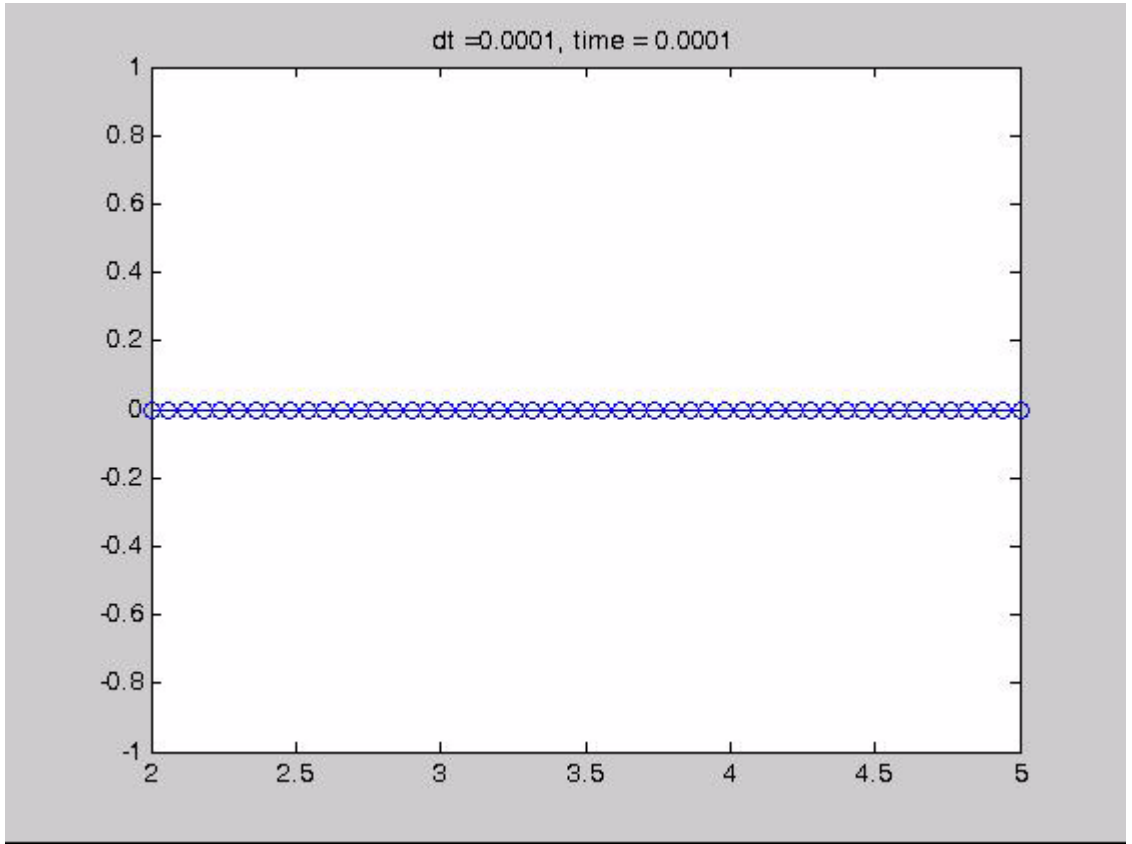
```
% Simulation of heat transfer in 1D bar:

x0 = 2;  % begining of a bar on x-axis
x1 = 5; % end of a bar on x-axis
N =50; % number of elements
k = 15; % heat conductivity (constant)
A = 1; % bar crossection area (constant)

Le = (x1-x0)/N;  % length of single element
x = linspace(x0,x1,N+1);

% Generate matrices:
K = sparse(N+1,N+1);
M = sparse(N+1,N+1);
L = zeros(N+1,1);

for p=1:N
    Ke = (k*A/Le)*[1 -1; -1 1];
    Me = (A*Le/6)*[2 1; 1 2];
    K(p:p+1,p:p+1) = K(p:p+1,p:p+1)+Ke;
    M(p:p+1,p:p+1) = M(p:p+1,p:p+1)+Me;
    L(p:p+1) = L(p:p+1)+Le/2;
end
```

```
% enforce zero on boundaries (u=0)
K = K(2:end-1,2:end-1);
M = M(2:end-1,2:end-1);
L = L(2:end-1,1);

% prepare to saveing animation
vidObj = VideoWriter('heat1D.avi');
open(vidObj);

U0(1:N+1,1) = 0;  % initial temperature
U0 = U0(2:end-1);  % remove first and last equation
Q = 400; % number of time steps
dt = 1e-3;  % set time step:
figure;
for p=1:Q

   U1 = (M+dt*K)\(M*U0 + dt*L); % apply Euler backward method
   U0 = U1;

   plot(x,[0; U1; 0] ,'b-o');
   ylim([0 0.1]);
   str  = ['dt =' num2str(dt) ', time = ' num2str(p*dt)]
   title(str);
  currFrame = getframe(gcf);
  writeVideo(vidObj,currFrame);

end

close(vidObj);

% now can play movie from disk
 implay('heat1D.avi')
```

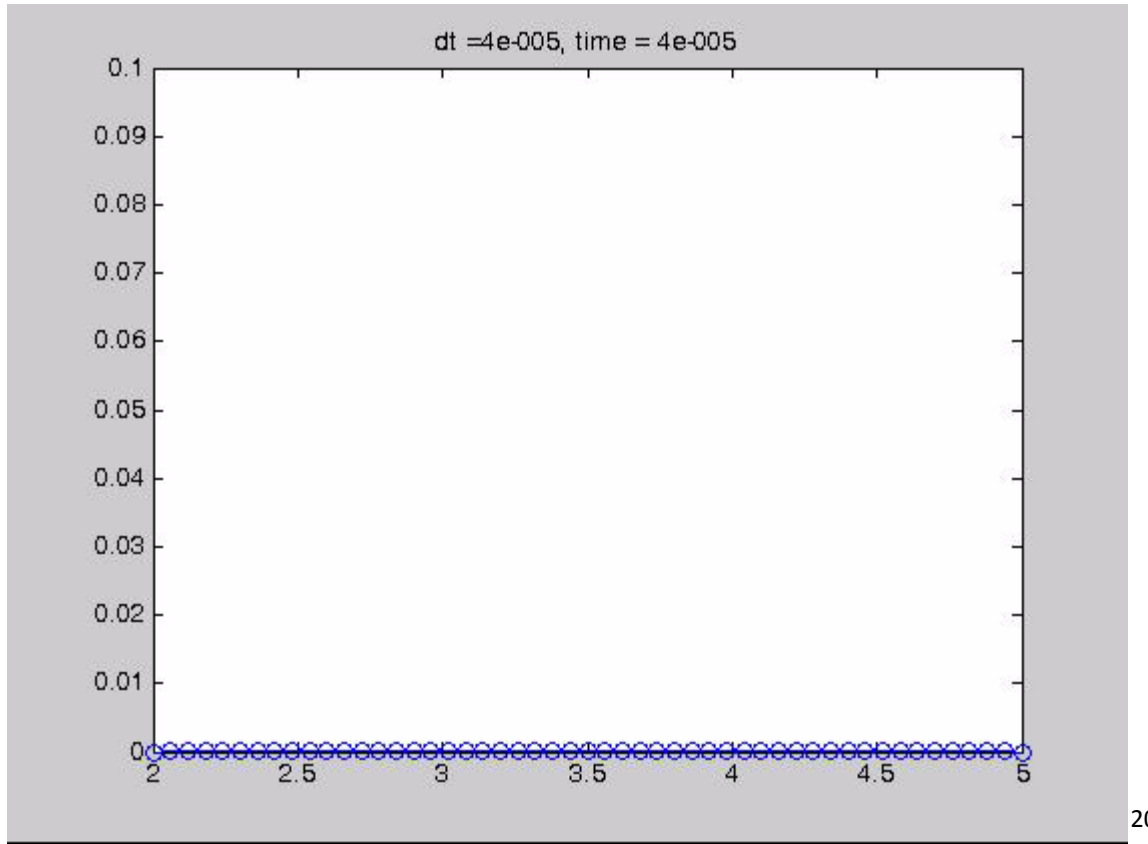
Unsteady Heat Transfer in 1D

- Results:

forward Euler, $\Delta t = 10^{-4}$

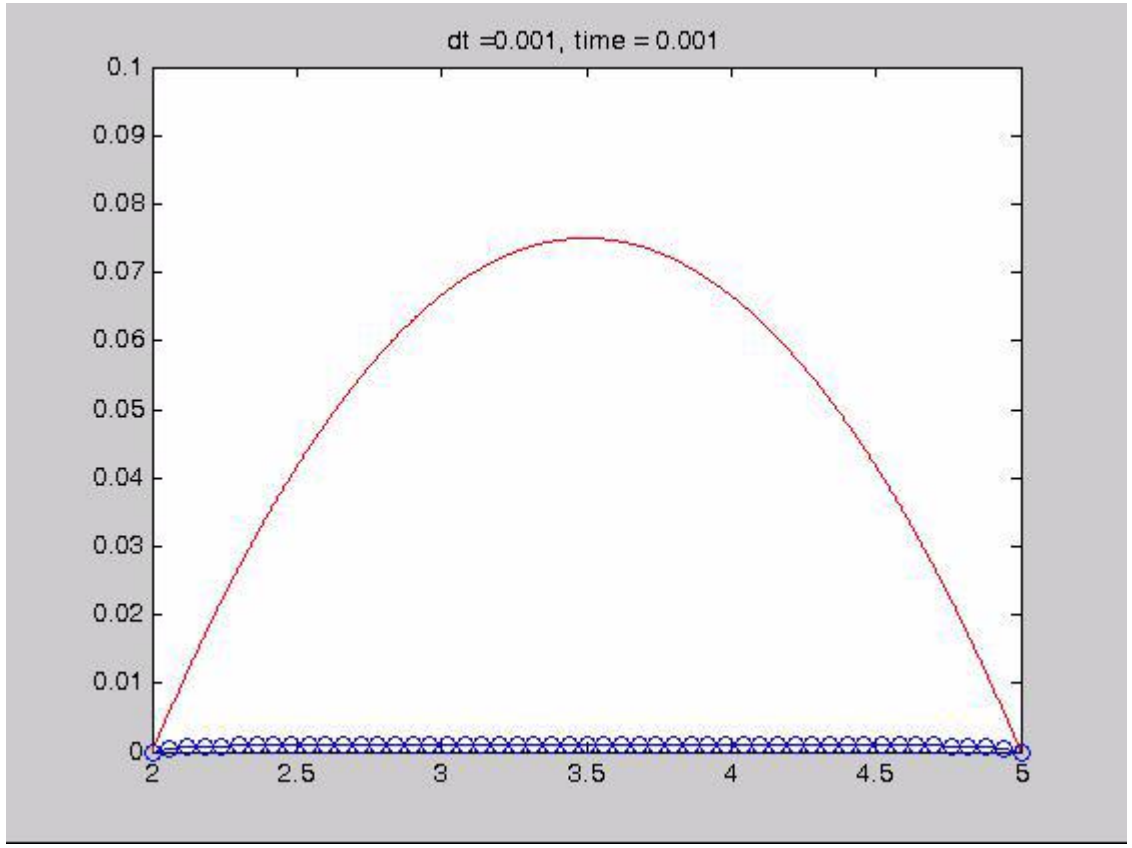


forward Euler, $\Delta t = 4 \cdot 10^{-5}$

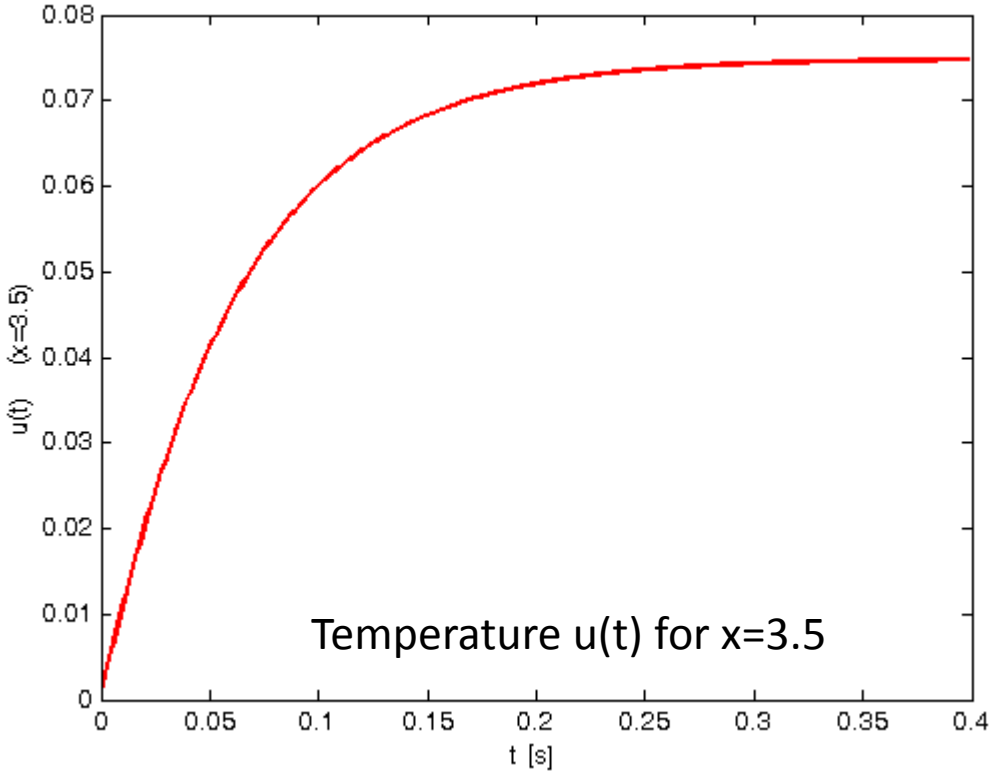


Unsteady Heat Transfer in 1D

- Results: backward Euler, $\Delta t = 10^{-3}$



The solution should converge to steady state.



Unsteady Heat Transfer in 1D

- As in steady-state, we can easily define inhomogeneous thermal conductivity and/or varable bar cross-section. Also one can introduce a variation of heat source in time and space, for example modify

$$\Omega = (2,5)$$

$$\mathrm{f(t)} = \begin{cases} 0 & t < 0 \\ \sin(2\pi t) & t \geq 1 \end{cases}$$

$$\kappa(x) = \kappa_0 \cdot \left[1 + \frac{(x-2)}{3}\right], \qquad \kappa_0 = 15\frac{W}{m\,K}$$

$$A(x) = x;$$

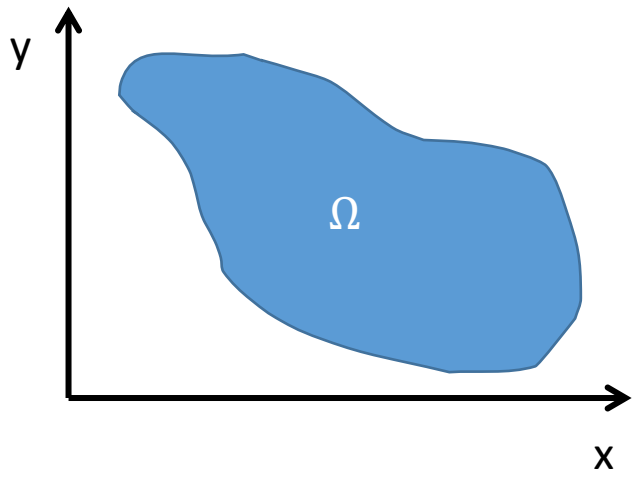
$$c_p \rho \frac{\partial u(t)}{\partial t} + \frac{d}{dx} A(x)\kappa(x)\frac{du(t)}{dx} = f(t),$$

$$u = \begin{cases} 0 & x = 2 \\ 0 & x = 5 \end{cases}$$

Q: What need to be modified?

Q: And what if we define $\rho = \rho(x)$?

FEM in 2D



- Let's investigate the 2D heat transfer → $u = u(x, y, t)$
- In general, our weak formulation remains the same:

$$\int_\Omega c_p \rho \frac{\partial u}{\partial t} w_j \mathrm{d}\Omega - \int_\Omega \nabla u \cdot \kappa \cdot \nabla w_j \, \mathrm{d}\Omega - \int_\Omega f w_j \, d\Omega - \kappa w_j \, (\nabla u \cdot \vec{n}) d\Omega = 0$$

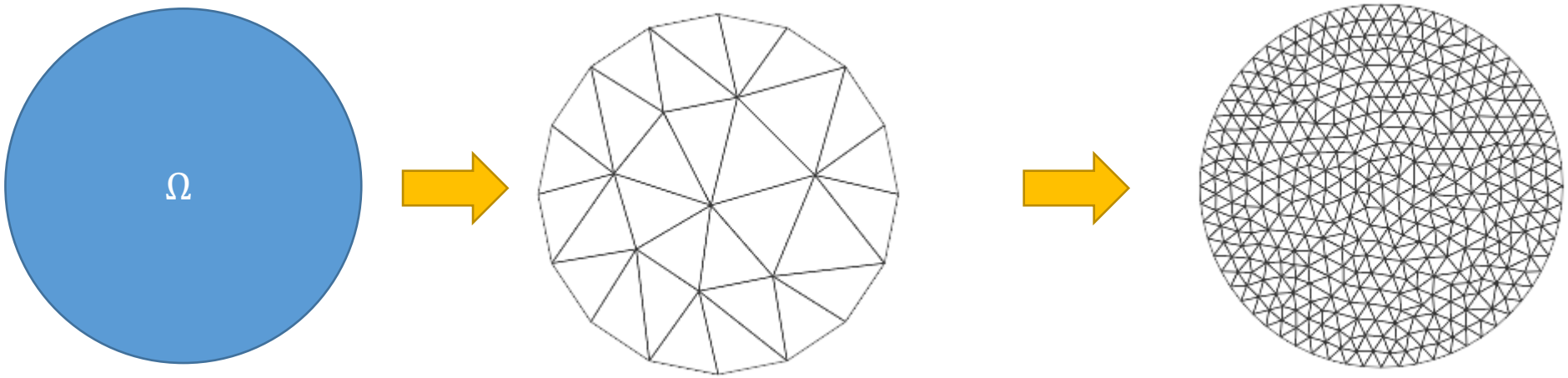
but we need to introduce domain discretization and base functions in 2D:

$$u(x, y, t) = \sum_i u_i(t) \cdot w_i(x, y)$$

- There are two basic choices here: divide our domain with rectangular or triangular elements
  - Can use regular or non-regular grid

FEM in 2D

- Discretization of space with triangular elements (triangular mesh):
  - Area is approximated with sum of 2D triangle elements ($\Omega = \sum_i e_i$)



coarse mesh

(high discretization error)

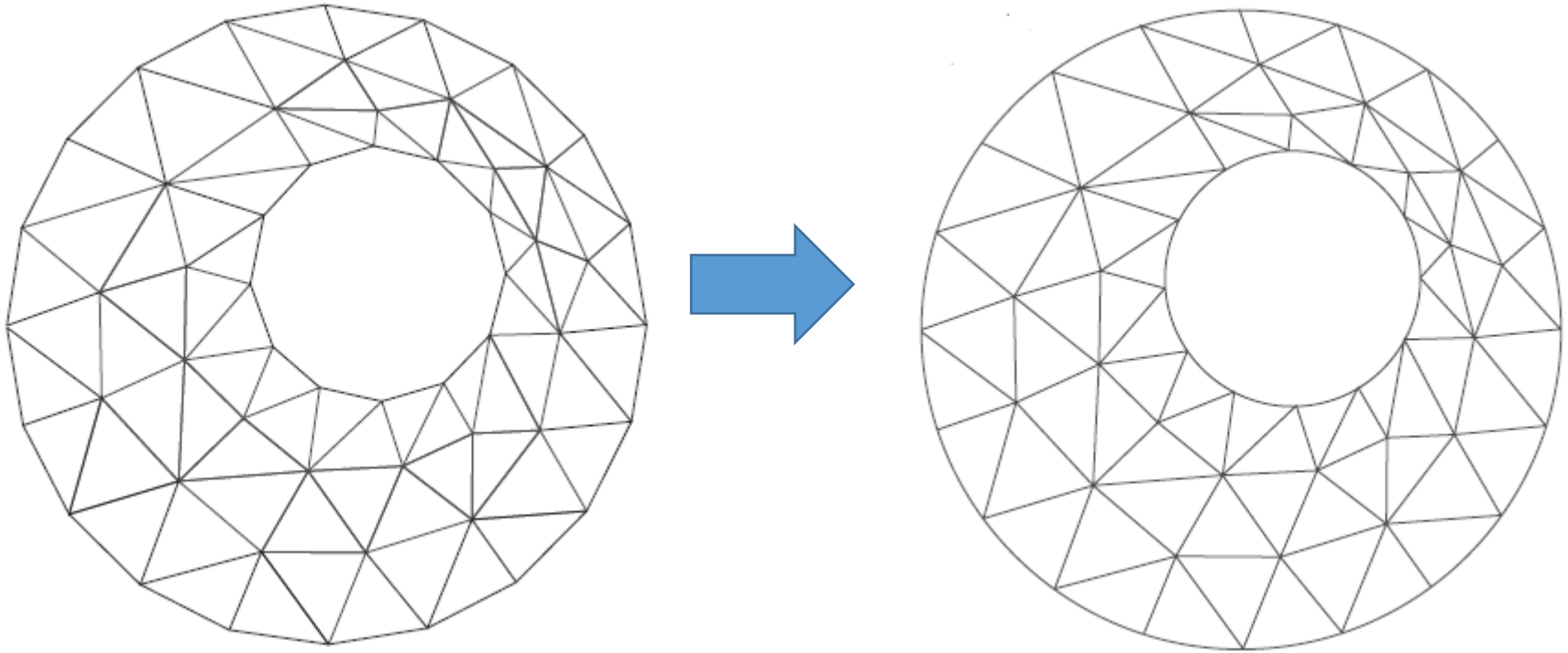
fine mesh

(lower discretization error)

- Usually to generate domain discretization a specialized software called ***mesh generator*** is applied

FEM in 2D

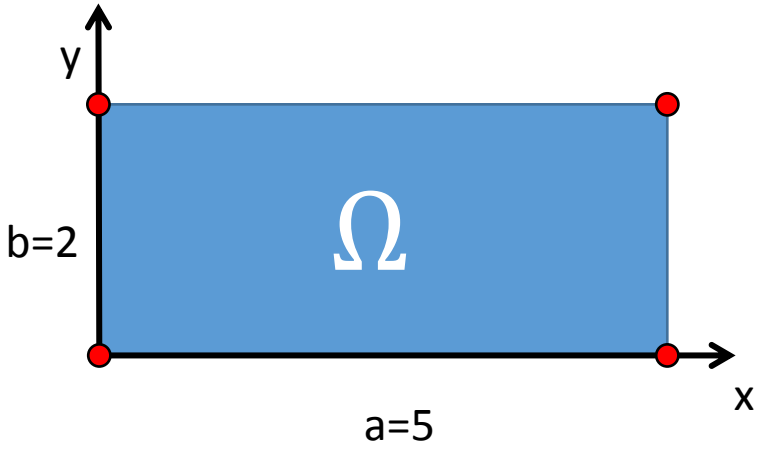
- How to improve geometry representation?
  → Use higher order meshes (curvilinear elements)



mesh edges are not lines (first order)
but parabolic (second-order)

2D FEM

- Advanced mesh generators can directly use files in CAD format as input (such as DXF or STEP)
- In case of 2D mesh, the domain Ω can be also specified with a dicrete set of node coordinates and edges connecting the nodes (a closed polyline contour).
- Example: A rectangle with a=5, b=2 - ***Mesh2D*** generator* (GitHub, for Matlab)




```
% nodes array in (x,y) plane:
nodes = [ 0, 0                  % outer square
          5, 0
          5, 2
          0, 2 ];

% array of edges - pairs of nodes
edges = [1 2;
         2 3;
         3 4;
         4 1];


h = 1; % maximum mesh edge length

[vert,etri, tria,tnum] = refine2(nodes,edges,[],[],h) ;

% show mesh

figure;
patch('faces',tria(:,1:3),'vertices',vert, ...
    'facecolor','w', ...
    'edgecolor',[.2,.2,.2]) ;
axis equal;
```

*https://github.com/dengwirda/mesh2d

[1] - Darren Engwirda, Locally-optimal Delaunay-refinement and optimisation-based mesh generation, Ph.D. Thesis, School of Mathematics and Statistics, The University of Sydney, September 2014.

FEM in 2D

- Mesh generator returns a list of: mesh nodes, mesh edges & triangles

```
vert =

    0.00000    0.00000
    5.00000    0.00000
    5.00000    2.00000
    0.00000    2.00000
    0.00000    1.00000
    5.00000    1.00000
    1.00000    0.00000
    4.00000    2.00000
    4.00000    0.00000
    1.00000    2.00000
    2.00000    0.00000
    3.00000    2.00000
    3.00000    0.00000
    2.00000    2.00000
    1.50000    0.86603
    2.50000    1.13397
    3.50000    0.86603
    0.73205    0.73205
    4.26795    0.73205
```

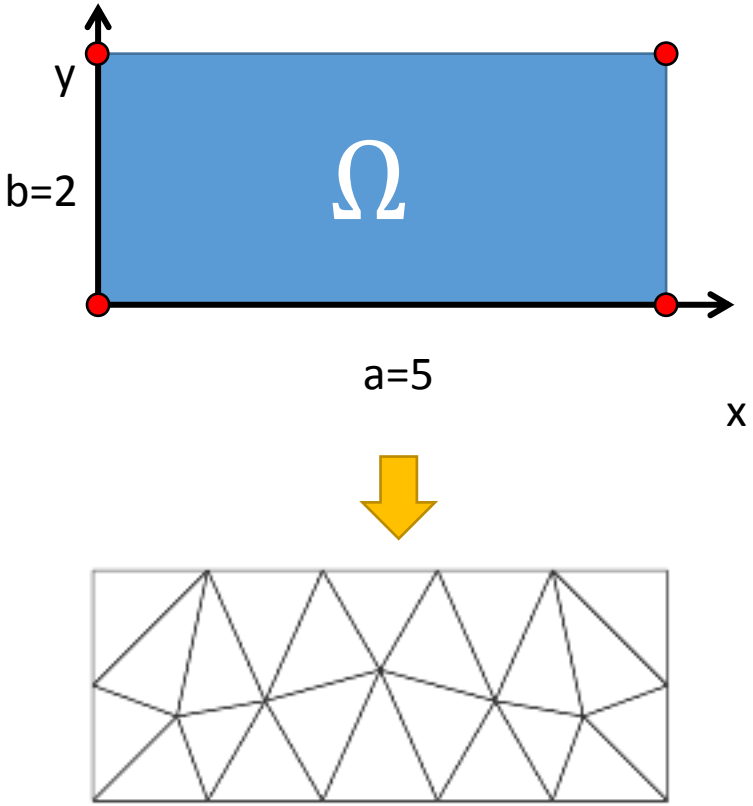
nodes

```
etri =

     1     5
     2     6
     4     5
     3     6
     1     7
     3     8
     2     9
     4    10
     7    11
     8    12
     9    13
    10    14
    11    13
    12    14
```

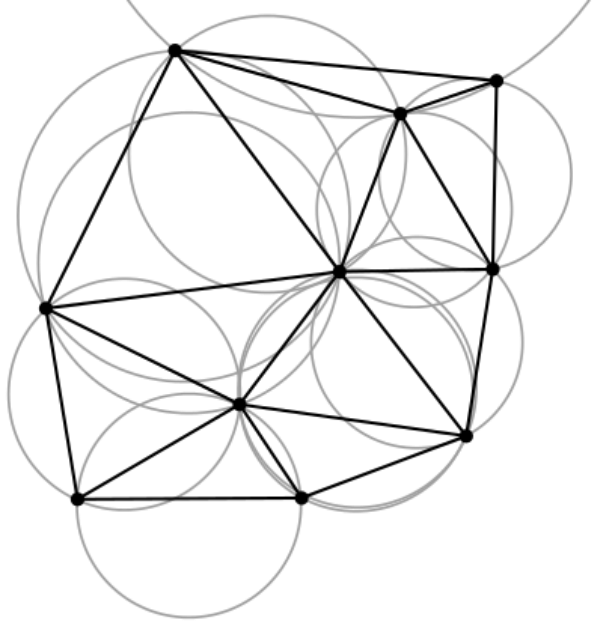
edges
(pairs of nodes)

```
tria =

    15    11    16
    13    16    11
    17    12    16
    17    16    13
    17    13     9
    18     5     1
    18    10     5
    18    15    10
    14    16    12
    14    15    16
    14    10    15
     4     5    10
    19    17     9
    19     2     6
    19     9     2
     8    12    17
     8    17    19
     8     6     3
     8    19     6
     7    18     1
     7    11    15
     7    15    18
```

triangles
(triplets of nodes)

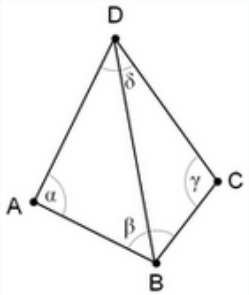


Delunay Triangulation (DT)

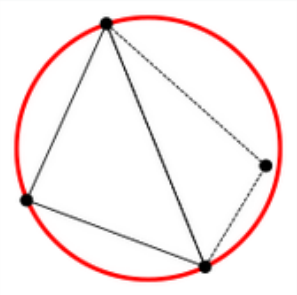


https://en.wikipedia.org/wiki/Delaunay_triangulation

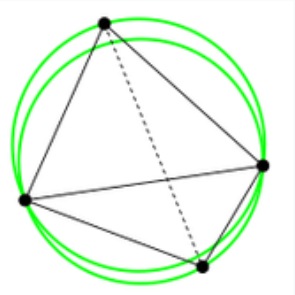
- How the mesh generator works in 2D?
  - First generate mesh on boundaries,
  - Add random points inside,
  - Have a set of points **P**
  - Create trangles in such a way that no other point from **P** is inside the circumcircle of any triangle
  - Then we can optimize the mesh to ensure regular elements



This triangulation does not meet the Delaunay condition (the sum of α and γ is bigger than 180°).



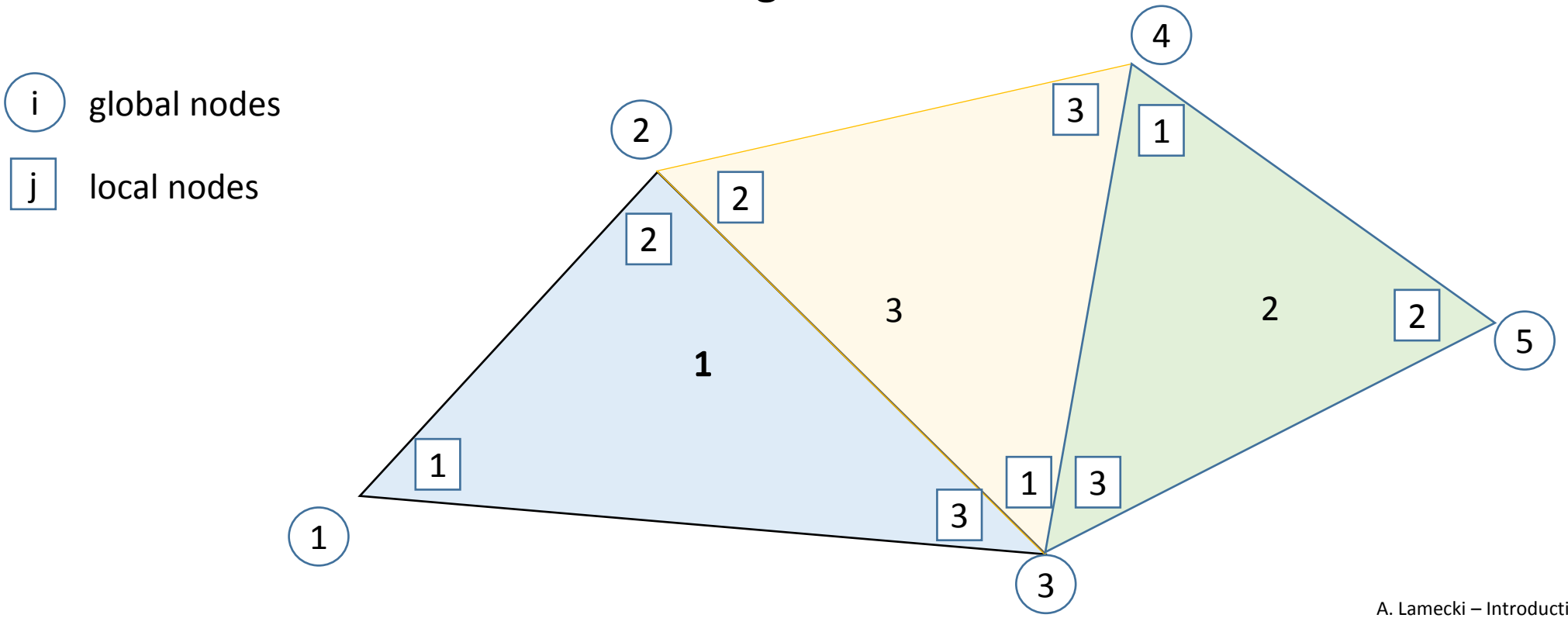
This pair of triangles does not meet the Delaunay condition (the circumcircle contains more than three points).



Flipping the common edge produces a valid Delaunay triangulation for the four points.

FEM in 2D

- For each triangle we can define two numberings:
  - Local – number of nodes 1,2,3
  - Global – number of nodes in global nodes list



For triangle 2:

Local: 1,2,3

Global: 4,5,3

FEM in 2D

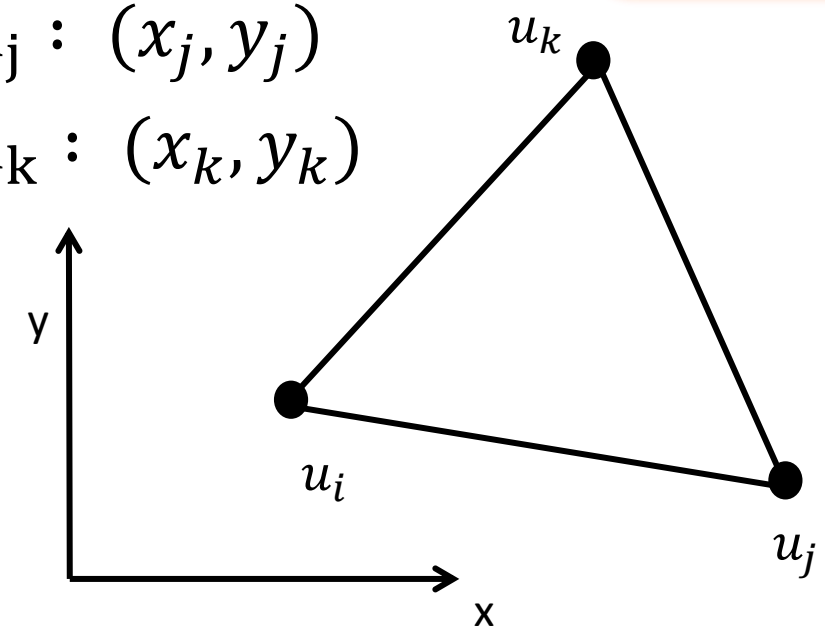
- Once we have the mesh we can generate a set of base functions on that mesh
- Let's start from single triangle and linear element:

$$\mathrm{u}(x,y) = \alpha_1 + \alpha_2 x + \alpha_3 y$$

$\mathrm{n_i}$: (x_i, y_i)

$\mathrm{n_j}$: (x_j, y_j)

$\mathrm{n_k}$: (x_k, y_k)

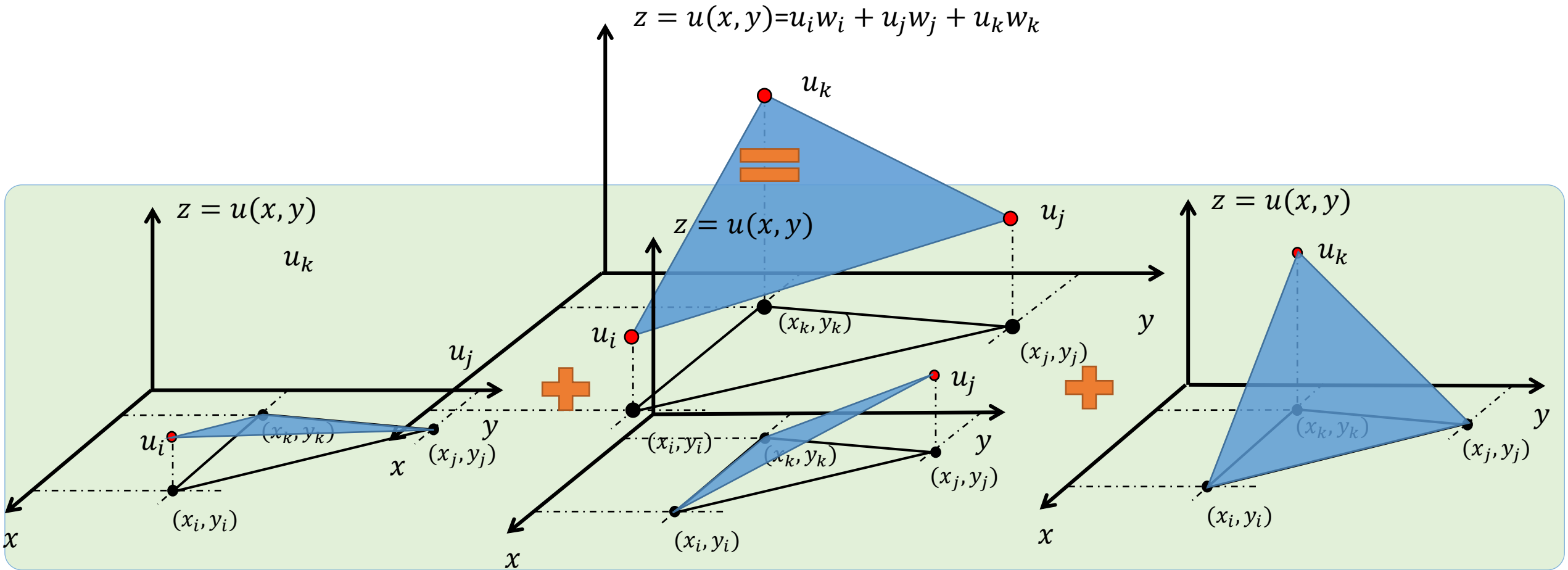


Construct a set of basis functions that is similar to 1D case:

$$u(x,y) = u_i w_i + u_j w_j + u_k w_k$$

FEM in 2D

- Piecewise linear approximation in 2D:



FEM in 2D

- The values of unknown $\alpha_1, \alpha_2, \alpha_3$ can be found for triangle solving the linear system:

$$u_i = \alpha_1 + \alpha_2 x_i + \alpha_3 y_i$$
$$u_j = \alpha_1 + \alpha_2 x_j + \alpha_3 y_j$$
$$u_k = \alpha_1 + \alpha_2 x_k + \alpha_3 y_k$$

$$\alpha_1 = \frac{1}{2A}\left[\left(x_j y_k - x_k y_j\right)u_i + \left(x_k y_i - x_i y_k\right)u_j + \left(x_i y_j - x_j y_i\right)u_k\right]$$
$$\alpha_2 = \frac{1}{2A}\left[\left(y_j - y_k\right)u_i + \left(y_k - y_i\right)u_j + \left(y_i - y_j\right)u_k\right]$$
$$\alpha_3 = \frac{1}{2A}\left[\left(x_k - x_j\right)u_i + \left(x_i - x_k\right)u_j + \left(x_j - x_i\right)u_k\right]$$

$$A = \frac{1}{2}\, det \begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{bmatrix} = \; area\ of\ the\ triangle$$

FEM in 2D

- We can introduce a matrix notation:

$$u(x,y) = u_i w_i + u_j w_j + u_k w_k = \begin{bmatrix} w_i & w_j & w_k \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \end{bmatrix} = [N][U]$$

Where:

$$w_i = \frac{1}{2A}(a_i + b_i x + c_i y)$$
$$w_j = \frac{1}{2A}(a_j + b_j x + c_j y)$$
$$w_k = \frac{1}{2A}(a_k + b_k x + c_k y)$$

Features of base functions

$$w_i(x_i, y_i) = 1; w_i(x_j, y_j) = 0; w_i(x_k, y_k) = 0$$

$$w_i + w_j + w_k = 1;$$

$$a_i = x_j y_k - x_k y_j; \quad b_i = y_j - y_k; \quad c_i = x_k - x_j$$
$$a_j = x_k y_i - x_i y_k; \quad b_j = y_k - y_i; \quad c_j = x_i - x_k$$
$$a_k = x_i y_j - x_j y_i; \quad b_k = y_i - y_j; \quad c_k = x_j - x_i$$

FEM in 2D

- We can derive a gradient of temperature:

$$\frac{\partial u}{\partial x} = \frac{\partial w_i}{\partial x} u_i + \frac{\partial w_j}{\partial x} u_j + \frac{\partial w_k}{\partial x} u_k$$
$$\frac{\partial u}{\partial y} = \frac{\partial w_i}{\partial y} u_i + \frac{\partial w_j}{\partial y} u_j + \frac{\partial w_k}{\partial y} u_k$$

- In matrix notation

$$g = \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} b_i & b_j & b_k \\ c_i & c_j & c_k \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \end{bmatrix} = [\mathrm{B}][\mathrm{U}] = \mathrm{const.}$$

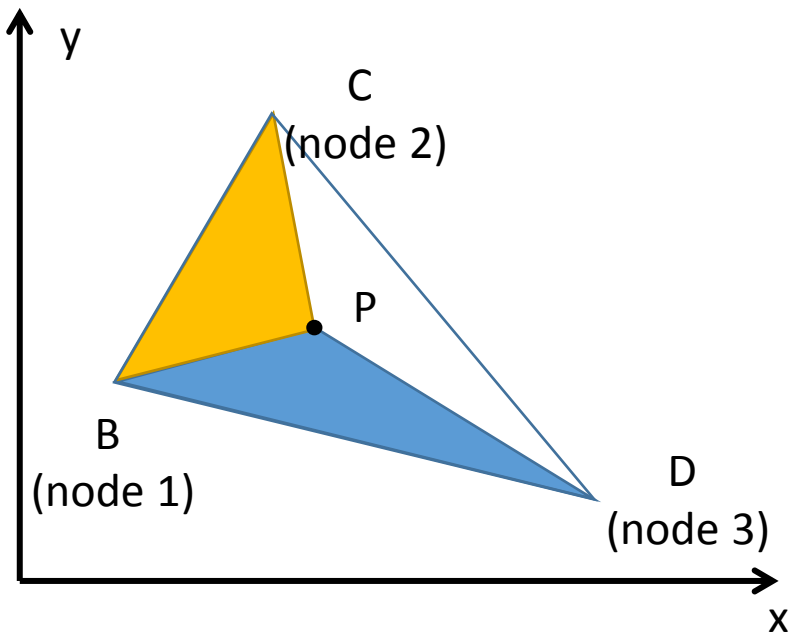
We can notice, that for linear base functions the gradient of $u(x, y)$ is constant within an element

FEM in 2D – Area coordinates

- Let's examine our weak form matrices K and M:

$$m_{ij} = \int_{\Omega} c_p \rho \, w_i w_j d\Omega$$

$$k_{ij} = \int_{\Omega} \nabla w_i \cdot \kappa \cdot \nabla w_j d\Omega$$

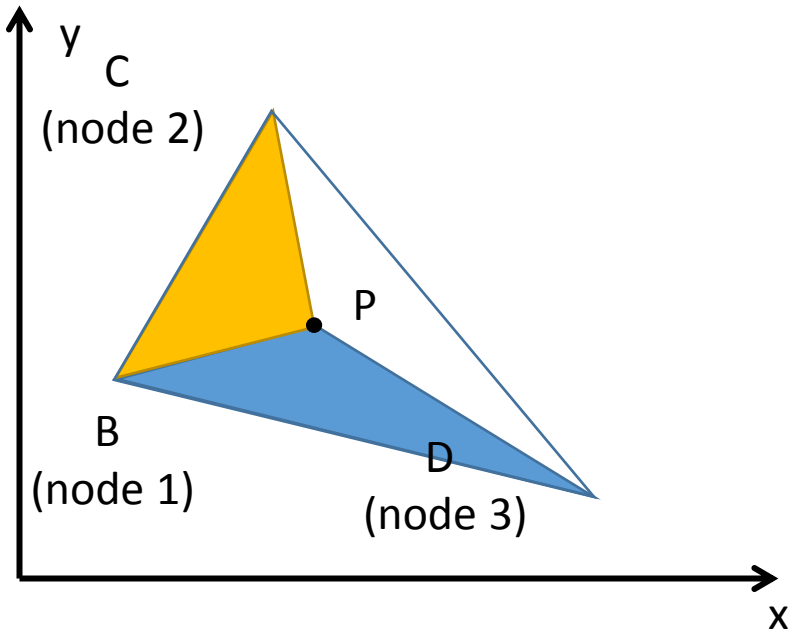


- Q: How to perform the integration over triangle?
- A: The easiest way is to use area coordinates!!
  - Triangle BCD with area equal *A*;
  - Point P → area of triangles BCP=BDP=CDP = *A*/3
  - Each coordinate L_i (i=1..3) denotes ratio of area of „small" triangle to triangle ABC
  - Point P has coordinates $L_1 = L_2 = L_3 = 1/3$
  - Midpoint of edge CD has coordinates $L_1 = L_2 = 0.5, L_3 = 0$

$$\sum_{i=1}^{3} L_i = 1$$

FEM in 2D – Area coordinates

- $L_1 = \dfrac{area(CDP)}{area(BCD)}$
- $L_2 = \dfrac{area(BDP)}{area(BCD)}$
- $L_3 = \dfrac{area(BCP)}{area(BCD)}$
- If $L_i = 0, L_{j,k} \neq 0$ – we are on the edge
- If $L_{i,j} = 0, L_k \neq 0$ – we are exactly at the node



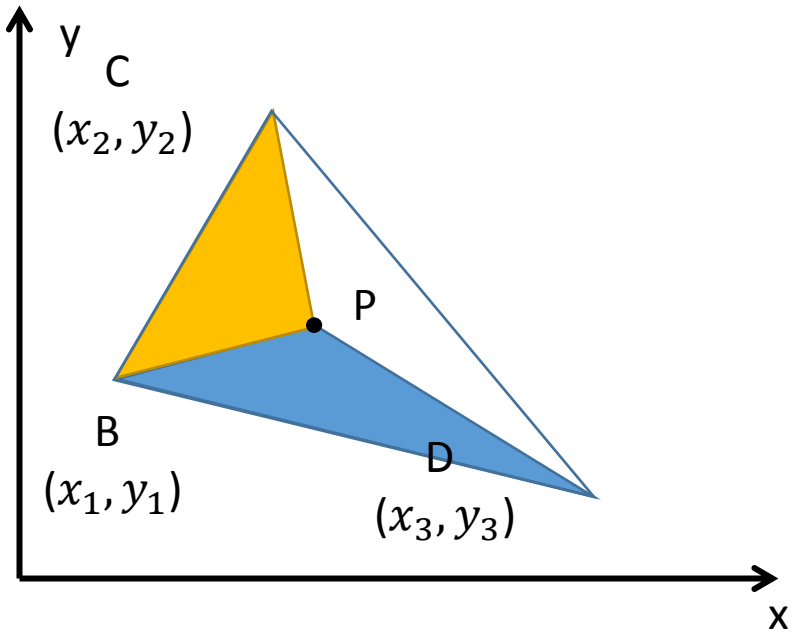
$$\sum_{i=1}^{3} L_i = 1$$

FEM in 2D – Area coordinates

$$L_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)}$$

$$L_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)}$$

$$L_3 = 1 - L_1 - L_2$$



$$\sum_{i=1}^{3} L_i = 1$$

FEM in 2D – Integration over triangle

- Integration over triangle can be efficiently organized with area (barycentric) coordinates:

$$\int_{ABC} {L_i}^a L_j^b L_k^c = 2A \frac{a!\, b!\, c!}{(a+b+c+2)!}$$

- Try to define our base functions in terms of area coordinates?
- In fact, we can notice that for linear base functions holds:

$$w_1 = L_1, \qquad w_2 = L_2, \qquad w_3 = L_3$$

FEM in 2D

- Evaluation of elemental matrices in 2D over triangle with area *A* with constant c_p, ρ, κ:

$$m_{ij} = \int_\Omega c_p \rho \, w_i w_j d\Omega = \int_\Omega c_p \rho L_i L_j d\Omega = c_p \rho \cdot A \begin{cases} \dfrac{1}{12}, & i \neq j \\ \dfrac{1}{6}, & i = j \end{cases}$$

$$k_{ij} = \int_\Omega \nabla w_i \cdot \kappa \cdot \nabla w_j d\Omega = \int_\Omega \boxed{g_i \kappa g_j} d\Omega = \kappa \cdot A \cdot g_i \, g_j$$

const.

FEM in 2D

- For more complex cases one can apply numerical integration techniques
  - Use special Gaussian quadrature formulas that allow to integrate over triangle using area (barycenrtic) coordinates

$$|T|\sum_{i=1}^{n} f(p_i)w_i = \int_T f(x)dx, \quad \forall f(x) \in \mathbb{P}^{(p)}$$

- Example:

1-point order 1 rule on triangle		
Orbit	Abscissas	Weight
S_3	.333333333333333333333333333333333	1.000000000000000000000000000000000
3-point order 2 rule on triangle		
Orbit	Abscissas	Weight
S_{21}	.166666666666666666666666666666667	.333333333333333333333333333333333
6-point order 3 rule on triangle		
Orbit	Abscissas	Weight
S_{21}	.162882850395891910900161804184906 3	.281149802440979648253514322702077 0
S_{21}	.477919883567563700000000000000000	.052183530892353685079819010631256 4
6-point order 4 rule on triangle		
Orbit	Abscissas	Weight
S_{21}	.445948490915964886318329253883051 9	.223381589678011465695007008433122 8
S_{21}	.091576213509770743459571463402201 5	.109951743655321867638326324900210 5

permutation star	barycentric coordinates	number of points
$S_3(\frac{1}{3})$	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	1
$S_{21}(a)$	$(a, a, 1-2a)$	3
$S_{111}(a, b)$	$(a, b, 1-a-b)$	6

Quadratures up to order 21 available

A Set of Symmetric Quadrature Rules on Triangles and Tetrahedra[*1)]

Journal of Computational Mathematics, Vol.26, No.3, 2008, 1–16.

Lin-bo Zhang, Tao Cui and Hui Liu

FEM in 2D

- We evaluate matrices K, M size of [3 x 3].
- Then we assembly the global matrix size of [*n* x *n*], where *n* denotes the number of nodes in mesh

→ the procedure is similar to 1D case, we add the components corresponing to the same nodes in global notation

- Need to compute the RHS:

$$\mathrm{f_j} = \int_\Omega f w_j \, d\Omega \;\rightarrow\; \text{easy if 'f' is represented as } \mathrm{f} = \sum_{i=1}^{n} e_i w_i$$

$$f_j = \int_\Omega \left(\sum_{i=1}^{n} e_i w_i \right) w_j d\Omega = e_i \int_\Omega w_i w_j d\Omega$$

- Then we enforce the boundary conditions → Naumann or Dirichlet

FEM in 2D

- What if f is not given as $\mathrm{f} = \sum_{i=1}^{n} e_i w_i$?

- We can to find the best approximation of f using our basis w_i:

$$f = \sum_{i=1}^{n} e_i w_i \qquad \rightarrow on\ both\ sides\text{:}\ \int_{\Omega} (\cdot) \cdot w_j d\Omega$$

$$\int_{\Omega} f w_j d\Omega = \int_{\Omega} e_i w_i w_j d\Omega\,, \qquad \text{let: } m^0_{(i,j)} = \int_{\Omega} w_j w_j d\Omega$$

$$F = M^0 E \qquad \rightarrow \qquad \boxed{E = (M^0)^{-1} F}$$

FEM in 2D

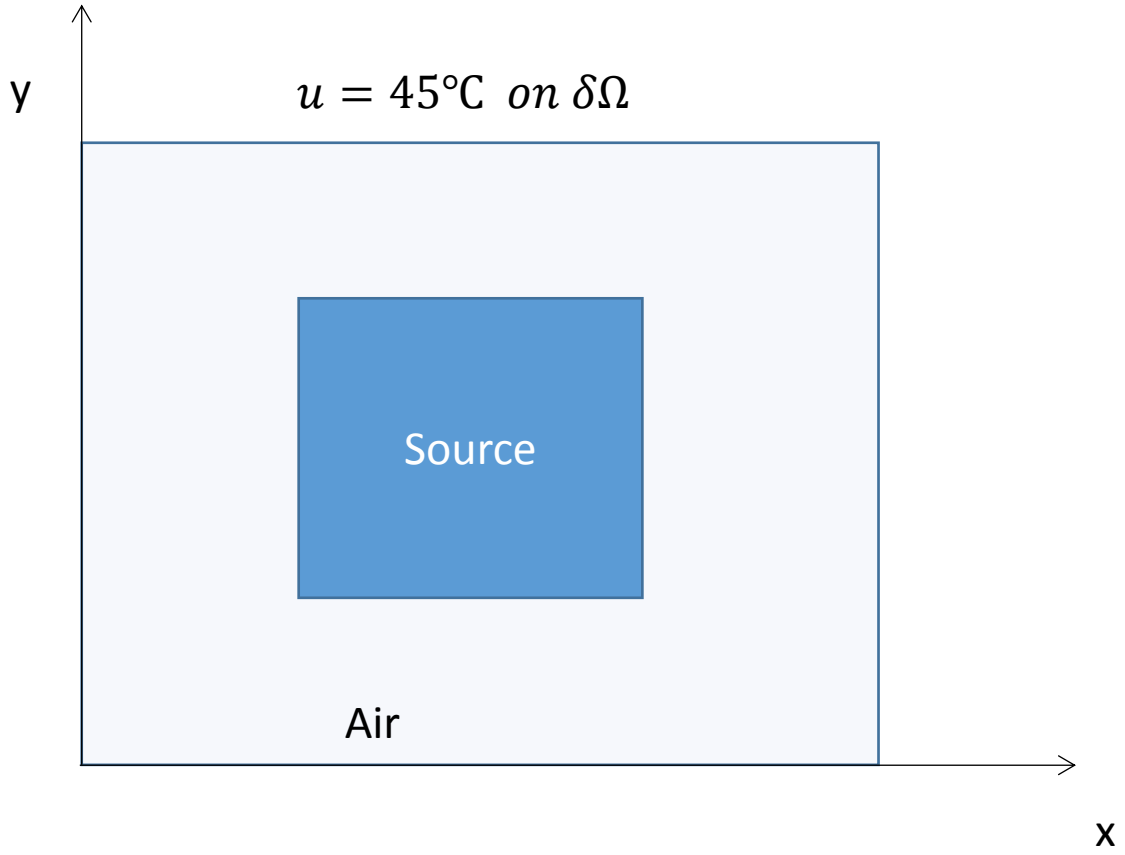
- Finally, solve the system for unknown vector U:

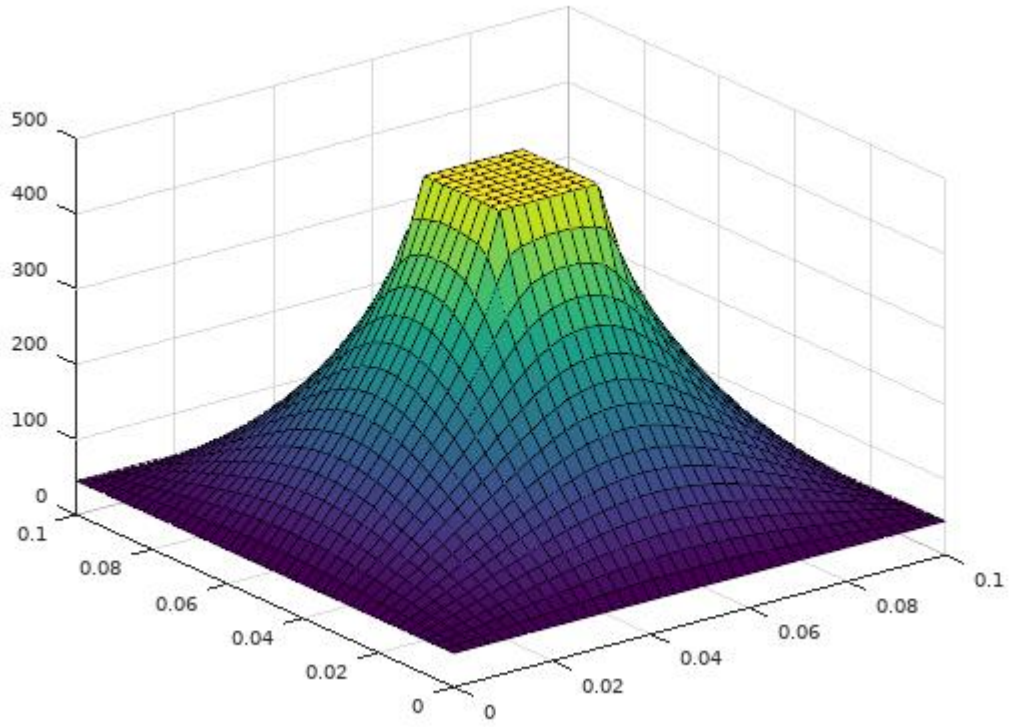
$$[M]\frac{\partial}{\partial t}U + K \cdot U = F$$

- U is a vector of amplitudes of base functions – corresponds to values of temperature at the mesh nodes
- Q : How can I compute temperature at arbitrary point inside the domain?
  - → for given point (x,y) find triangle element and evaluate the temperature
  - → need to have a fast algorithm to locate a proper triangle

2D heat example

- Basic CPU thermal simulation





Source size: 20x20 [mm] → Surface = $0.004m^2$
TDP: 90W
→ incoming heat flux: 225000 W/m2 (source)

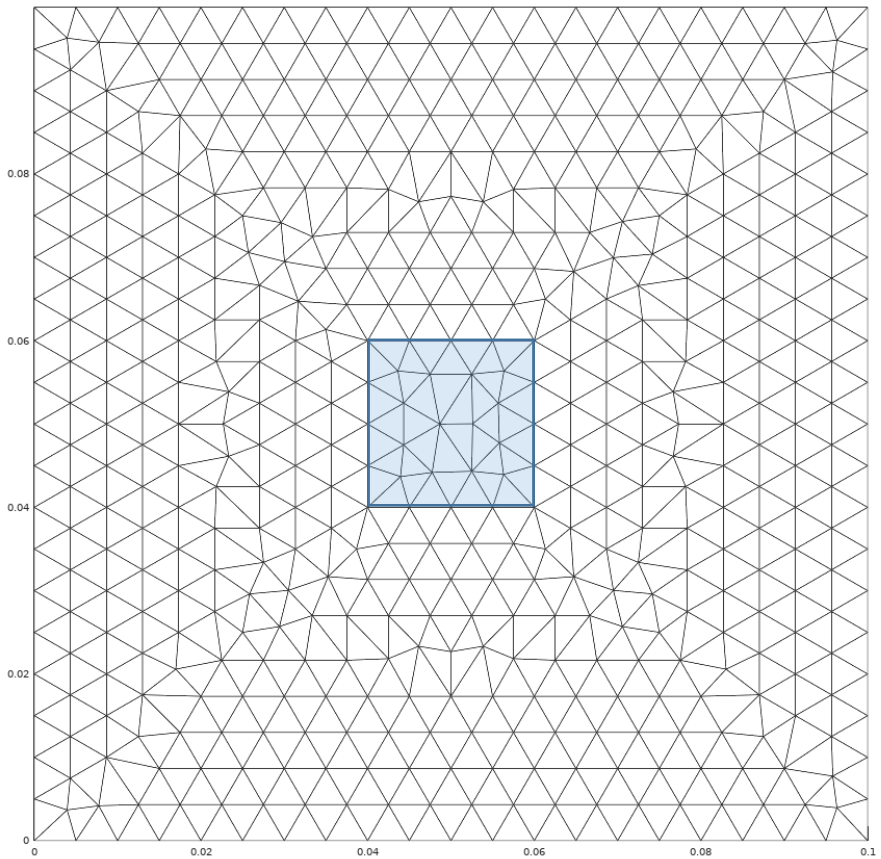
Thermal conductivity of Aluminium: $\kappa = 54$ [W/(m*K)
Air area: size of 100x100 [mm]

Heat convection in air area: 50 [W/m*K] (energy loss)
Thermal conductivity of air: $\kappa = 0.05$ [W/(m*K)

2D heat example

- First we generate a mesh with mesh2D:

```
% nodes array in (x,y) plane:
nodes = [ 0, 0                  % outer square
          0.1, 0
          0.1, 0.1
          0, 0.1
          0.04, 0.04
          0.06, 0.04
          0.06, 0.06
          0.04, 0.06 ];


% array of edges - pairs of nodes
edges = [1 2;
         2 3;
         3 4;
         4 1;
         5 6
         6 7
         7 8
         8 5];

parts{1} = [1 2 3 4];
parts{2} = [5 6 7 8];

h = 0.005; % maximum mesh edge length

[vert,etri, tria,tnum] = refine2(nodes,edges,parts,[],h) ;
```



- 938 triangles in total
- 42 triangles on heat source
- 510 nodes

2D heat example

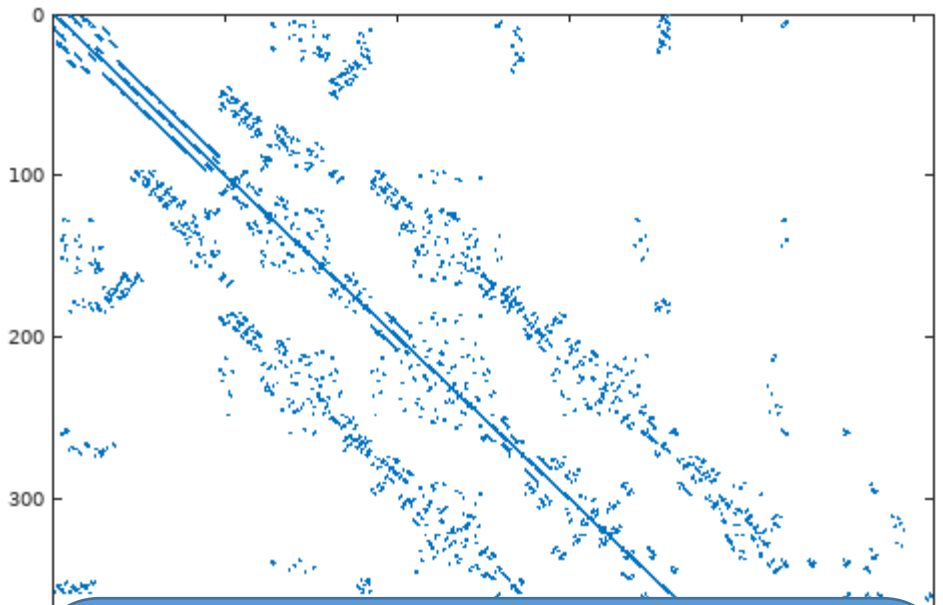
- Next we need to generate local matrices K_e, M_e using formulas given previousely for 2D elements, for example for p-th element:

```
triangle_nodes = tria(p,:);

x = vert(triangle_nodes,1);
y = vert(triangle_nodes,2);

J = [ones(3,1) , x , y];

A = 0.5*det(J);

Q = ones(3,3)/12;
Q(1:4:end) = 1/6;

Me = cp*rho*A*Q;


b1 = y(2) - y(3);
b2 = y(3) - y(1);
b3 = y(1) - y(2);

c1 = x(3) - x(2);
c2 = x(1) - x(3);
c3 = x(2) - x(1);

G = (0.5/A)*[b1 b2 b3; c1 c2 c3];

Ke = A*kk*(G.'*G);
```

2D heat example

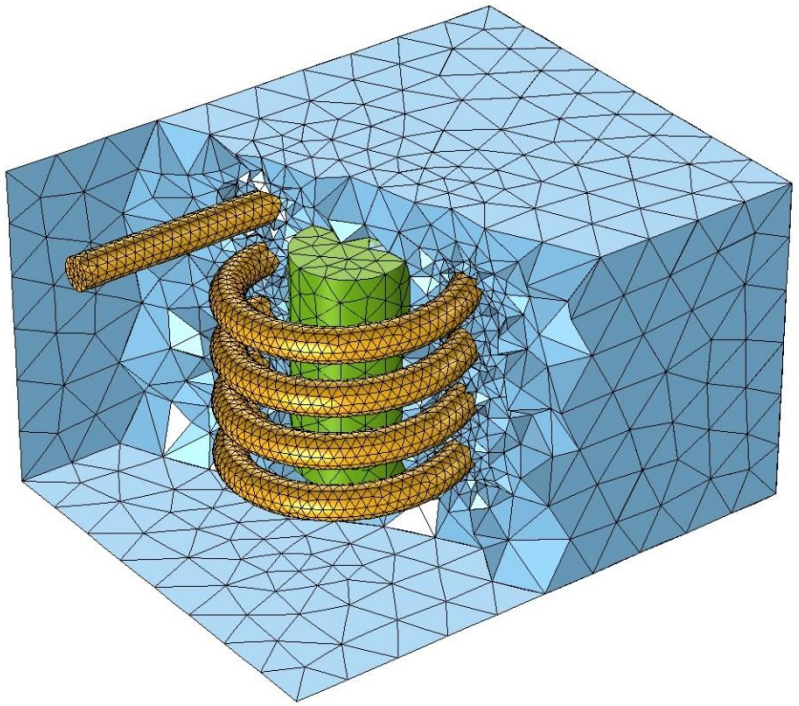
- Assembly the global matrices K,M:

```
cp = 1;
rho = 1;

Ne = size(tria,1);
Nn = size(vert,1);


K = sparse(Nn,Nn);
M = sparse(Nn,Nn);
F = zeros(Nn,1);
for p=1:Ne

   if (tnum(p)==1)
     kk = 0.05;
   else
     kk = 54;
   end;

   [ Ke,Me] = get2Delement(p,kk,cp, rho, vert, tria,tnum);

   idx = tria(p,:);

   K(idx,idx) = K(idx,idx)+Ke;
   M(idx,idx) = K(idx,idx)+Me;


end
```



…Then provide boundary conditions (e.g. Dirichlet), optionally define the source/sink and solve the linear problem…

Heat equation in 3D



- In 3D the procedure is straightforward, we just:
  - Operate on 3D tetrahedral mesh
  - Define the linear base functions in 3D
  - Compute local & global K,M matrices
  - Solve the linear problem for unknown nodes

How about solid mechanics?

Advanced solid mechanics?

More than 1mln downloads (Google store)
Estimated revenue ?

FEM for 3D Solid Mechanics

- Imagine we have a simple spring and apply a force *F,* then we get a Hooke's law:

$$F = k \cdot \Delta x$$

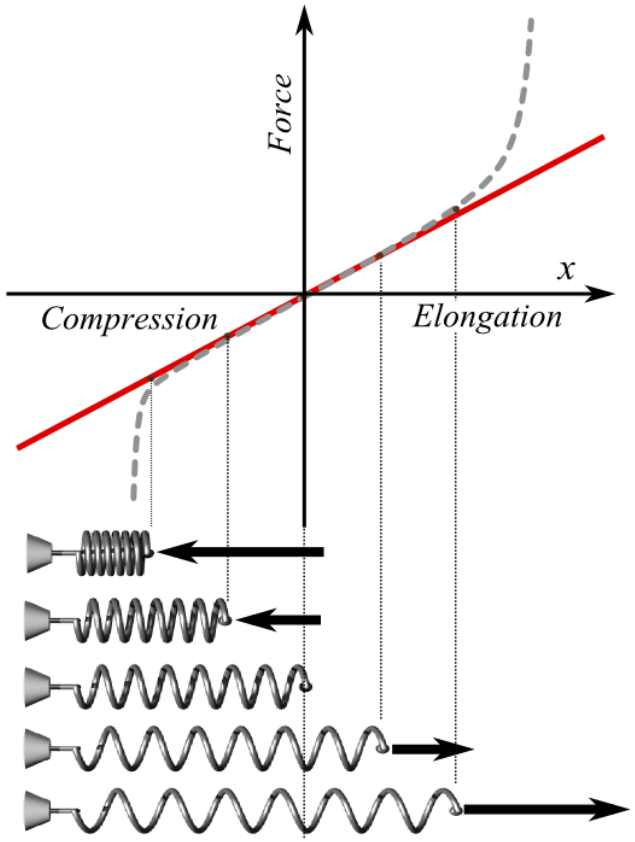
- If we replace the spring with continous media, then we have a ralation between the stress and strain

$$\sigma = E \cdot \epsilon$$

where (in general):

  - σ is a stress tensor
  - E is a Young modulus of elasticity
  - ϵ is a strain tensor

- This relation is valid at any point (x,y,z) of a continous volume



Basic equations of solid mechanics

- For **isotropic media** we have 6 independent components in both ϵ and σ, we can write a „special" form of Hooke's law (linear elasticity model):

$$\sigma_m = c \cdot \epsilon_m$$

$$\sigma_m{}^T = \begin{bmatrix} \sigma_{xx} & \sigma_{yy} & \sigma_{zz} & \sigma_{yz} & \sigma_{xz} & \sigma_{xy} \end{bmatrix}$$

$$\epsilon_m{}^T = \begin{bmatrix} \epsilon_{xx} & \epsilon_{yy} & \epsilon_{zz} & \epsilon_{yz} & \epsilon_{xz} & \epsilon_{xy} \end{bmatrix}$$

where μ_m and λ are so called Lame coefficient's, v is so-called Poisson ratio:

$$\mu_m = \frac{E_m}{2(1+v)}$$

$$\lambda = \frac{Ev}{(1+v)(1-2v)}$$

$$c = \begin{bmatrix} \lambda + 2\mu_m & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda + 2\mu_m & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda + 2\mu_m & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu_m & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu_m & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu_m \end{bmatrix}$$

Basic equations of solid mechanics

The strain is related to the displacement vector u as:

$$\epsilon_m = L \cdot u$$

where L is a differential operator:

$$L = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \end{bmatrix}$$

Finally, the dynamic equilibrium equation of the mechanical problem can be given in a matrix form:

$$L^T \sigma_m + f_b = \rho \cdot \ddot{u}$$

In this equation, ρ represents the density of the material and f_b is a vector of external forces applied to the solid. In the static case, the right-hand side vanishes.

This is equation to solve with FEM (!)

Basic equations of solid mechanics

- Then we introduce a matrix of basis functions:

$$W = \begin{bmatrix} w_1 & 0 & 0 & w_2 & 0 & 0 & \\ 0 & w_1 & 0 & 0 & w_2 & 0 & \cdots \\ 0 & 0 & w_1 & 0 & 0 & w_2 & \end{bmatrix}$$

For each node we have 3 base functions to model (x,y,z) displacements

and **node displacement** vector d (size of 3* number of nodes!!):

$$d = [\underbrace{x_1 \quad y_1 \quad z_1}_{\text{node 1}} \quad \underbrace{x_2 \quad y_2 \quad z_2}_{\text{node 2}} \ldots.]^{\mathrm{T}}$$

- The displacement u inside the element can be computed as $u = W \cdot d$
- With this we get $\epsilon_m = L \cdot W \cdot d$
- We can define a strain martix $B = L \cdot W$

Discrete weak form:

- If we proceed with FEM procedure, derive the weak form of this equation and apply Galerkin procedure, we get a matrix equation:

$$M\frac{\partial^2}{\partial t^2}\{d\} + K \cdot d = F$$

$$\frac{\partial^2}{\partial t^2}\{d\} \rightarrow \text{Acceleration}$$

where:

$$M = \int_\Omega W^T \rho W d\Omega$$

$$K = \int_\Omega B^T c\, B d\Omega$$

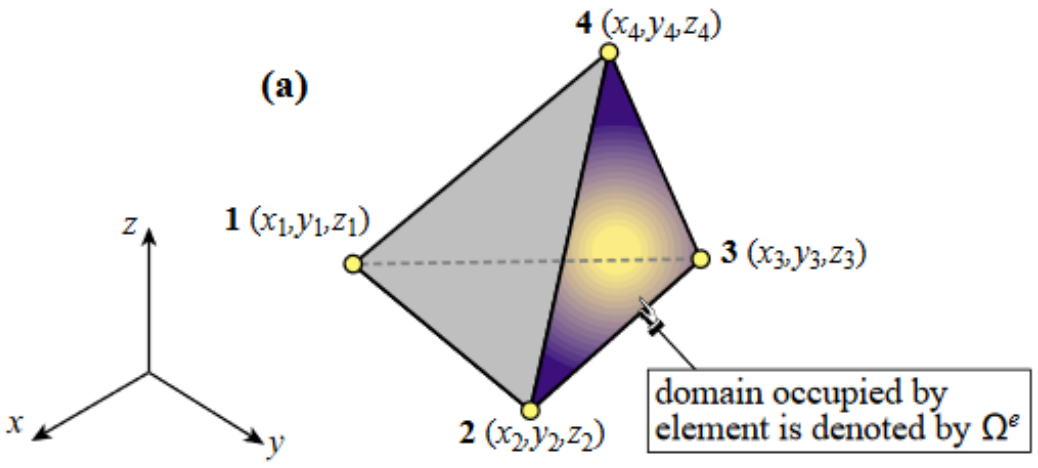
$$F = \int_\Omega W^T f_v + \int_{\partial\Omega} W^T f_s$$

K – stiffness matrix, M – mass matrix

Vector of internal/external forces applied to our solid

Linear basis function in 3D

- Start with the 3D tetrahedral mesh – list of:
  - Nodes, (matrix of *x,y,z* coordinates)
  - Edges (each of 2 nodes)
  - Elements (each of 4 nodes)
- We can generalize formerly defined 2D base functions to 3D case:



- We use a barycentric coordinates (L_1, L_2, L_3, L_4) defined over tetrahedra
- $\sum_{i=1}^{4} L_i = 1$

$$L_1 = \frac{V_{P234}}{V_{1234}} \quad L_2 = \frac{V_{P134}}{V_{1234}}, \quad L_3 = \frac{V_{P124}}{V_{1234}}, \quad L_4 = \frac{V_{P123}}{V_{1234}}$$

- $w_1 = L_1, w_2 = L_2, w_3 = L_3, w_4 = L_4$

```
function K = get3DMassStiffness(x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4, Em, miu)

% returns a 3D tetrahedron (CST) element K & M matrix
% in global coordinates
% INPUT:
%       x1, y1, z1 - coordinates of joint 1 of the triangle element
%       x2, y2, z2 - coordinates of joint 2 of the triangle element
%       x3, y3, z3 - coordinates of joint 3 of the triangle element
%       x4, y4, z4 - coordinates of joint 3 of the triangle element
%       Em     - elastic modulus for isotropic material
%       miu    - Poisson's ratio
% ---------------------------------------------------------------------------

J = [1 1 1 1; x1 x2 x3 x4; y1 y2 y3 y4; z1 z2 z3 z4];
detJ = det(J);
V = 1/6 * detJ;
Q = inv(J)*detJ;

a1 = Q(1,2); b1 = Q(1,3); c1 = Q(1,4);
a2 = Q(2,2); b2 = Q(2,3); c2 = Q(2,4);
a3 = Q(3,2); b3 = Q(3,3); c3 = Q(3,4);
a4 = Q(4,2); b4 = Q(4,3); c4 = Q(4,4);

s=(1-miu);
E  = Em*(1-miu)/((1+miu)*(1-2*miu)).*...
         [   1    miu/s  miu/s        0                  0                  0;
           miu/s    1    miu/s        0                  0                  0;
           miu/s miu/s     1          0                  0                  0;
             0      0      0    (1-2*miu)/(2*s)          0                  0;
             0      0      0          0          (1-2*miu)/(2*s)            0;
             0      0      0          0                  0          (1-2*miu)/(2*s)];

B = 1/(6*V)*[ a1    0    0   a2    0    0    a3    0    0    a4    0    0;
               0   b1    0    0   b2    0    0    b3    0    0    b4    0;
               0    0   c1    0    0   c2    0    0    c3    0    0   c4;
               0   c1   b1    0   c2   b2    0    c3   b3    0   c4   b4;
              c1    0   a1   c2    0   a2   c3    0    a3   c4    0   a4;
              b1   a1    0   b2   a2    0   b3    a3    0   b4    a4    0];

K = V.*B'*E*B ;
```

Example matlab code for computation of local K,M matrices

```
M  =   V/20*  [2 0 0 1 0 0 1 0 0 1 0 0;
               0 2 0 0 1 0 0 1 0 0 1 0;
               0 0 2 0 0 1 0 0 1 0 0 1;
               1 0 0 2 0 0 1 0 0 1 0 0;
               0 1 0 0 2 0 0 1 0 0 1 0;
               0 0 1 0 0 2 0 0 1 0 0 1;
               1 0 0 1 0 0 2 0 0 1 0 0;
               0 1 0 0 1 0 0 2 0 0 1 0;
               0 0 1 0 0 1 0 0 2 0 0 1;
               1 0 0 1 0 0 1 0 0 2 0 0;
               0 1 0 0 1 0 0 1 0 0 2 0;
               0 0 1 0 0 1 0 0 1 0 0 2];
```

Next steps

- We assembly global matrices K,M using local-global node numbering scheme
- For each element compute the RHS component according to force applied to the solid and assembly a global vector *F*
- We can apply Dirichlet boundary conditions (for example freezing some nodes!)
- In steady state solve the matrix equation :

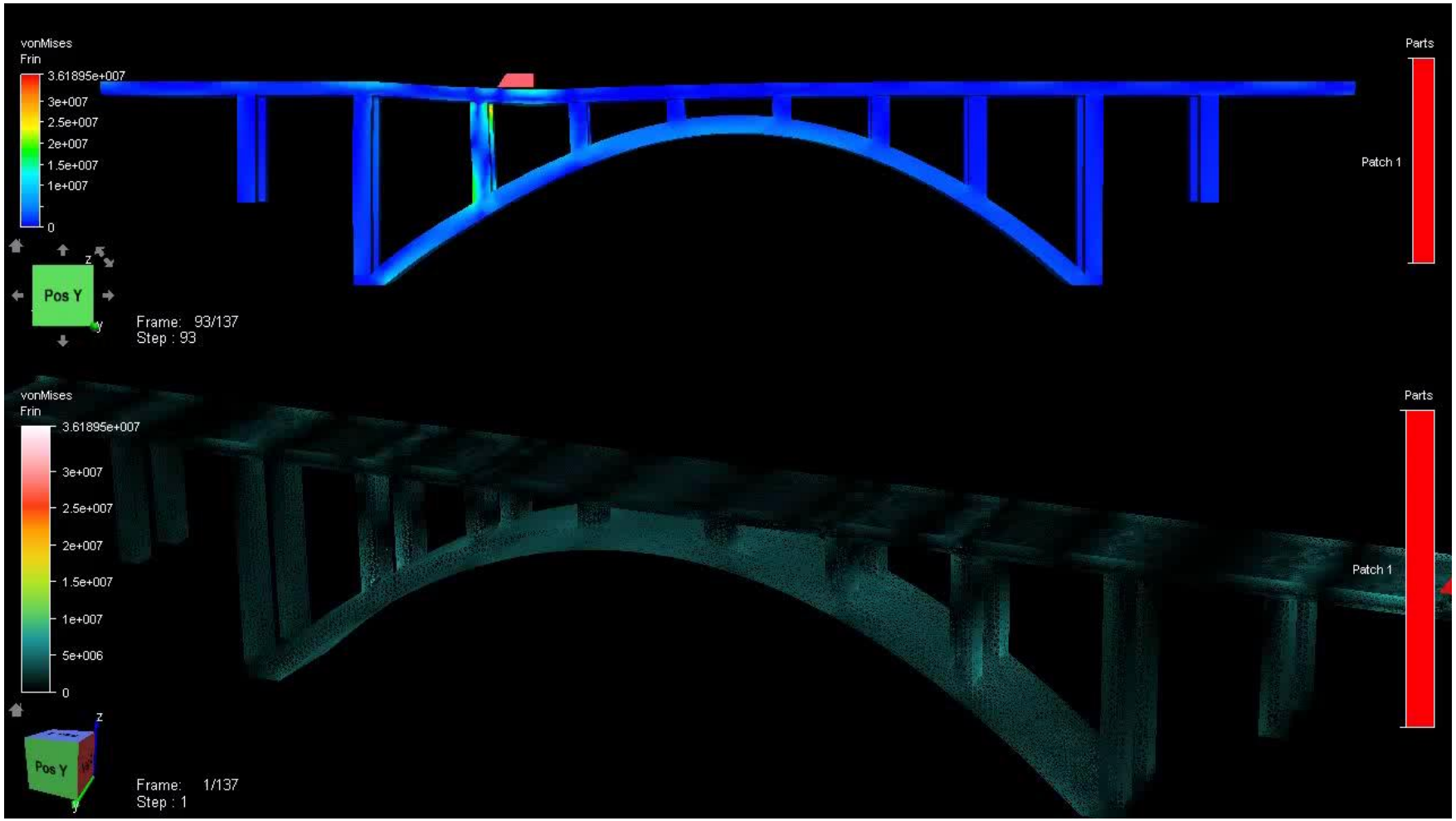
$$K \cdot d = F$$

- In transient simulation apply the time stepping to equation

$$M\ddot{d} + K \cdot d = F$$

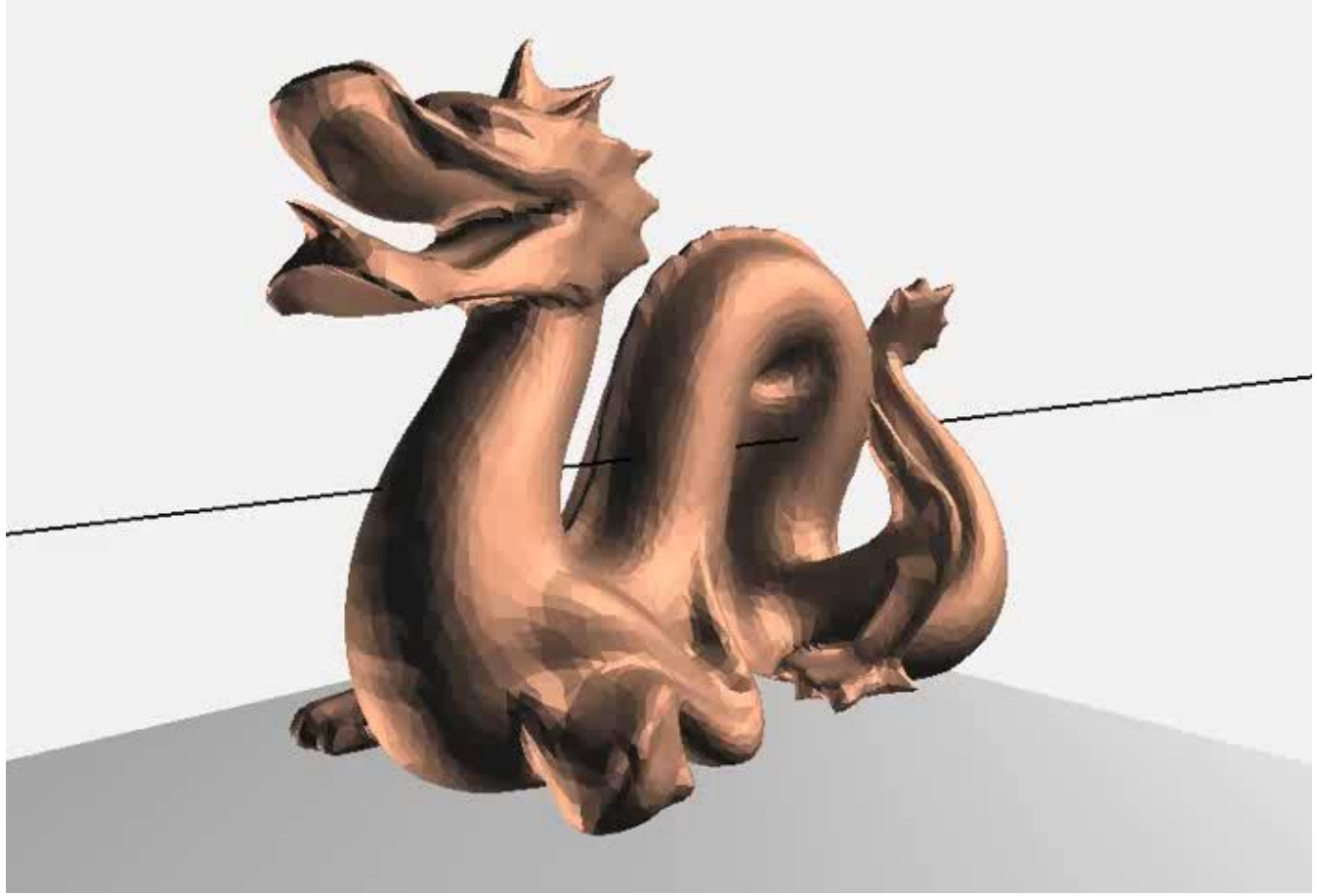
using central differences (e.g. *Newmark Method*)

Example 1.



https://www.youtube.com/watch?v=IgfWjCvCSuk

Example 2.



https://www.youtube.com/watch?v=AGsA5DDnzuQ

Example 3.

If one puts some more effort then includes:

- Non-linear effects
- Mesh-tearing
- …

Houdini software: https://www.youtube.com/watch?v=rrllYE5w9mw

That’s all about FEM on this course!

Modeling and simulation of systems

Lecture 12: How to make your simulations efficient?

Efficiency in numerical analysis

- Our goal: reduce the time of computations
- Often one have to choose:

  Effciency or productivty – need to find balance!

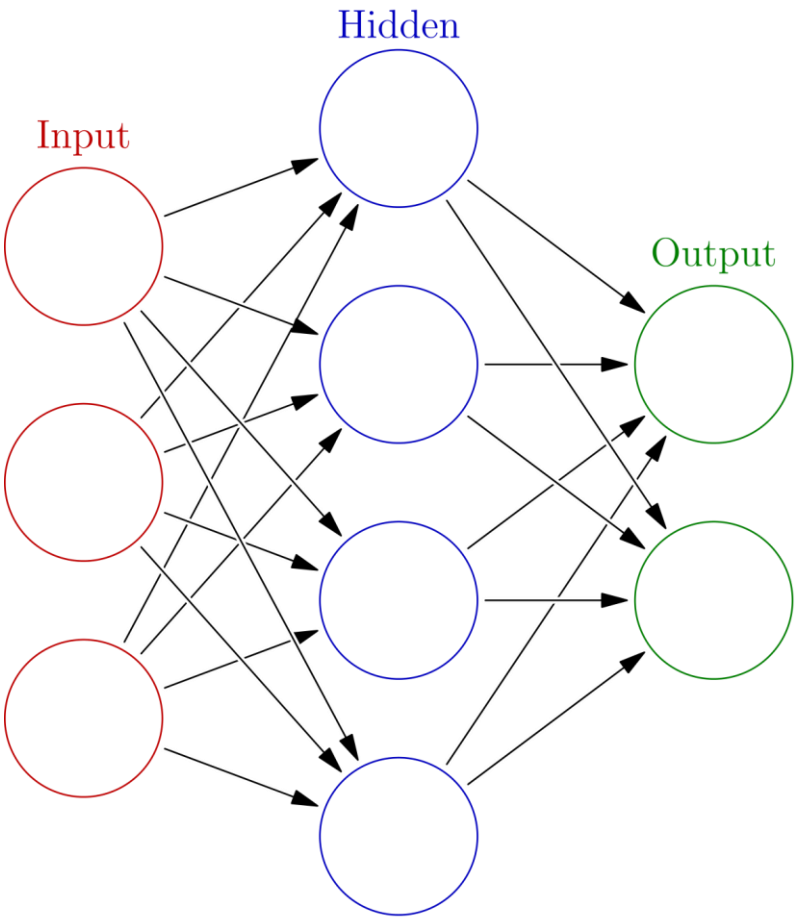
- What influences effciency?
  - Algorithm selection
  - Algorithm implementation
  - Hardware platform

Basic matrix operations

- Many simulation techniques, like finite differences (FD) of finite element method (FEM), involves matrix operations:
  - Matrix-vector product $y = A \cdot x + z$
  - Matrix-matrix product $C = A \cdot B + D$
  - Solution of linear problem $A \cdot X = B$

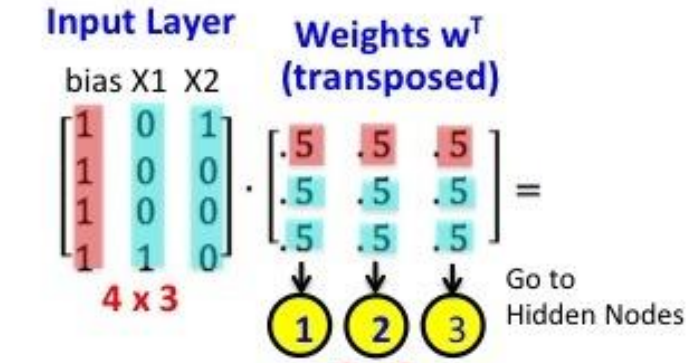
- It might be convinent to express other numerical algorithms using matrix operations…

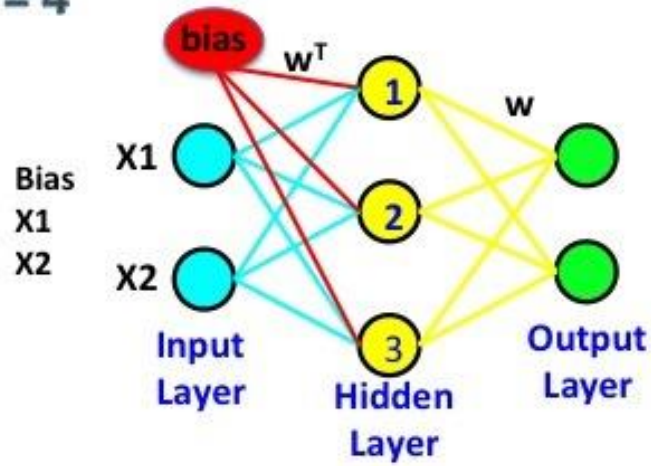
Single layer ANN

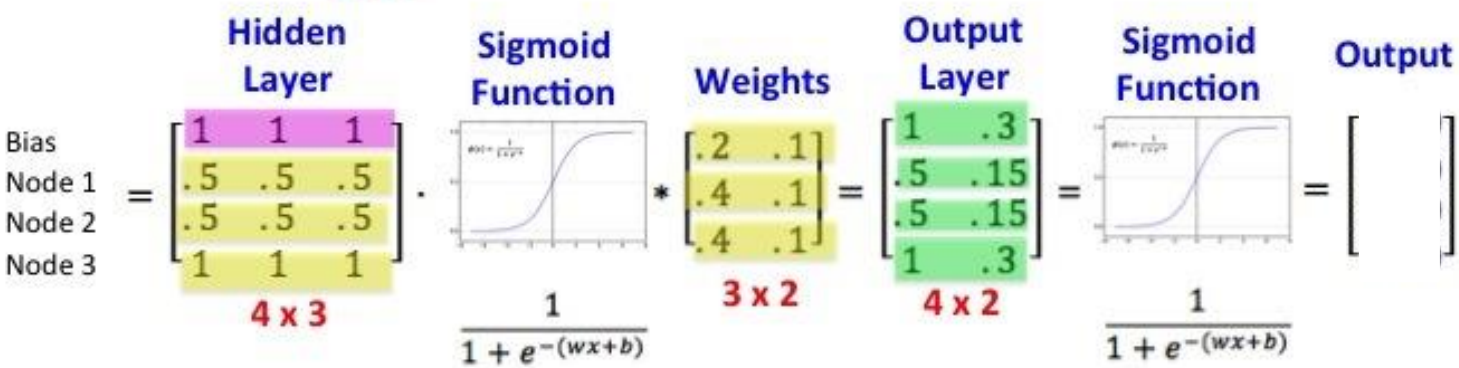


Neural Networks

Color Guided Matrix Multiplication for a Binary Classification Task with N = 4





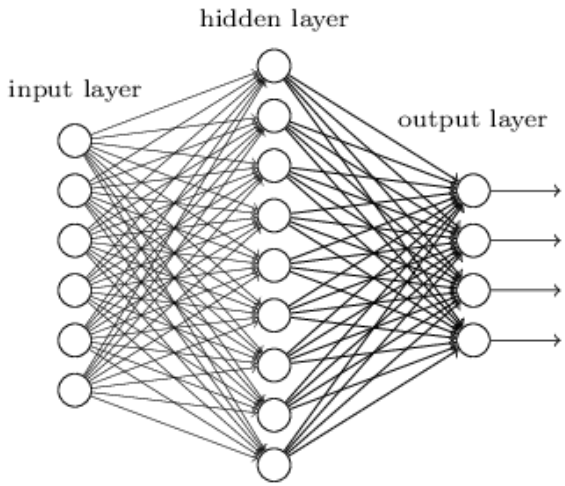


Rubens Zimbres

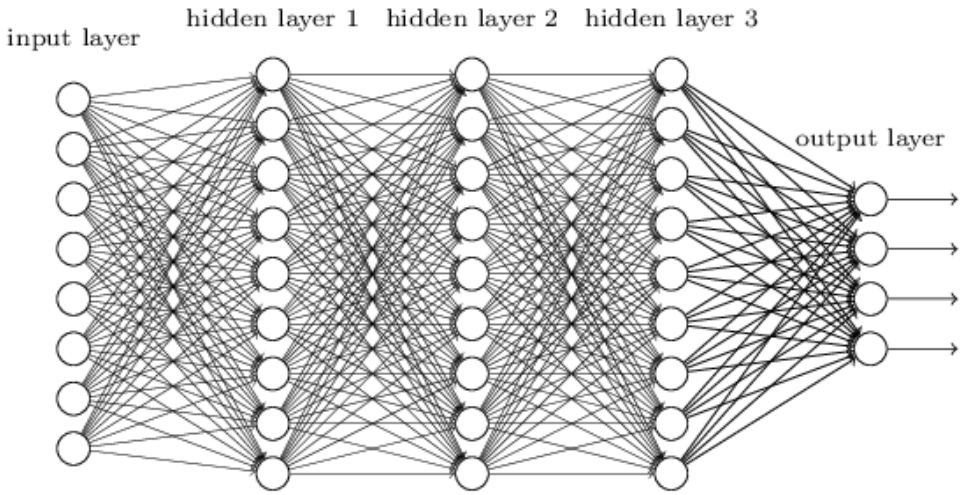
https://www.datasciencecentral.com/profiles/blogs/matrix-multiplication-in-neural-networks

DNN

"Non-deep" feedforward neural network



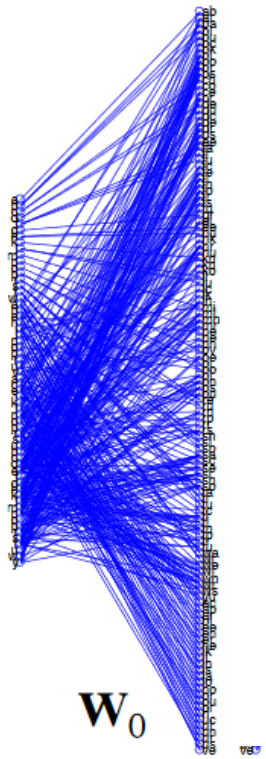
Deep neural network

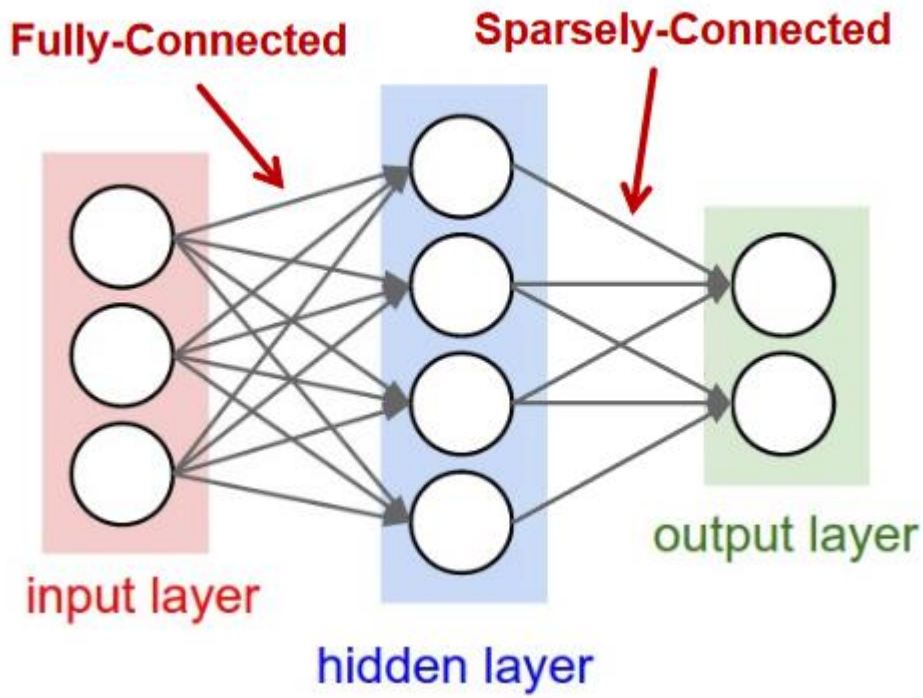


This and many other problems can be expressed as matrix operations!

Sparse DNN

- Human brain has approximately 86 billion neurons and 150 trillion synapses. Its graph representation would have approximately 2,000 edges per node, or a sparsity of 0.000002% (*)





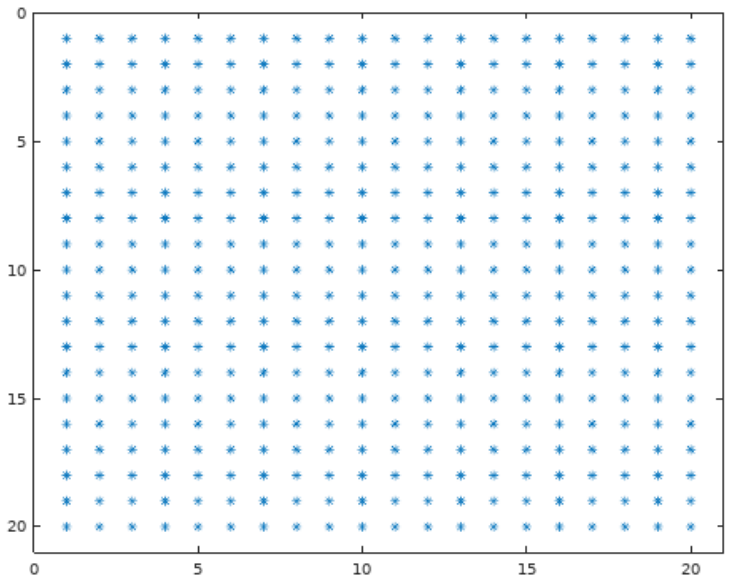
/ork Exact Solutions

Jeremy Kepner[1], Vijay Gadepally[1], Hayden Jananthan[1,2], Lauren Milechin[1], Sid Samsi[1]
[1]Massachusetts Institute of Technology, [3]Vanderbilt University

Fig. 8. One of many possible three layer exact DNN solutions over categories of popular four letter words.

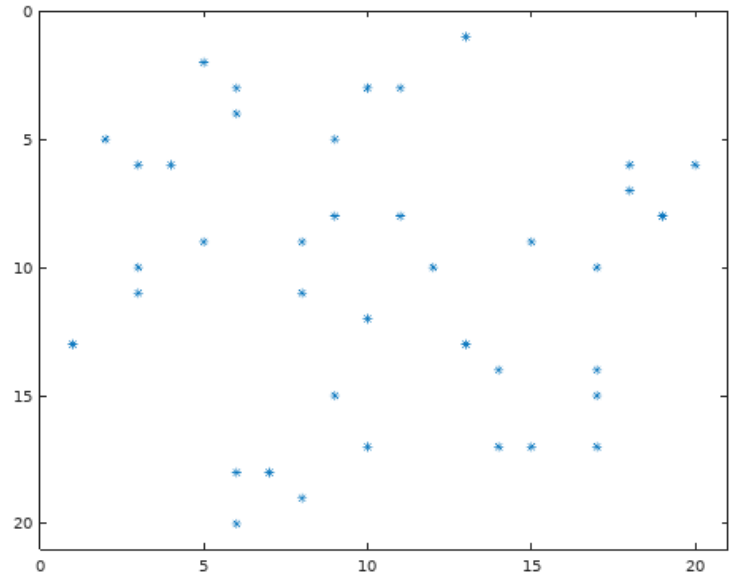
Dense vs. Sparse matrices

- spy(A) – only non zero elements are shown:



A is dense

Most of elements are non-zero



A is sparse

Low number of non-zero elements

Dense $y = A \cdot x$

- One can write a simple C++ code to perform operation $y = A \cdot x$ for dense square matrix:

```
for (int i = 0; i<size; i++)
{
        y[i] = 0;
        for (int j = 0; j<size; j++)
                y[i] += A[i*size+j] * x[j];
}
```

- Size = 20000
  - optimization disabled, CPU time = 1.45s (VS C++)
  - optimization /O2, CPU time = 0.596s (VS C++ , vectorized loopes)
  - In Matlab "y=A*b" → CPU time: 0.148s (!?)

BLAS

- **Basic Linear Algebra Subprograms (BLAS)** is a **specification** that prescribes a set of low-level routines for performing common linear algebra operations such as vector addition, scalar multiplication, dot products, matrix-vector and matrix-matrix multiplication (and many more…)

- Originated as **Fortran** procedures in 1979
- Specification handles single/double precision, real/complex valued:
  - S : single real C – single complex
  - D: double real Z – double complex

- **Architecture dependent implementations** – highly optimized for speed for **large** datasets, using multithreading/MPI for parallel execution

BLAS

- BLAS:
  - Level 1:
    - dot products, vector norms, a generalized vector additions
  - Level 2:
    - matrix-vector operations
  - Level 3:
    - matrix-matrix operations, including a "general matrix multiplication"

BLAS

Example: BLAS Level 2 matrix-vector multiplication functions:

SGEMV and **DGEMV** compute the matrix-vector product for either a real general matrix or its transpose, using the scalars α and β, vectors x and y, and matrix A or its transpose (^T):

$y \leftarrow \beta y + \alpha Ax$

$y \leftarrow \beta y + \alpha A^T x$

CGEMV and **ZGEMV** compute the matrix-vector product for either a complex general matrix, its transpose, or its conjugate transpose, using the scalars α and β, vectors x and y, and matrix A, its transpose (^T), or its conjugate transpose (^H):

$y \leftarrow \beta y + \alpha Ax$

$y \leftarrow \beta y + \alpha A^T x$

$y \leftarrow \beta y + \alpha A^H x$

BLAS

- Example call (C++) to execute $y = A \cdot x$

```
void matvecprod(double* A, double* v, double* u, int N)
{
    double alpha= 1.0, beta= 0.0;
    char no= 'N', tr= 'T';
    int m= N, n= N, lda= N, incx= 1, incy= 1;
    double* tmp= new double[N];
    initvec(tmp, N);
    dgemv_(&no,&m,&n,&alpha,A,&lda,v,&incx,&beta,tmp,&incy);
    for(int i= 0; i<N; ++i) { u[i]= tmp[i]; }
    delete [] tmp;
}
```

BLAS

- SGEMV,DGEMV,CGEMV, ZGEMV – work for **general rectangular** matrices

BUT we have also:

- SSYMV,DSYMV,CSYMV, ZSYMV – for **symmetric matrices**

- SSYMV,DSYMV,CSYMV, ZSYMV – for **general banded** matrices

SparseBLAS

- Use SparseBLAS for sparse matrices
- Only a small subset of matrix operations is specified with SparseBLAS:
  - Matrix vector product
  - Dot product
  - Triangular solve
- Direct solvers for linear problems not available!

LAPACK

Linear Algebra Package (LAPACK) is a standard software library for numerical linear algebra (**dense**). Provides routines for:

- solving systems of linear equations and linear least squares
- dense eigenvalue problems
- singular value decomposition (SVD)
- matrix factorizations, such as:
  - QR (general rectangular matrices, used for least-squares solutions!)
  - LU (general square matrices)
  - LL^T (Cholesky, symmetric positive definite matrices)

- For example Matlab uses LAPACK routines extensively!

QR, LU, LL^T

- Matrix factorizations are essential for solutions of linear system of equations:

$$A \cdot x = b$$

- If only we compute factorization of A as:

$$A = L \cdot U$$

where L is lower triangular, U is upper triangular, then:

$$L \cdot U \cdot x = b \quad \rightarrow \quad x = (L \cdot U)^{-1} \cdot b = U^{-1} \cdot L^{-1} \cdot b$$

- We solve the system in two steps:

$$L \cdot y = b$$
$$U \cdot x = y$$

Need to solve two triangular systems!!!

NO MATRIX INVERSION!!

Triangular solve

- Once factorization is available the solution of triangular system is easy….

$$
\begin{cases}
y_1 & = & b_1 \\
\ell_{2,1} y_1 + y_2 & = & b_2 \\
\ell_{3,1} y_1 + \ell_{3,2} y_2 + y_3 & = & b_3 \\
 & \vdots & \\
\ell_{n,1} y_1 + \ell_{n,2} y_2 + \ell_{n,3} y_3 + \cdots + \ell_{n,n-1} y_{n-1} + y_n & = & b_n
\end{cases}
$$

- BUT: In practice such process is quite hard to parallelize

QR, LU, LL^T

- Why one would use matrix factorization instead of Gaussian-elimination to solve $A \cdot x = b$?

- Example: 1D FEM heat transfer in time-domain:

```
for p=1:Q

    U1 = (M+dt*K)\(M*U0 + dt*L); % apply Euler backward metho
    U0 = U1;

    plot(x,[0; U1; 0] ,'b-o');
    ylim([0 0.1]);
    str = ['dt =' num2str(dt) ', time = ' num2str(p*dt)]
    title(str);
   currFrame = getframe(gcf);
   writeVideo(vidObj,currFrame);

end
```

It would be more efficient to compute LU factorization outside loop: $[L, U] = lu(M + dt \cdot K)$;

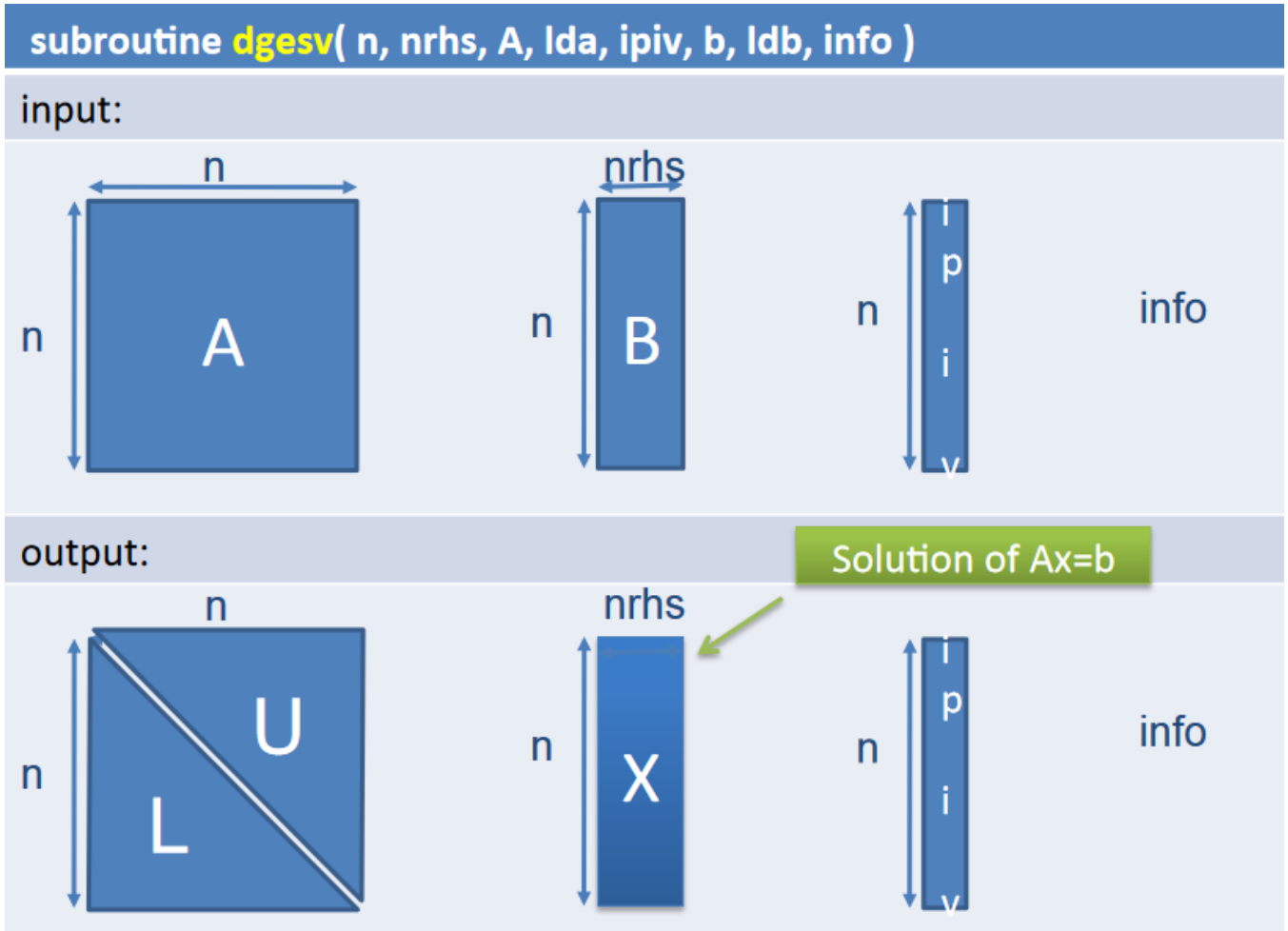
In each iteration just use this factorization to solve linear problem:

$L \cdot y = b$

$U \cdot x = y$

Memory efficient

- In-place matrix factorization:



BLAS & LAPACK implementations

- ATLAS (**A**utomatically **T**uned **L**inear **A**lgebra **S**oftware)
  - empirical tuning system that produces a BLAS library which has been specifically optimized for the platform you install ATLAS on.
  - natively provides a few routines from the LAPACK

Intel MKL

- A set of numerical routines **optimized for Intel processors** (precompiled libraries, Windows/Linux)
- Includes:
  - **BLAS Level 1-3, SparseBLAS**
  - **LAPACK**
  - Sparse linear solver PARDISO
  - Sparse QR factorization routines
  - Sparse eigenvalue solver (FEAST)
  - FFT
  - Many more…
- BUT: Maximum performance only on Intel CPUs!

AMD CPUs

- **BLIS**

BLIS is a portable open-source software framework for instantiating high-performance Basic Linear Algebra Subprograms (BLAS) – like dense linear algebra libraries. The framework was designed to isolate essential kernels of computation that, when optimized, immediately enable optimized implementations of most of its commonly used and computationally intensive operations. Select kernels have been optimized for the AMD EPYC™ processor family by AMD and others. https://github.com/amd/blis.

- **libFLAME**

libFLAME is a portable library for dense matrix computations, providing much of the functionality present in Linear Algebra Package (LAPACK). It includes a compatibility layer, FLAPACK, the FORTRAN interface, which includes complete LAPACK implementation. The library provides scientific and numerical computing communities with a modern, high-performance dense linear algebra library that is extensible, easy to use, and available under an open source license. In combination with the BLIS library which includes optimizations for the AMD EPYC™ processor family, libFLAME enables running high performing LAPACK functionalities on AMD platforms. https://github.com/amd/libflame.

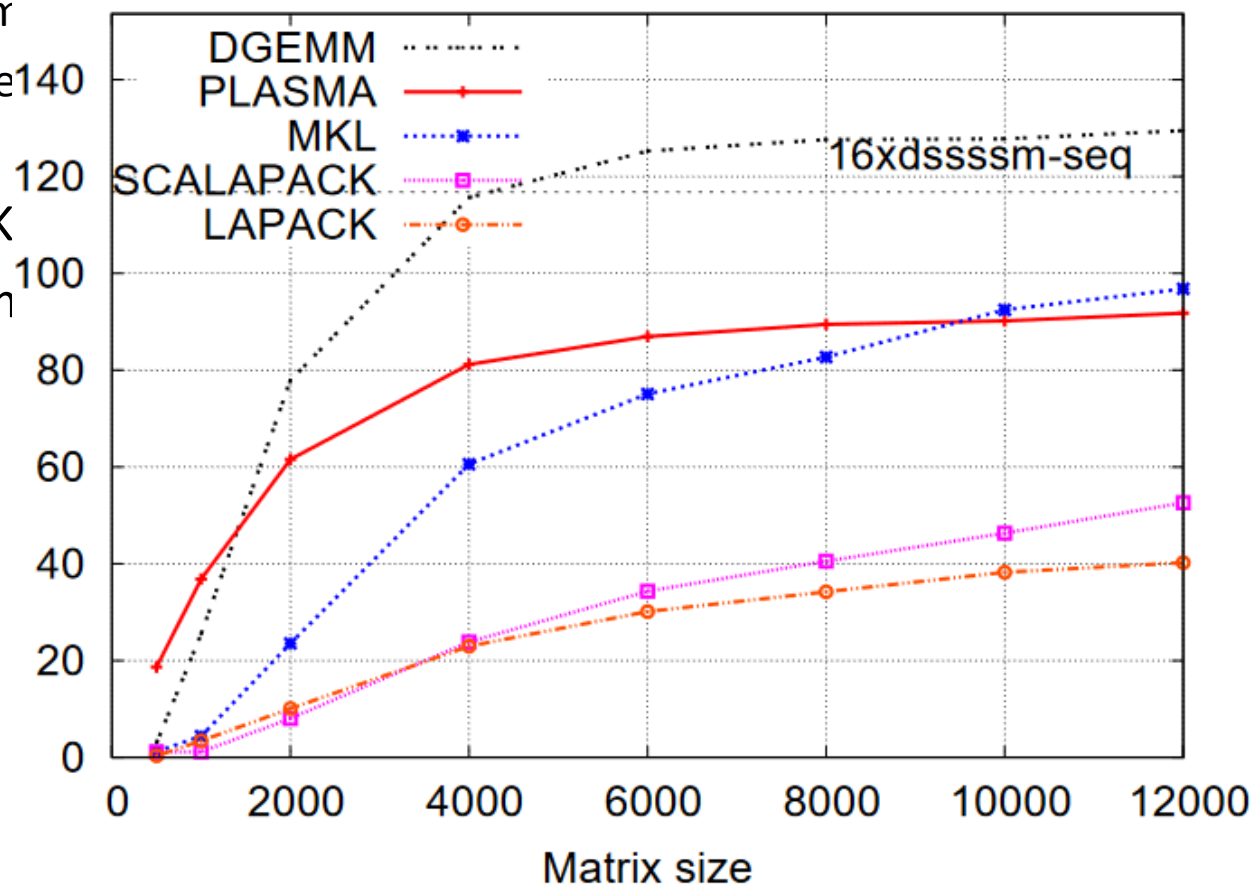
Parallelism

- Q: Is LAPACK parallel?
  - In general LAPACK is sequential code
  - Depending if the BLAS are parallel (possible in shared memory systems!)

- For distrubuted memory systems: ScaLAPACK
  - Uses MPI
  - Different data structures, algorithms than LAPACK
  - Only subset of LAPACK's functionality available: dense and banded linear systems, least squares problems, eigenvalue problems, and singular value problems
  - BSD License

PLASMA

- PLASMA (Parallel Linear Algebra Software for Multicore Archtectures)
- Objectives
  - High utilization of each core
  - Scaling to large num
  - Shared or distribute
- In contrary to LAPACK
- Designed for shared n



Gflops/s
Intel Xeon
E7340, 16 core

What else?

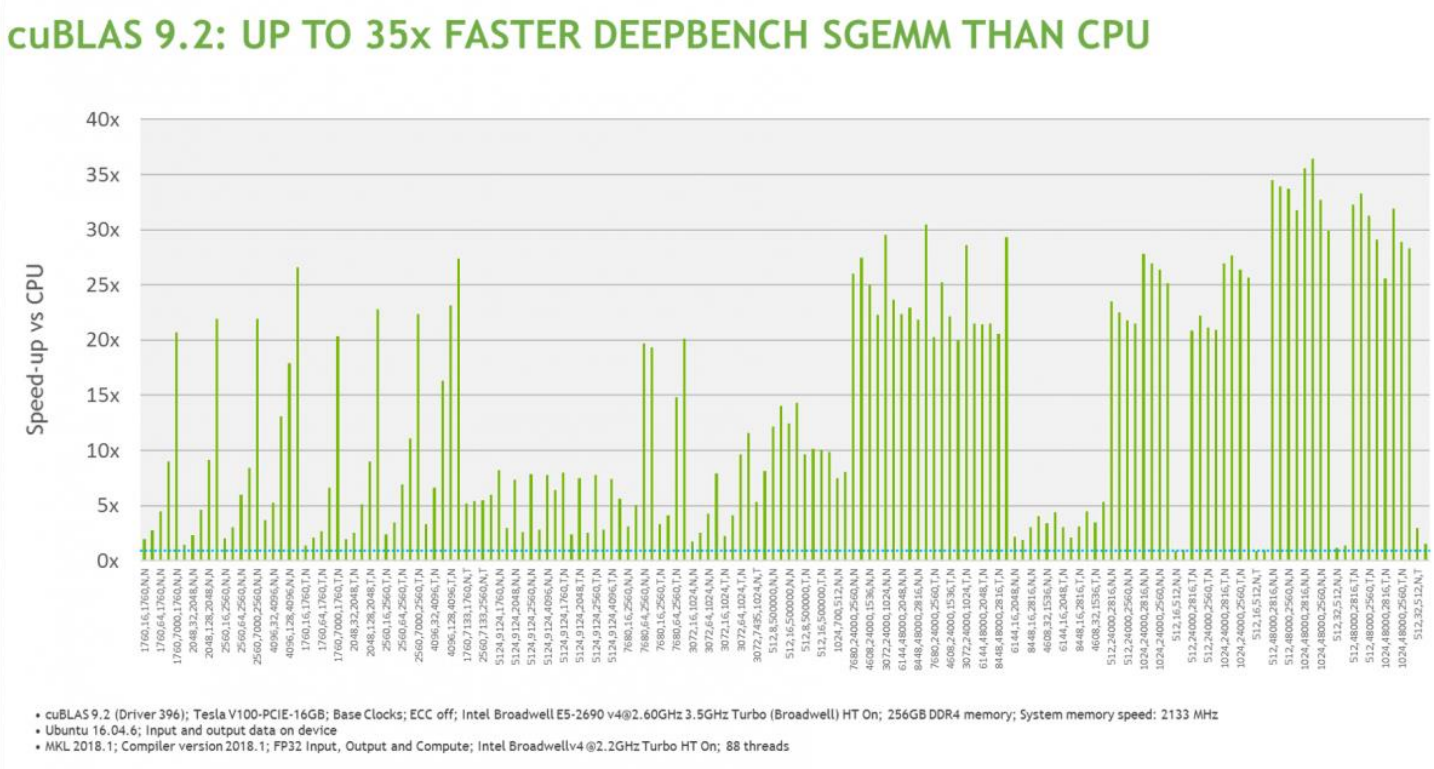
- GPU-computing, e.g. using Nvidia CUDA

- Use thousends of threads working in parallel
- Fast GDDR5 memory

- BUT:
  - All threads shall execute the same code
  - High cost of logical operators
  - Only selected algorithms can be executed efficiently!

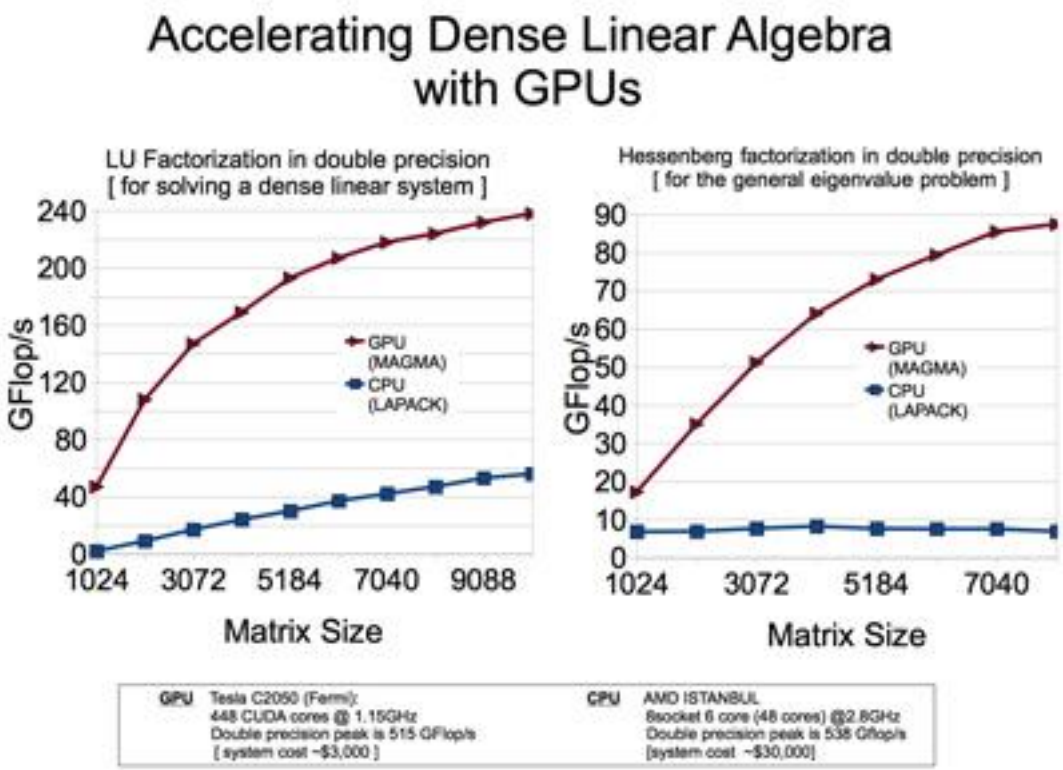
What else?

- **cuBLAS** – BLAS implementation for Nvidia GPU's



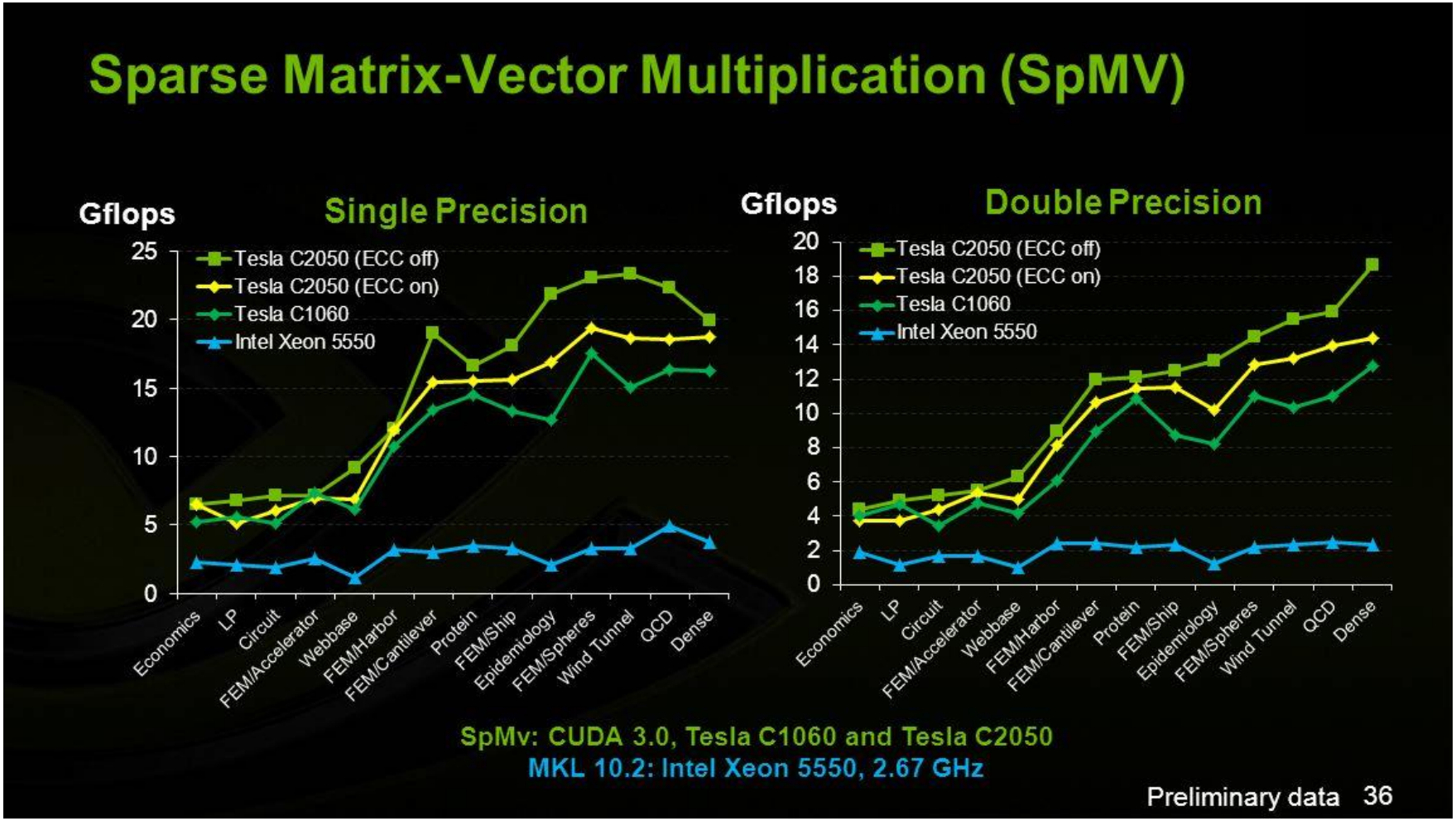
MAGMA

- Excellent performance and high accuracy (**LAPACK compliant**)
- Multiple precision arithmetic support (S/D/C/Z)
- Hybrid algorithms using both multicore CPUs and GPUs



cuSparse

- A set of routines for sparse matrix algebra



GPU computing

- How can we benefit from GPU when solving linear systems of equations?

$$A \cdot x = b$$

- Do not try to factorize matrix, use iterative methods, like conjugate-gradients:

Assume inital guess x_0, then execute:

$r_0 := b - Ax_0$

$p_0 := r_0$

$k := 0$

repeat

$$\alpha_k := \frac{r_k^\top r_k}{p_k^\top A p_k}$$

$x_{k+1} := x_k + \alpha_k p_k$

$r_{k+1} := r_k - \alpha_k A p_k$

if r_{k+1} jest "wystarczająco mały" **then** exit loop **end if**

$$\beta_k := \frac{r_{k+1}^\top r_{k+1}}{r_k^\top r_k}$$

$p_{k+1} := r_{k+1} + \beta_k p_k$

$k := k + 1$

end repeat

The most intensive operation is matrix-vector product!

Modern GPU processing applications

- **NVIDIA DRIVE AGX**
- Peak 8T FLOPS
- Integrated and discrete Pascal GPU cores
- Diverse deep learning algorithms
- Real-time processing sensor data from the camera, radar, lidar sensors and location

TESLA

GPU computing

- Q: What is the main limitation of GPU in numerical simulations?

  AVAILABLE MEMORY SIZE!

- Nvidia TESLA V100: 16GB RAM, ~8000USD!
  - Not enough to factorize large sparse matrices

- Higher programming complexity

How about PC hardware?

- Q: What is a main difference between consumer i7, i9 and server Intel Xeon processors?

Procesor Intel® Core™ i9-9900K
pamięć cache 16 MB, nawet 5,00 GHz

Specyfikacje
- Niezbędne zasoby
- Wydajność
- Informacje uzupełniające
- Dane techniczne pamięci
- Grafika wbudowana w procesor
- Opcje rozszerzeń
- Specyfikacja obudowy
- Technologie zaawansowane
- Niezawodność i bezpieczeństwo

Zamawianie i zgodność

Obrazy prezentujące produkty

Zgodne produkty

Materiały do pobrania i oprogramowanie

Niezbędne z

Kolekcja produkt	
Nazwa kodowa	
Segment rynku p	
Numer procesora	
Stan	
Data rozpoczęcia	
Litografia	
Obejmuje elemer	
Rekomendowana	
Warunki użytkow	

Wydajność

Liczba rdzeni	
Liczba wątków	
Bazowa częstotliwość procesora	3,60 GHz
Maks. częstotliwość turbo	5,00 GHz

Dane techniczne pamięci

Maks. wielkość pamięci (w zależności od rodzaju pamięci)	64 GB
Rodzaje pamięci	DDR4-2666
Maks. liczba kanałów pamięci	2
Maks. przepustowość pamięci	41,6 GB/s
Obsługa pamięci ECC ‡	Nie

Intel® 64 ‡	Tak
Zestaw instrukcji	64-bit
Rozszerzony zestaw instrukcji	Intel® SSE4.1, Intel® SSE4.2, Intel® AVX2

How about PC hardware?

- Q: What is a main difference between consumer i7, i9 and server Intel Xeon processors?

Intel® Xeon® Gold 6144 Pr
24.75M Cache, 3.50 GHz

Specyfikacje
Niezbędne zasoby
Wydajność
Informacje uzupełniające
Dane techniczne pamięci
Opcje rozszerzeń
Specyfikacja obudowy
Technologie zaawansowane
Niezawodność i bezpieczeństwo

Zamawianie i zgodność

Obrazy prezentujące produkty

Zgodne produkty

Materiały do pobrania i oprogramowanie

Dane techniczne pamięci

Maks. wielkość pamięci (w zależności od rodzaju pamięci)	768 GB
Rodzaje pamięci	DDR4-2666
Maksymalna szybkość pamięci	2666 MHz
Maks. liczba kanałów pamięci	6
Obsługa pamięci ECC ‡	Tak

Bazowa częstotliwość procesora	3,50 GHz
Maks. częstotliwość turbo	4,20 GHz

Intel® 64 ‡	Tak
Rozszerzony zestaw instrukcji	Intel® SSE4.2, Intel® AVX, Intel® AVX2, Intel® AVX-512
Liczba jednostek AVX-512 FMA	2

Summary

- Number of cores per chip doubles every second year, while clock speed decreases (not increases)
- Number of threads of execution doubles every second year

- Multihreading programming - interchip and intrachip
  - **If not on your own**, **please use third-party, parallel libraries**:
    - BLAS
    - LAPACK
    - ATLAS
    - PLASMA
    - …
  - Do not reinvent the wheel, unless you know what you are doing!