PeC^3 Spring School on

Introduction to Numerical Modeling with Differential Equations

Thomas Wick[1] and Peter Bastian[2]

[1]Leibniz Universität Hannover
Institut für Angewandte Mathematik
Welfengarten 1, 30167 Hannover
email: thomas.wick@ifam.uni-hannover.de

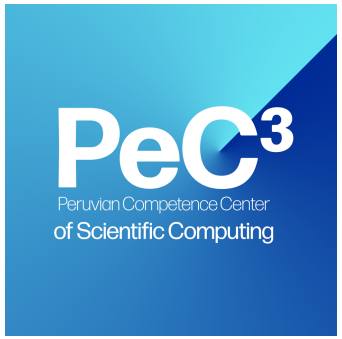
[2]Universität Heidelberg
Interdisziplinäres Zentrum für Wiss. Rechnen
Im Neuenheimer Feld 205, D-69120 Heidelberg
email: Peter.Bastian@iwr.uni-heidelberg.de

Universidad Nacional Agraria La Molina,
Lima, Perú, October 23–31, 2019

Acknowledgments

This spring school is part of the DAAD (German academic exchange service) program

PeC^3 : Peruvian Competence Center of Scientific Computing



`https://www.pec3.org/`

- The support from the DAAD and the Federal Ministry for Economic Cooperation and Development is gratefully acknowledged.
- Moreover, we highly appreciate the support of the local organizers, specifically Dandy Rueda Castillo.

Key literature related to this summer school

1. K. Eriksson, D. Estep, P. Hansbo, C. Johnson; Computational Differential Equations, Vol. 1, Cambridge University Press, 538 pages, 1996; online pdf-version available from the year 2009 `http://www.csc.kth.se/~jjan/private/cde.pdf`
2. A. Quarteroni, F. Saleri, P. Gervasio; Scientific Computing with MATLAB and octave, Springer 2014
3. R. Rannacher; Numerik 1: Numerik gewöhnlicher Differentialgleichungen, Heidelberg University Publishing, 2017 (in german)
4. P. Bastian; Vorlesung Numerik, lecture notes, Heidelberg University, 2017
5. P. Bastian; Lecture Notes Scientific Computing with Partial Differential Equations, Universität Heidelberg, WS 2017/2018. `https://conan.iwr.uni-heidelberg.de/data/teaching/finiteelements_ws2017/num2.pdf`
6. T. Wick; Numerical methods for partial differential equations; Lecture notes, Leibniz Universität Hannover, 2018, online available `http://www.thomaswick.org/links/lecture_notes_Numerics_PDEs_Oct_12_2019.pdf`
7. T. Wick; Introduction to Numerical Modeling, lecture notes, MAP 502, Ecole Polytechnique, 2018, online available `http://www.thomaswick.org/map_502_winter_2018_engl.html`
8. O. Klein; Lecture Notes Object-Oriented Programming for Scientific Computing, 2018, `https://conan.iwr.uni-heidelberg.de/teaching/oopfsc_ss2018/`
9. O. Klein; Lecture Notes Einführung in die Numerik, 2018, `https://conan.iwr.uni-heidelberg.de/teaching/numerik0_ss2018/`

Contents

Contents

Plan

- Motivation: Differential equations in their historical context
- Very short review of basics of calculus
- What is a differential equation?
- Derivation of some differential equation models

The Modern Scientific Method

- *Experiment*: Observe and measure some phenomenon. Nowadays the amount of data acquired may be abundant.
- *Theory*: Try to explain observations by a model. Here we consider mathematical models in terms of differential equations.
- *Scientific Computing*: Often the parametrization and prediction of the model is achieved with the help of computers.
- Compare measurements and predictions to improve your model and/or observations.

- Today often *data-driven* "models" obtained with machine-learning are used. A disadvantage of these models is that they do not help us to *understand* how the observed system works.

Why Differential Equations?

- Differential equations are ubiquitous in science and technology
  - Solid mechanics: stability of bridges and buildings
  - Fluid mechanics: drag and lift of an airfoil
  - Material science: phase diagram of a substance
  - Protein folding: How do molecules bind
  - Mathematical biology: How does cancer grow
  - Hydrology: movement of contamination in groundwater
  - Weather and climate prediction
- Numerical methods are often the only way to solve these equations in practical situations
- Approach was enabled by the digital computer (although already envisioned by Leibniz 300 years ago!)
- Most of the time on supercomputers is spent solving differential equations

Historical Perspective

- Sir Isaac Newton (1642-1727) and Gottfried Wilhelm Leibniz (1646-1716) invent calculus
- Newton described motion of the planets by differential equations
- Leibniz developed also mechanical calculating machines
- Euler (1707-1783) and Lagrange (1736-1813) develop *variational calculus*
- Euler finds equations for inviscid fluid flow (1757)


Newton


Leibniz

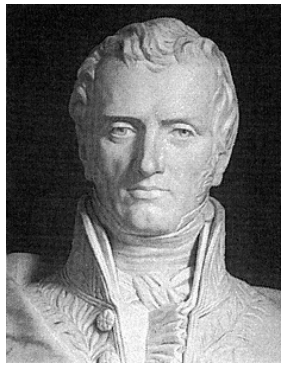

Euler


Lagrange

Equations of Mathematical Physics

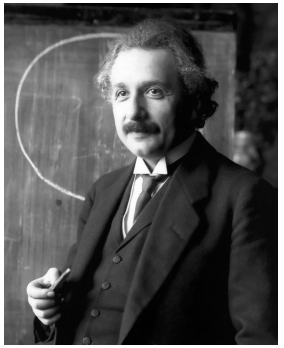
- PDEs ubiquitous in physics
- E.g. to express conservation of mass, momentum and energy in quantitative form
- Poisson (electrostatics, gravity) ∼1800
- Navier-Stokes (viscous flow) 1822/1845
- Maxwell (electrodynamics) 1864
- Einstein (general relativity) 1915
- Schrödinger (quantum mechanics) 1926


Poisson


Navier


Stokes


Maxwell


Einstein


Schrödinger

Calculating Machines and Computers

- 1671: Leibniz builds a machine to do $+, -, \cdot, /$
- 1831: Charles Babbage designs the steam-powered *Analytical Engine*, a programmable calculator including conditions and loops (not completed)
- 1921: Lewis Fry Richardson proposes to predict the weather based on differential equations using 64000 human computers
- 1937: Alan Turing founds *computability theory* based on the *Turing machine*
- World War II pushed the development of computers: firing tables (ENIAC 1943-46), deciphering codes (Colossus, 1943)
- 1945: John von Neumann proposes the stored-program computer (based on others' ideas)
- 1965: Gordon Moore predicts doubling of # transistors on a chip every 12 month (it is more like 18)
- Today: *You* will witness the end of Moore's law

Milestone Algorithms

- ca. 1670: Newton introduces a method to solve polynomial equations
- 1823: Carl Friedrich Gauß mentions an iterative method to solve (least squares) linear systems of equations arising in the triangulation of Hannover
- 1943: Richard Courant publishes an early version of the Finite Element Method based on the Ritz-Galerkin principle
- 1965: Cooley and Tukey publish the Fast Fourier Transform algorithm
- 1977: Wolfgang Hackbusch and Achi Brandt independently introduce the multigrid method
- 1987: Leslie Greengard and Vladimir Rokhlin Jr. introduce the fast multipole method
- 1999: Wolfgang Hackbusch introduces H-matrices allowing sparse approximation of non-sparse matrices

Sequences

- By $\{\xi_i : i \in \mathbb{N}\}$ we denote a *sequence* of real numbers
- Examples:

$$\xi_i = \frac{1}{i}, \qquad \xi_i = \frac{2i}{i+1}, \qquad \xi_{i+1} = \frac{1}{2}\left(\xi_i + \frac{a}{\xi_i}\right)$$

- A sequence $\{\xi_i\}$ *converges* to the limit $\xi \in \mathbb{R}$ if for any $\epsilon > 0$ there exists $N_\epsilon \in \mathbb{N}$ such that

$$|\xi - \xi_i| < \epsilon \qquad \text{for all } i \geq N_\epsilon$$

- A sequence $\{\xi_i\}$ *is a Cauchy sequence* if for any $\epsilon > 0$ there exists $N_\epsilon \in \mathbb{N}$ such that

$$|\xi_j - \xi_i| < \epsilon \qquad \text{for all } i, j \geq N_\epsilon$$

- Every convergent sequence is a Cauchy sequence
- Completeness *axiom*: Every Cauchy sequence converges in $\mathbb{R}$ (this is what makes $\mathbb{R}$ different from $\mathbb{Q}$)

Functions and Continuity

- $\Omega \subseteq \mathbb{R}$ connected. $u : \Omega \to \mathbb{R}$ denotes a *function* u mapping

$$x \in \Omega \to u(x) \in \mathbb{R}$$

- u is said to be *continuous in* x if

$$\{x_i\} \to x \quad \Rightarrow \quad \{u(x_i)\} \to u(x)$$

- Equivalently: for any $\epsilon > 0$ there exists $\delta_\epsilon(x) > 0$ such that

$$|u(x) - u(y)| < \epsilon \qquad \text{for all } 0 < |x - y| < \delta_\epsilon(x)$$

- If u is continuous in all $x \in \Omega$ then u is a continuous function
- If δ_ϵ is independent of x then u is *uniformly* continuous
- The set $C(\Omega)$ of all continuous functions on Ω is closed under addition and scalar multiplication, thus it forms a *vector space*
- For Ω closed and bounded u is bounded and $(C(\Omega), \|.\|_\infty)$ is a normed vector space

Differentiable Functions I

- $u'(x)$ is called the *derivative of u in x* if for *all* sequences $\{x_i\}$ converging to x the limit

$$\lim_{x_i \to x} \frac{u(x) - u(x_i)}{x - x_i}$$

  exists and is the same
- Clearly, continuity of u is a necessary condition for the derivative to exist
- Then both, nominator and denominator, converge to zero
- If u is differentiable in all $x \in \Omega$, u is *differentiable* and the function u' with values $u'(x)$ is called the *derivative of u*
- Later we will use the equivalent notation $\frac{du}{dx}(x) = u'(x)$
- We may write $u' = Du$. D maps a differentiable function to its derivative

Differentiable Functions II

- And we may differentiate again: $u'' = Du' = DDu = D^2u$
- And in general $u^{(n)} = D^n u$ is the *n'th derivative of u*
- It is convenient to set $D^0 u = u$
- $D : \mathcal{D} \to \mathcal{R}$ maps a function to a function and is called "differentiation operator"
- What are its domain $\mathcal{D}$ and range $\mathcal{R}$?
- $\mathcal{D} = C^1(\Omega) = \{u \in C(\Omega) : u \text{ is differentiable}\} \subseteq C(\Omega)$
- $\mathcal{R} = C(\Omega) = C^0(\Omega)$
- In general we denote by $C^n(\Omega)$ the space of n times continuously differentiable functions
- D is a linear operator:

$$D(u_1 + u_2) = Du_1 + Du_2, \quad D(cu) = cDu, c \in \mathbb{R}$$

What is a Differential Equation?

- A *differential equation* (DE) relates values of derivatives of one or more functions over a domain $\Omega \subseteq \mathbb{R}$
- An example is

$$\Psi(u'(x), u(x), x) = 0 \quad \text{for all } x \in \Omega \tag{1}$$

- $\Psi : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a function in three arguments
- Only values at *the same point* x are related with each other
- Continuity, differentiability of u relate values at nearby points!
- (1) is a *first order ordinary DE in implicit form*
- Order: highest derivative occuring
- Ordinary DE (ODE): functions in *one* variable are involved
- Partial DE (PDE): functions in *several* variables are involved
- *Explicit form* would read: $u'(x) = f(x, u(x))$ for all $x \in \Omega$

Fundamental Theorem of Calculus

- The simplest differential equation would be the following:

$$u'(x) = f(x) \quad \forall x \in \Omega \tag{2}$$

- Assume $u(a)$ is known, then for any $b > a$ we get

$$\int_a^b u'(x)dx = u(b) - u(a) = \int_a^b f(x)dx \Leftrightarrow u(b) = u(a) + \int_a^b f(x)dx$$

- So, the function $u(x) = u(a) + \int_a^x f(\xi)d\xi$ solves (2)
- In fact, any function $u(x) + c$, $c \in \mathbb{R}$, is also a solution
- So, the solution of (2) is unique up to a constant
- A specific solution is picked by fixing $u(a)$ at some point a
- The *fundamental theorem of calculus* states that the function $\int_a^x f(\xi)d\xi$ is well defined and solves (2)

Two Simple Approximation Methods

We want to solve $u'(x) = f(x)$ in $(a, b]$, $u(a) = u_a$

- **Method 1**: Set $h_N = (b-a)/N$ and $x_i^N = a + ih_N$, $0 \leq i \leq N$
  For h_N sufficiently small,

$$\frac{u(x_{i+1}^N) - u(x_i^N)}{h_N} \approx u'(x_i^N) = f(x_i^N)$$

  leading to the scheme $u_{i+1}^N = u_i^N + h_N f(x_i^N)$
- **Method 2**: A good approximation of the integral is the *trapezoidal rule*

$$\int_x^{x+h} f(\xi)d\xi \approx \frac{h}{2}\left(f(x) + f(x+h)\right)$$

  leading to the scheme $u_{i+1}^N = u_i^N + \frac{h}{2}\left(f(x_i^N) + f(x_{i+1}^N)\right)$

Conceptual Model for Growth

- We wish to model growth of a population, e.g. bacteria in a petri dish
- In a *conceptual model* we list properties we consider (un-) important for the model
  - $N(t)$ is the number of individuals at time t
  - As a generalization, let $N(t)$ be a real number
  - We assume the spatial extend to be unimportant
  - Increase in population during an interval Δt is proportional to number $N(t)$ and Δt
  - This last assumption means that *infinite resources* (food, energy) are available for growth

Mathematical Growth Model

- Consider the *change* of the population in a time interval Δt:

$$N(t + \Delta t) = N(t) + \lambda \Delta t N(t) \tag{3}$$

  with a constant $\lambda \in \mathbb{R}$
- Observe similarity of (3) with the numerical method 1 above!
- For a fixed Δt this is a discrete model. For different Δt, different sequences of numbers are obtained
- Rearranging gives

$$\frac{N(t + \Delta t) - N(t)}{\Delta t} = \lambda N(t)$$

- Considering the limit $\Delta t \to 0$ we obtain a linear ordinary differential equation:

$$N'(t) = \lambda N(t) \tag{4}$$

Analytical Solution of Growth Model

- This simple ODE can be solved analytically
- Verify that $Ce^{\lambda t}$ for any $C \in \mathbb{R}$ is a solution
- The initial condition $N(t_0) = N_0$ picks the solution $N_0 e^{\lambda(t-t_0)}$
- Are *all* solutions of the form $Ce^{\lambda t}$?
- Let $N(t)$ be any solution of (4), then:

$$\left(N(t)e^{-\lambda t}\right)' = N'(t)e^{-\lambda t} - \lambda N(t)e^{-\lambda t} = \left(N'(t) - \lambda N(t)\right)e^{-\lambda t} = 0$$

- From $\left(N(t)e^{-\lambda t}\right)' = 0$ we conclude

$$N(t)e^{-\lambda t} = C \Leftrightarrow N(t) = Ce^{\lambda t}$$

- More realistic growth models consider finite resoures, e.g. *logistic growth*

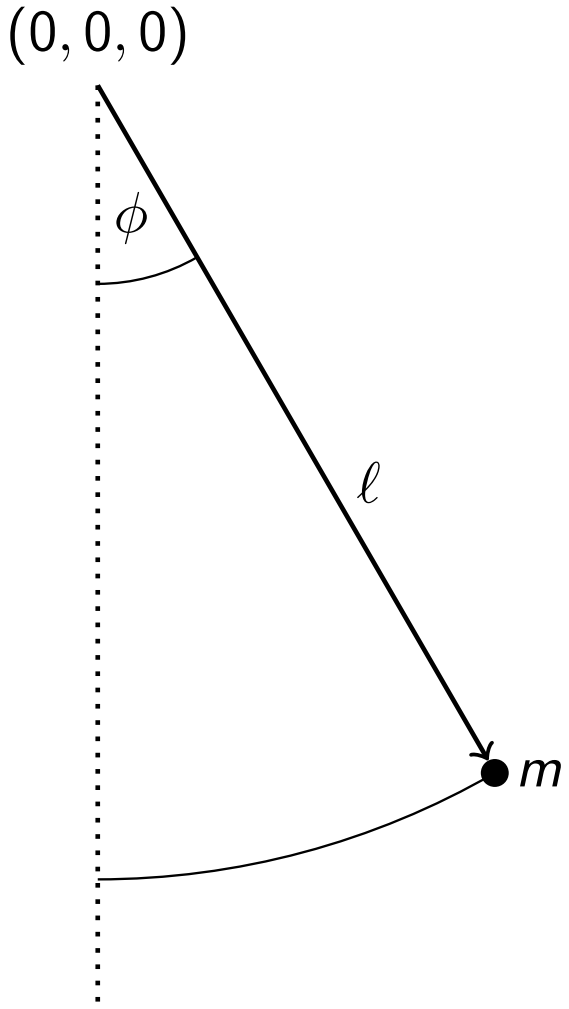
$$N'(t) = \lambda(G - N(t))N(t) \qquad \text{(Bernoulli DE)}$$

Conceptual Model of a Pendulum

We wish to model a pendulum
In a *conceptual model* we list properties we consider (un-) important for the model

- The weight is concentrated in a point of mass m
- The rod of length ℓ is assumed rigid and massless
- The rod is fixed at $(0, 0, 0)$ and movement is in the plane $y = 0$
- Air resistance is neglected

We develop a mathematical model based on Newton's equations of motion



Forces

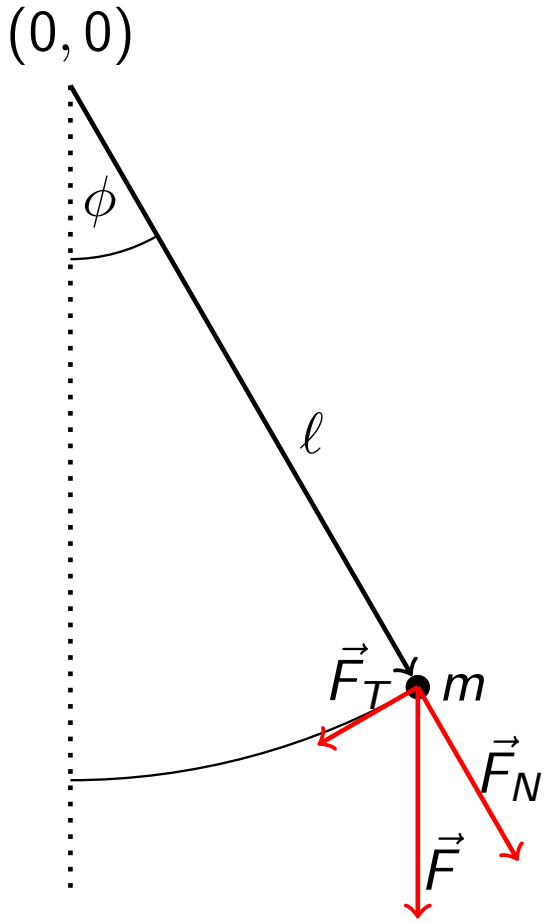
- Movement is along a circle, only force in *tangential direction* is relevant for acceleration
- Tangential force for deflection angle ϕ:

$$\vec{F}_T(\phi) = -\underbrace{m\,g}_{|\vec{F}|} \sin(\phi) \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \end{pmatrix}$$

- For example $\phi = 0$, $\phi = \pi/2$:

$$\vec{F}_T(0) = mg\,0 \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \quad \vec{F}_T(\pi/2) = mg \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

- Sign encodes direction



Distance, Velocity, Acceleration

- Distance $s(t)$, velocity $v(t)$, acceleration $a(t)$ satisfy:

$$v(t) = \frac{ds(t)}{dt}, \qquad a(t) = \frac{dv(t)}{dt}.$$

- The distance (including sign) satisfies $s(t) = \ell\phi(t)$.
- Therefore velocity satisfies

$$v(t) = \frac{d\, s(\phi(t))}{dt} = \frac{d\, \ell\phi(t)}{dt} = \ell\frac{d\phi(t)}{dt}$$

- and acceleration satisfies

$$a(t) = \frac{d\, v(\phi(t))}{dt} = \ell\frac{d^2\phi(t)}{dt^2}.$$

Equations of Motion

- Insert into Newton's second law $m\,a(t) = F(t)$ gives:

$$m\ell \frac{d^2\phi(t)}{dt^2} = -mg\sin(\phi(t)) \qquad \forall t > t_0.$$

- The force is scalar as we are only considering the distance travelled where sign encodes direction
- We obtain a second-order nonlinear ordinary differential equation for the deflection angle $\phi(t)$:

$$\frac{d^2\phi(t)}{dt^2} = -\frac{g}{\ell}\sin(\phi(t)) \qquad \forall t > t_0. \tag{5}$$

- A unique solution is determined by *two* initial conditions ($t_0 = 0$):

$$\phi(0) = \phi_0, \qquad \frac{d\phi}{dt}(0) = \phi_0'. \tag{6}$$

Solution for Small Deflection Angle

- For *small* deflection angle ϕ observe

$$\sin(\phi) \approx \phi,$$

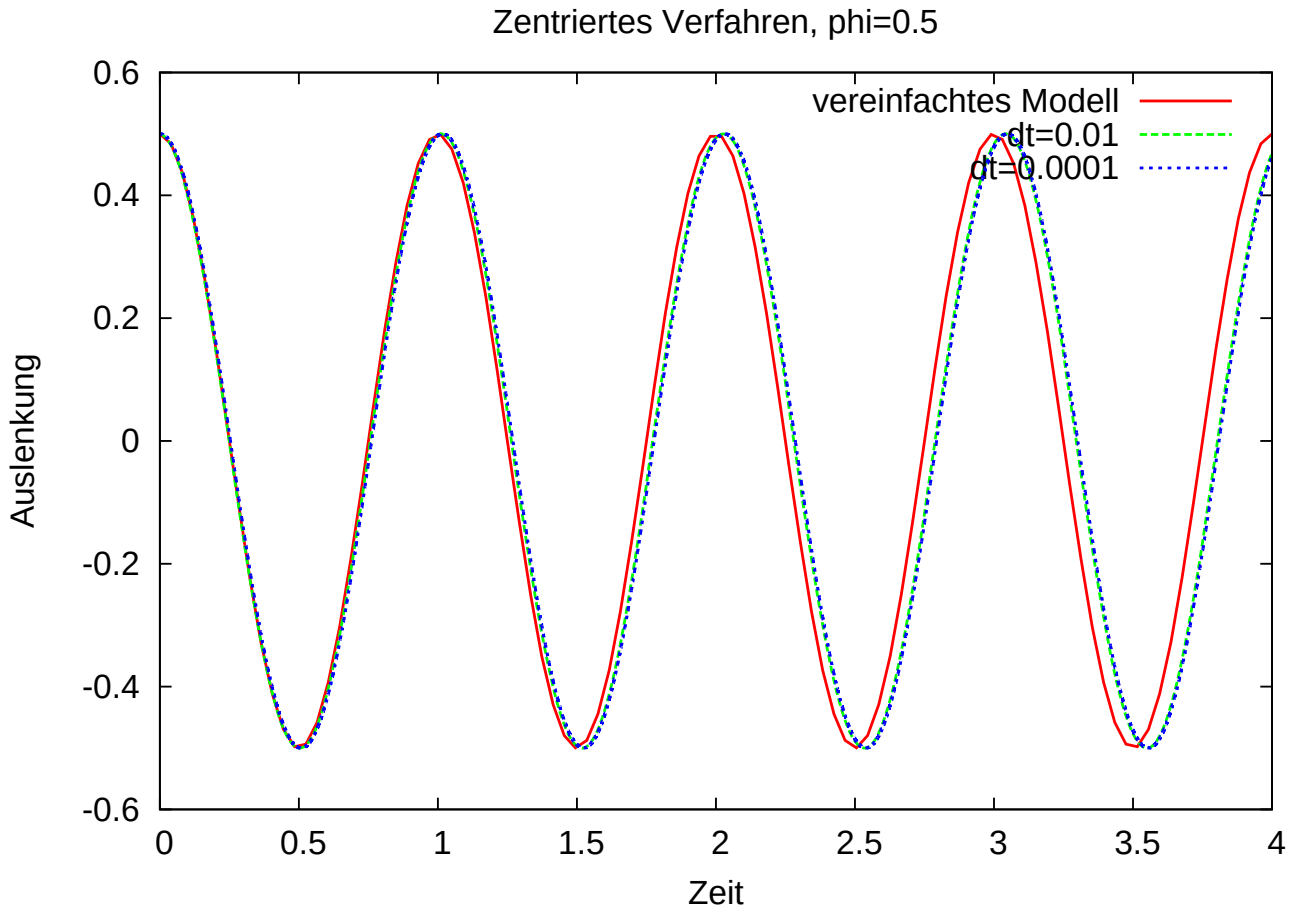
  e.g. $\sin(0.1) = 0.099833417$.
- Using this *approximation* yields the linear ODE

$$\frac{d^2\phi(t)}{dt^2} = -\frac{g}{\ell}\phi(t)$$

- Which is solved by $\phi(t) = A\cos(\omega t)$. The constants are fixed by the initial conditions $\phi(0) = \phi_0$, $\frac{d\phi}{dt}(0) = 0$, giving

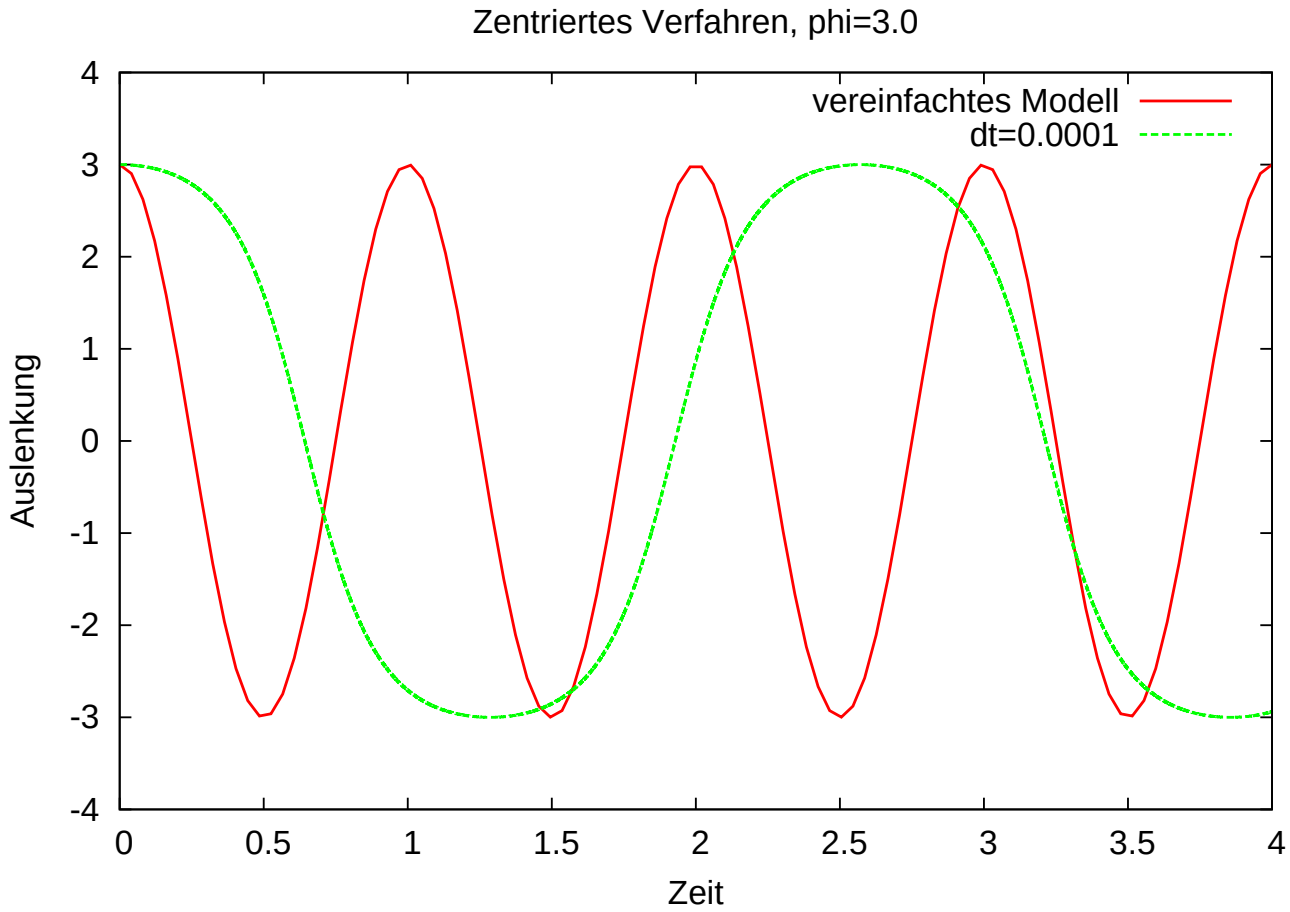
$$\phi(t) = \phi_0 \cos\left(\sqrt{\frac{g}{\ell}}t\right) \tag{7}$$

Model Comparison $\phi_0 = 0.5 \approx 28.6°$



Even for 28.6° the approximation is quite accurate

Model Comparison $\phi_0 = 3.0 \approx 171°$



For such large deformations the approximate model is very inaccurate

Elementary Reactions

- A reaction of three substances A, B, C is denoted by

$$\nu_a A + \nu_b B \quad \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \quad \nu_c C \tag{8}$$

- Forward reaction: ν_a molecules of A react with ν_b molecules of B to give ν_c molecules of C with reaction speed k_1
- The reverse reaction has speed k_2
- ν_a, ν_b, ν_c are the *stoichiometric coefficients*
- Reaction rates $k_j = A_j \exp(-E_j/(RT))$ given by *Arrhenius' law*
- *ODE system*: concentrations $c_i(t)$, $i = A, B, C$ in mol/m^3:

$$\frac{dc_C}{dt}(t) = \nu_c k_1 c_A^{\nu_a}(t) c_B^{\nu_b}(t) - \nu_c k_2 c_C^{\nu_c}(t)$$
$$\frac{dc_A}{dt}(t) = -\nu_a k_1 c_A^{\nu_a}(t) c_B^{\nu_b}(t) + \nu_a k_2 c_C^{\nu_c}(t)$$
$$\frac{dc_B}{dt}(t) = -\nu_b k_1 c_A^{\nu_a}(t) c_B^{\nu_b}(t) + \nu_b k_2 c_C^{\nu_c}(t)$$

- Equilibrium given by: $\frac{k_1}{k_2} = \frac{c_C^{\nu_c}}{c_A^{\nu_a} c_B^{\nu_b}}$ (*mass action law*)

Astrophysical N-body Problem

- Consider N bodies of mass m_i at positions $x_i(t) \in \mathbb{R}^3$
- The *gravitational force* $F_{ij} \in \mathbb{R}^3$ excerted from body j on body i is

$$F_{ij}(x_i, x_j) = G \frac{m_i m_j}{\|x_j - x_i\|^2} \frac{x_j - x_i}{\|x_j - x_i\|}$$

  where G is the *gravitational constant*
- Newton's 2nd law $F_i(t) = m_i a_i(t)$ gives ODE system:

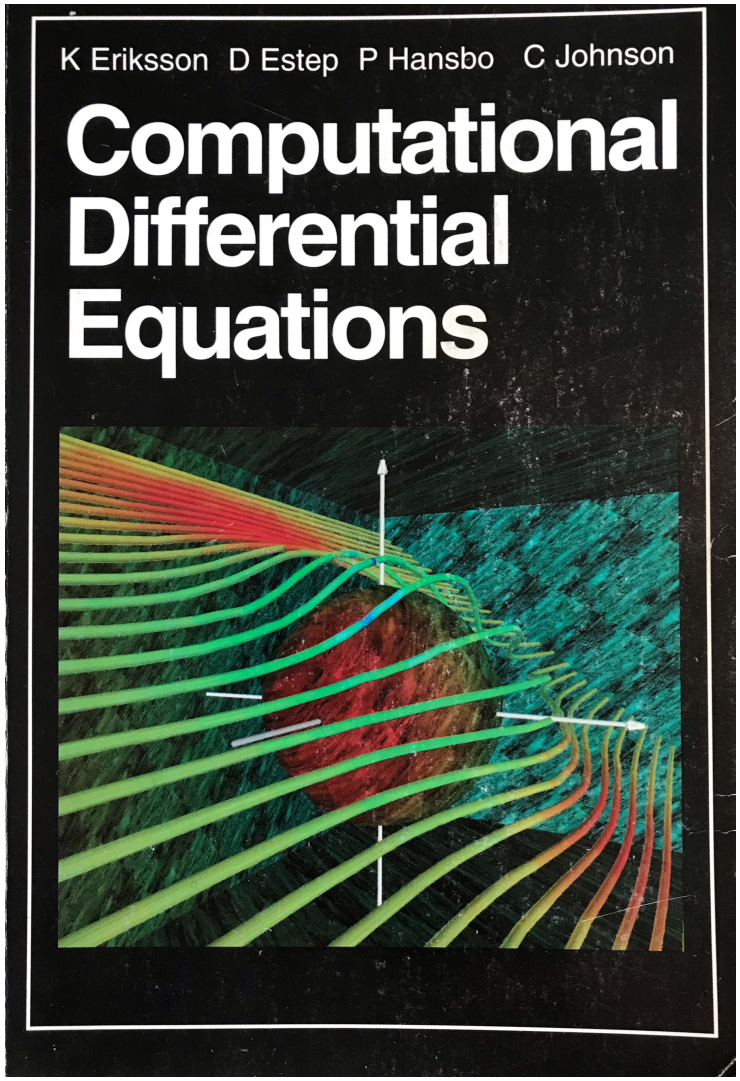
$$\frac{d^2 x_i(t)}{dt^2} = G \sum_{j=1, j\neq i}^{N} m_j \frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|^3} \qquad i = 1, \ldots, N$$

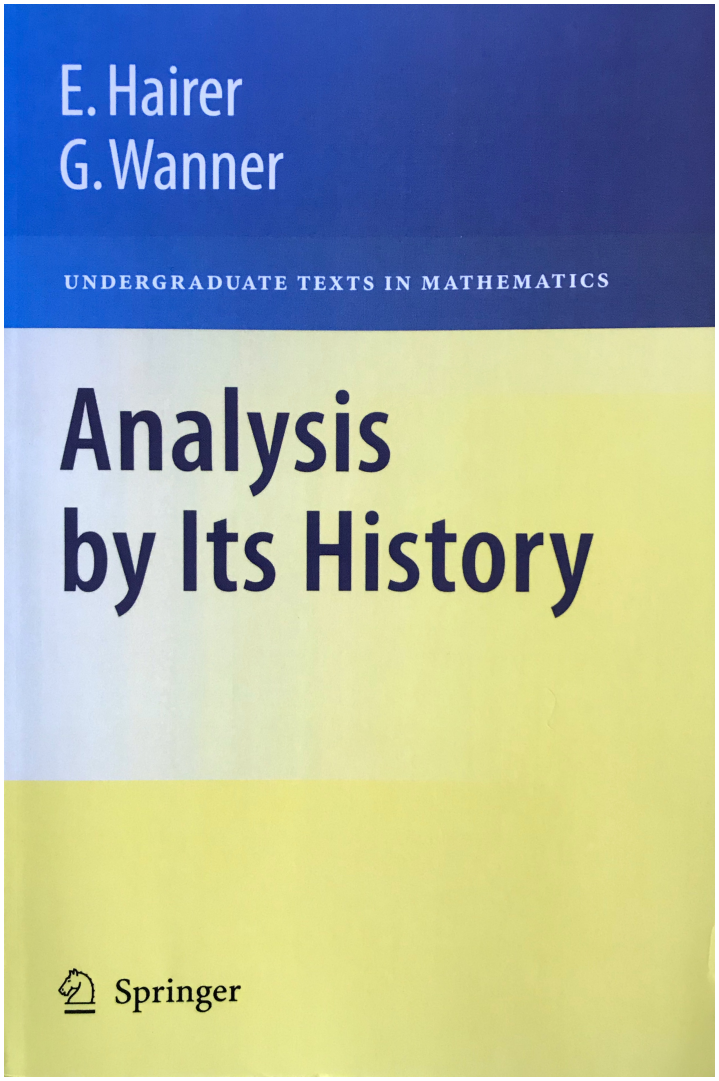
- Introducing velocity $v_i(t) = \frac{dx_i(t)}{dt}$ results in first-order system:

$$\frac{dx_i(t)}{dt} = v_i(t), \quad \frac{dv_i(t)}{dt} = G \sum_{j=1, j\neq i}^{N} m_j \frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|^3} \quad i = 1, \ldots, N$$

- These are $6N$ coupled ODEs requiring $6N$ initial conditions

Recommended Reading





http://www.csc.kth.se/~jjan/private/cde.pdf

Summary Lecture 01

- Differential equations as part of the scientific method
- First glimpse on numerical solution
- Derivation of some differential equation models

Contents

Plan

- First order and second order differential equations
- Classification into linear/nonlinear problems
- Explicit and implicit formulations
- Model problem $u' = f(t, u)$
- Guiding questions and numerical concepts
- In all parts: a lot of notation and 'language' to talk and deal with differential equations

Differential equations

We recall from lecture 01:

Definition 1 (Differential equation (DE))

A differential equation is a mathematical equation that relates a (unknown) function with its derivatives.

Differential equations

We recall from lecture 01:

Definition 1 (Differential equation (DE))

A differential equation is a mathematical equation that relates a (unknown) function with its derivatives.

Differential equations can be split into two classes:

Definition 2 (Ordinary differential equation (ODE))

An ordinary differential equation (ODE) is an equation (or equation system) involving an unknown function of one independent variable and certain of its derivatives. Often: either space x or time t.

Definition 3 (Partial differential equation (PDE))

A partial differential equation (PDE) is an equation (or equation system) involving an unknown function of two or more variables and certain of its partial derivatives. Often: x and t or even (x, y, z) and t.

Classifications

- Order of a differential equation
- Single equations and PDE systems
- Nonlinear problems:
  - Nonlinearity in the PDE
  - The function set is not a vector space yielding a variational inequality
- Coupled problems and coupled PDE systems.

Order of differential equations

- The **order** of a differential equation is determined by its highest derivatives
- First order ODE:

$$y'(t) = f(y, t), \quad y(t_0) = y_0$$

- Second order ODE:

$$y''(t) = f(y', y, t), \quad y(t_0) = y_0, y'(t_0) = v_0$$

- Higher order:

$$y^{(m)}(t) = f(y^{(m-1)}, \ldots, y, t)$$

plus $m - 1$ initial conditions.

Reduction of Higher-order Equations to First-order Systems

Higher-order DE can be reduced to low-order DE systems by introducing auxiliary solution functions.

- Given $y^{(m)}(t) = f(y^{(m-1)}, \ldots, y, t)$
- Define

$$y_1(t) = y(t)$$

$$\vdots$$

$$y_m(t) = y^{(m-1)}(t)$$

- This results in the first-order DE system:

$$y_1'(t) = y_2(t)$$

$$\vdots$$

$$y_{m-1}'(t) = y_m(t)$$

$$y_m'(t) = f(t, y_1, \ldots, y_m)$$

- With this, we can write the original problem in compact form (vector-valued problem!) as

$$y'(t) = f(t, y)$$

Examples

1. $y'(t) = t^{-1}y(t)$: is of 1st order, linear with the solution $y(t) = t$
2. $y'(t) = ty(t)^{-1}$: 1st order nonlinear, singular solution for $y(t) \to 0$.
3. $y'(t) = y(t)^2$: 1st order nonlinear with a singularity at $t = 1$. Only local solution with $y(t) = \sqrt{1 + t^2}$
4. Clothesline problem (membrane deformation):

$$-u''(x) = f \quad \text{and } u(0) = u(1) = 0.$$

Single equations vs. systems

Definition 4 (Single equation)

A single DE consists of determining one solution variable, e.g.,

$$u : \Omega \subset \mathbb{R}^d \to \mathbb{R}.$$

Typical examples are Poisson's problem, the heat equation, wave equation, Monge-Ampère equation, Hamiliton-Jacobi equation, p-Laplacian.

Definition 5 (DE system)

A diff. eq. system determines a solution vector

$$u = (u_1, \ldots, u_d) : \Omega^d \to \mathbb{R}^d.$$

For each $u_i, i = 1, \ldots, d$, a DE must be solved. Inside these diff. equ. the solution variables may depend on each other or not. Typical examples of systems are predator-prey systems, linearized elasticity, nonlinear elasto-dynamics, Maxwell's equations.

Implicit differential equations

- So far: $y'(t) = f(y, t)$, which is an explicit representation of a DE
- However, not always, we can resolve with respect to the highest derivative:
$$F(y', y, t) = 0.$$
- Example:
$$F(t, y, y') = (y')^2 + uu' - 3(y')^5.$$
- Of course explicit forms of DE can always be written as implicit forms (e.g., model problem):
$$F(t, y, y') = y'(t) - f(y, t).$$
- Nonlinear numerical methods (fixed-point or Newton) required for the solution!

The model problem

- In general a model problem stands as a characteristic class of similar equations and which can exemplarely analyzed as a prototype problem!
- The model problem for ODEs is defined as:

Formulation 1

Given $a \in \mathbb{R}$. Let $I := [t_0, t_0 + T]$ the time interval with the end time value $T > 0$. Find $y : I \to \mathbb{R}$ such that

$$y'(t) = ay(t), \quad y(t_0) = y_0, \quad t \geq t_0$$

The first term is the ODE. Here, $y' = \frac{dy}{dt}$. The second term is the so-called initial condition.

The model problem

- In general a model problem stands as a characteristic class of similar equations and which can exemplarely analyzed as a prototype problem!
- The model problem for ODEs is defined as:

Formulation 1

Given $a \in \mathbb{R}$. Let $I := [t_0, t_0 + T]$ the time interval with the end time value $T > 0$. Find $y : I \to \mathbb{R}$ such that

$$y'(t) = ay(t), \quad y(t_0) = y_0, \quad t \geq t_0$$

The first term is the ODE. Here, $y' = \frac{dy}{dt}$. The second term is the so-called initial condition.

- Important **theoretical** concepts such as existence, uniqueness, stability are usually introduced in terms of this ODE.
- Moreover, important **numerical** concepts such as convergence order, efficiency, accuracy, stability are analyzed for this ODE as well.

Well-posedness: preliminaries

The concept of well-posedness is very general and in fact very simple:

Definition 6 (Hadamard 1923)

1. The problem under consideration has a solution;
2. This solution is unique;
3. The solution depends continuously on the problem data.

The first condition is immediately clear. The second condition is also obvious but often difficult to meet - and in fact many physical processes do not have unique solutions. The last condition says if a variation of the input data (right hand side, boundary values, initial conditions) vary only a little bit, then also the (unique) solution should only vary a bit.

Remark 1

Problems in which one of the three conditions is violated are **ill-posed**.

Well-posedness: existence, uniqueness, stability

Definition 7 (Lipschitz condition)

The function $f(t, y)$ on D is said to be (uniformly) Lipschitz continuous if for $L(t) > 0$ it holds

$$\|f(t, x_1) - f(t, x_2)\| \leq L(t)\|x_1 - x_2\|, \quad (t, x_1), (t, x_2) \in D.$$

The function is said to be (locally) Lipschitz continuous if the previous statement holds on every bounded subset of D.

Theorem 8 (Picard-Lindelöf)

Let $f : D \to \mathbb{R}^n$ be continuous and Lipschitz. Then there exists for each $(t_0, y_0) \in D$ a $\varepsilon > 0$ and a solution $y : I := [t_0 - \varepsilon, t_0 + \varepsilon] \to \mathbb{R}^n$ of the IVP such that

$$y'(t) = f(t, y(t)), \quad t \in I, \quad y(t_0) = y_0.$$

The proof can be found in classical textbooks on ordinary differential equations.

What is the model problem good for?

- Model problem is a simplified problem to predict growth of a population: human beings, animals, bacteria, virus
  (see also lecture 01)
- Example:
$$y' = (g - m)y, \quad y(t_0) = y_0$$
  with growth g and mortalities rates m
- Exact solution (here possible):
$$y(t) = c \exp((g - m)(t - t_0)).$$
  with
$$y(t_0) = \exp(C) \exp[(g - m)(t_0 - t_0)] = \exp(C) = y_0 =: c.$$

What is the model problem good for?

- Let us say in the year $t_0 = 2011$ there have been two members of this species: $y(2011) = 2$. Supposing a growth rate of 25 per cent per year yields $g = 0.25$. Let us say $m = 0$ - nobody will die.

What is the model problem good for?

- Let us say in the year $t_0 = 2011$ there have been two members of this species: $y(2011) = 2$. Supposing a growth rate of 25 per cent per year yields $g = 0.25$. Let us say $m = 0$ - nobody will die.
- In the following we compute two estimates of the future evolution of this species: for the year $t = 2014$ and $t = 2022$. We first obtain:

$$y(2014) = 2\exp(0.25 * (2014 - 2011)) = 4.117 \approx 4.$$

What is the model problem good for?

- Let us say in the year $t_0 = 2011$ there have been two members of this species: $y(2011) = 2$. Supposing a growth rate of 25 per cent per year yields $g = 0.25$. Let us say $m = 0$ - nobody will die.
- In the following we compute two estimates of the future evolution of this species: for the year $t = 2014$ and $t = 2022$. We first obtain:

$$y(2014) = 2\exp(0.25 * (2014 - 2011)) = 4.117 \approx 4.$$

- Thus, four members of this species exist after three years. Secondly, we want to give a 'long term' estimate for the year $t = 2022$ and calculate:

$$y(2022) = 2\exp(0.25 * (2022 - 2011)) = 31.285 \approx 31.$$

What is the model problem good for?

- Let us say in the year $t_0 = 2011$ there have been two members of this species: $y(2011) = 2$. Supposing a growth rate of 25 per cent per year yields $g = 0.25$. Let us say $m = 0$ - nobody will die.
- In the following we compute two estimates of the future evolution of this species: for the year $t = 2014$ and $t = 2022$. We first obtain:

$$y(2014) = 2\exp(0.25 * (2014 - 2011)) = 4.117 \approx 4.$$

- Thus, four members of this species exist after three years. Secondly, we want to give a 'long term' estimate for the year $t = 2022$ and calculate:

$$y(2022) = 2\exp(0.25 * (2022 - 2011)) = 31.285 \approx 31.$$

- In fact, this species has an increase of 29 members within 11 years.

What is the model problem good for?

- Let us say in the year $t_0 = 2011$ there have been two members of this species: $y(2011) = 2$. Supposing a growth rate of 25 per cent per year yields $g = 0.25$. Let us say $m = 0$ - nobody will die.
- In the following we compute two estimates of the future evolution of this species: for the year $t = 2014$ and $t = 2022$. We first obtain:

$$y(2014) = 2\exp(0.25 * (2014 - 2011)) = 4.117 \approx 4.$$

- Thus, four members of this species exist after three years. Secondly, we want to give a 'long term' estimate for the year $t = 2022$ and calculate:

$$y(2022) = 2\exp(0.25 * (2022 - 2011)) = 31.285 \approx 31.$$

- In fact, this species has an increase of 29 members within 11 years.
- Translating this to human beings, we observe that the formula works quite well for a short time range but becomes somewhat unrealistic for long-term estimates though.
- Solution: construct a better mathematical model! For instance the logistic law (see e.g., M. Braun; Differential equations and their applications, 1993)

Solving differential equations: numerical discretization

- Analytical solutions are often too difficult or even impossible

$\rightarrow$ Why?

Solving differential equations: numerical discretization

- Analytical solutions are often too difficult or even impossible

→ Why? Too expensive, too far away (moon, planets, ...), too smale (nanoscale)

- **Numerical discretization**

→ Treat infinite-dimensional problems with the help of finite-dimensional discretizations

- Simple example: cut numbers! For instance:
  $x = 3.14456456456089823450020340984309 86...$
  $\tilde{x} = 3.14456456456089$

→ Infinite number is x and finite number is $\tilde{x}$.

→ What is the error between $x - \tilde{x}$?

Solving differential equations: numerical discretization

- **Discretization parameter** often denoted by k or Δt (for temporal discretization) and h for spatial discretization.

$\rightarrow$ Represent a DE with finite numbers! And not infinite numbers!

$\rightarrow$ Allows to solve DE with a computer!

- Paradigm: Design numerical schemes in such a way that physical conservation properties (mass, momentum, energy, ...) are as much as possible conserved after the discretization!

Guiding questions for numerics of DE (1)

- What type of equation are we dealing with?
- What kind of discretization scheme shall we use?
- How do we design algorithms to compute u_h (or y_h - notation for ODEs)?
- Can we proof that these algorithms really work?
- Are they robust (stable with respect to parameter variations), accurate, and efficient?
- Can we construct physics-based algorithms that maintain as best as possible conservation laws; in particular when several equations interact (couple)?

Guiding questions for numerics of DE (2)

- How far is u_h away from u in a certain (error) norm? Hint: comparing color figures gives a first impression, but is not science!
- The discretized systems (to obtain u_h) are often large with a huge number of unknowns: how do we solve these linear equation systems?
- What is the computational cost?
- How can we achieve more efficient algorithms? Hint: adaptivity and/or parallel computing.
- How can we check that the solution is correct?

Errors: related to numerics and programming

1. The set of numbers is finite and a calculation is limited by machine precision (floating point arithmetics), which results in **round-off errors**. Typical issues are overflow and underflow of numbers.

Errors: related to numerics and programming

1. The set of numbers is finite and a calculation is limited by machine precision (floating point arithmetics), which results in **round-off errors**. Typical issues are overflow and underflow of numbers.
2. The memory of a computer (or cluster) is finite and thus functions and equations can only be represented through approximations. Thus, continuous information has to be represented through discrete information, which results into the investigation of so-called **discretization errors**.

Errors: related to numerics and programming

1. The set of numbers is finite and a calculation is limited by machine precision (floating point arithmetics), which results in **round-off errors**. Typical issues are overflow and underflow of numbers.
2. The memory of a computer (or cluster) is finite and thus functions and equations can only be represented through approximations. Thus, continuous information has to be represented through discrete information, which results into the investigation of so-called **discretization errors**.
3. All further simplifications of a numerical algorithm (in order to solve the discrete problem), with the final goal to reduce the computational time, are so-called **systematic errors**. Mostly, these are so-called **iteration errors**, for instance the stopping criterion after how many steps an iterative method terminates.

Errors: related to numerics and programming

1. The set of numbers is finite and a calculation is limited by machine precision (floating point arithmetics), which results in **round-off errors**. Typical issues are overflow and underflow of numbers.
2. The memory of a computer (or cluster) is finite and thus functions and equations can only be represented through approximations. Thus, continuous information has to be represented through discrete information, which results into the investigation of so-called **discretization errors**.
3. All further simplifications of a numerical algorithm (in order to solve the discrete problem), with the final goal to reduce the computational time, are so-called **systematic errors**. Mostly, these are so-called **iteration errors**, for instance the stopping criterion after how many steps an iterative method terminates.
4. Finally, **programming errors (code bugs)** are an important error source. Often these can be identified since the output is strange. But there are many, which are hidden and very tedious to detect.

Errors: related to modeling (will not be discussed in this school)

1. In order to make a 'quick guess' of a possible solution and to start the development of an algorithm to address at a later stage a difficult problem, often complicated (nonlinear) differential equations are reduced to simple (in most cases linear) versions, which results in the so-called **model error**.
2. **Data errors**: the data (e.g., input data, boundary conditions, parameters) are finally obtained from experimental data and may be inaccurate themselves.

Errors: final statements

- It very important to understand that we **never can avoid** all these errors.
- The important aspect is to **control** these errors and to provide answers if these errors are sufficiently big to influence the interpretation of numerical simulations or if they can be assumed to be small.
- A big branch of numerical mathematics is to derive error estimates that allow to predict about the size of arising errors.

Numerical concepts[1]

1. **Approximation**: since analytical solutions are not possible to achieve as we just learned in the previous section, solutions are obtained by **numerical approximations**.

Numerical concepts[1]

1. **Approximation**: since analytical solutions are not possible to achieve as we just learned in the previous section, solutions are obtained by **numerical approximations**.
2. **Convergence**: is a qualitative expression that tells us when members a_n of a sequence $(a_n)_{n\in\mathbb{N}}$ are sufficiently close to a limit a. In numerical mathematics this limit is often the solution that we are looking for.

Numerical concepts[1]

1. **Approximation**: since analytical solutions are not possible to achieve as we just learned in the previous section, solutions are obtained by **numerical approximations**.
2. **Convergence**: is a qualitative expression that tells us when members a_n of a sequence $(a_n)_{n\in\mathbb{N}}$ are sufficiently close to a limit a. In numerical mathematics this limit is often the solution that we are looking for.
3. **Order of convergence**: While in analysis, we are often interested in the convergence itself, in numerical mathematics we must pay attention how long it takes until a numerical solution has sufficient accuracy. The longer a simulation takes, the more time and more energy (electricity to run the computer, air conditioning of servers, etc.) are consumed. In order to judge whether a algorithm is fast or not we have to determine the order of convergence.

Numerical concepts

4 **Errors**: Numerical mathematics can be considered as the branch 'mathematics of errors'. What does this mean? Numerical modeling is not wrong, inexact or non-precise! Since we cut sequences after a final number of steps or accept sufficiently accurate solutions obtained from our software, we need to say *how well the (unknown) exact solution by this numerical solution is approximated.* In other words, we need to determine the error, which can arise in various forms as we discussed in the previous section.

Numerical concepts

4 **Errors**: Numerical mathematics can be considered as the branch 'mathematics of errors'. What does this mean? Numerical modeling is not wrong, inexact or non-precise! Since we cut sequences after a final number of steps or accept sufficiently accurate solutions obtained from our software, we need to say *how well the (unknown) exact solution by this numerical solution is approximated.* In other words, we need to determine the error, which can arise in various forms as we discussed in the previous section.

5 **Error estimation**: This is one of the biggest branches in numerical mathematics. We need to derive error formulae to judge the outcome of our numerical simulations and to measure the difference of the numerical solution and the (unknown) exact solution in a certain norm.

Numerical concepts

6 **Efficiency**: In general we can say, the higher the convergence order of an algorithm is, the more efficient our algorithm is. But numerical efficiency is not automatically related to resource-effective computing. For instance, developing a parallel code using MPI (message passing interface) will definitely yield in less CPU (central processing unit) time a numerical solution. However, whether a parallel machine does need less electricity (and thus less money) than a sequential desktop machine/code is a priori unclear.

Numerical concepts

6 **Efficiency**: In general we can say, the higher the convergence order of an algorithm is, the more efficient our algorithm is. But numerical efficiency is not automatically related to resource-effective computing. For instance, developing a parallel code using MPI (message passing interface) will definitely yield in less CPU (central processing unit) time a numerical solution. However, whether a parallel machine does need less electricity (and thus less money) than a sequential desktop machine/code is a priori unclear.

7 **Stability**: Despite being the last concept, in most developments, this is the very first step to check. How robust is our algorithm against different model and physical parameters? Is the algorithm stable with respect to different input data? This condition relates in the broadest sense to the third condition of Hadamard. In practice non-robust or non-stable algorithms exhibit very often non-physical oscillations. For this reason, it is important to have a feeling about the physics whether oscillations in the solution are to be expected or if they are introduced by the numerical algorithm.

Summary lecture 02

- DE (differential equations) - classifications, examples
- General numerical concepts
- One goal of this spring school is to learn numerical techniques and corresponding programming in order to analyze and implement DE.

Exercise 1 Overview

In this exercise we explore the pendulum in more detail by

- Recapitulating the derivation of the full model and the simplified model
- Deriving two numerical methods for its solution
- Implementing these methods in your own C++
- Evaluating these methods by
  - Looking at stability,
  - Discretization error and
  - Modeling error

Task 1

- Recapitulate the model for the pendulum

$$\frac{d^2\phi(t)}{dt^2} = -\frac{g}{\ell}\sin(\phi(t)) \qquad \forall t > t_0.$$

  with the two initial conditions

$$\phi(0) = \phi_0, \qquad \frac{d\phi}{dt}(0) = \phi_0'.$$

- For *small* deflection angle ϕ derive the *approximation*

$$\frac{d^2\phi(t)}{dt^2} = -\frac{g}{\ell}\phi(t)$$

- Show that it has the general solution $\phi(t) = A\cos(\omega t)$ and determine the constants A, ω from the initial conditions

Full Model, Method 1

- In the first method, begin by rewriting the second order ODE as a first order system

$$\frac{d\phi(t)}{dt} = u(t), \qquad \frac{d^2\phi(t)}{dt^2} = \frac{du(t)}{dt} = -\frac{g}{\ell}\sin(\phi(t)).$$

- Replacing derivatives by difference quotients

$$\frac{\phi(t+\Delta t)-\phi(t)}{\Delta t} \approx \frac{d\phi(t)}{dt} = u(t),$$
$$\frac{u(t+\Delta t)-u(t)}{\Delta t} \approx \frac{du(t)}{dt} = -\frac{g}{\ell}\sin(\phi(t)).$$

- yields the *one step* scheme

$$\phi^{n+1} = \phi^n + \Delta t\, u^n \qquad \phi^0 = \phi_0$$
$$u^{n+1} = u^n - \Delta t\,(g/\ell)\sin(\phi^n) \qquad u^0 = u_0$$

Where ϕ^n approximates $\phi(n\Delta t)$ for a chosen Δt Rekursion (*Euler*):

Full Model, Method 2

- Now we derive a method that directly approximates the second-order ODE
- It uses a *central difference quotient* for the second derivative

$$\frac{\phi(t+\Delta t) - 2\phi(t) + \phi(t-\Delta t)}{\Delta t^2} \approx \frac{d^2\phi(t)}{dt^2} = -\frac{g}{\ell}\sin(\phi(t)).$$

- Solving for $\phi(t+\Delta t)$ yields the *two step* scheme ($n \geq 2$):

$$\phi^{n+1} = 2\phi^n - \phi^{n-1} - \Delta t^2\,(g/\ell)\,\sin(\phi^n) \tag{9}$$

with the initial condition

$$\phi^0 = \phi_0, \qquad \phi^1 = \phi_0 + \Delta t\, u_0. \tag{10}$$

The starting value ϕ^1 is derived with method 1

Task 2

- Write a C++ program implementing schemes 1 and 2 using a time step Δt that can be entered by the user
- Write the results to a file, where every line contains

$$t_i \qquad \phi^i \qquad u^i$$

- you can visualize the results using `gnuplot` as follows
  `plot "filename"u 1:2`
  where the x-axis uses the first column and the y-axis uses the second column
- You may start from the file `eemodelproblem.cc` available on the cloud `https://cloud.ifam.uni-hannover.de/index.php/s/Cwe4ZqwLRMixS3J`. It solves the problem $u' = \lambda u$ using `hdnum`
- Download the file and put it in the directory `hdnum/examples/num1`. Compile it with
  `g++ -o eemodelproblem -I../.. eemodelproblem.cc`

Task 3: Comparisons

- For method 1: choose an initial deflection angle $\phi_0 = 0.1$ and a time step $\Delta = 0.1$ and compute the solution up to time 4.0. What do you observe?
- Repeat the experiment with successively smaller time steps, say 0.01, 0.001, 0.0001. What do you observe?
- Try to compute the solution for longer times with the small timesteps. What happens?
- Repeat the same experiments with method 2. Is there a difference?
- Compare the solution of the full model and the reduced model for different initial angles $\phi_0 = 0.1, 0.5, 3.0$. Use your favourite method and a timestep Δt that is small enough to avoid any visibly numerical error.
- Recapitulate the concepts stability, discretization error and modeling error in the light of the results of exercise 1.