SF2520 — Applied numerical methods

Lecture 1

Course introduction, ODEs

Olof Runborg
Numerical analysis
Department of Mathematics, KTH

2023-08-28

Today's lecture

- Course introduction
  - Contents
  - Structure
- Ordinary differential equations
  - Some theory review
  - Numerical methods, introduction

Course contents — Scientific computing

- Mathematical models = mathematical representation of real life systems/phenomena
- Differential equations (DE) are very common models:

$$y' = f(t, y), \qquad \Delta u = f, \qquad u_{tt} = \nabla \cdot (A \nabla u).$$

- Difficult or impossible to analyze/solve by hand, in general.
- Use computers and numerical methods to solve models — to simulate reality.
- Complement to theory and experiments in the scientific method:
  - More control and more details
  - Larger scope: large, small, dangerous, expensive, slow, fast problems.

Course contents, cont.

The course is about numerical methods for

- Ordinary differential equations (ODE) — period 1
  - Mechanical systems
  - Molecular dynamics
  - Chemical reactions
  - Electrical networks
  - Population dynamics
  - Approximation of PDE
  - ...
- Partial differential equations (PDE) — period 1+2
  - Heat equation
  - Wave equation
  - Poisson equation
  - Maxwell equations
  - Navier–Stokes equations
  - Schrödinger equation
  - ...
- Linear algebra — period 2

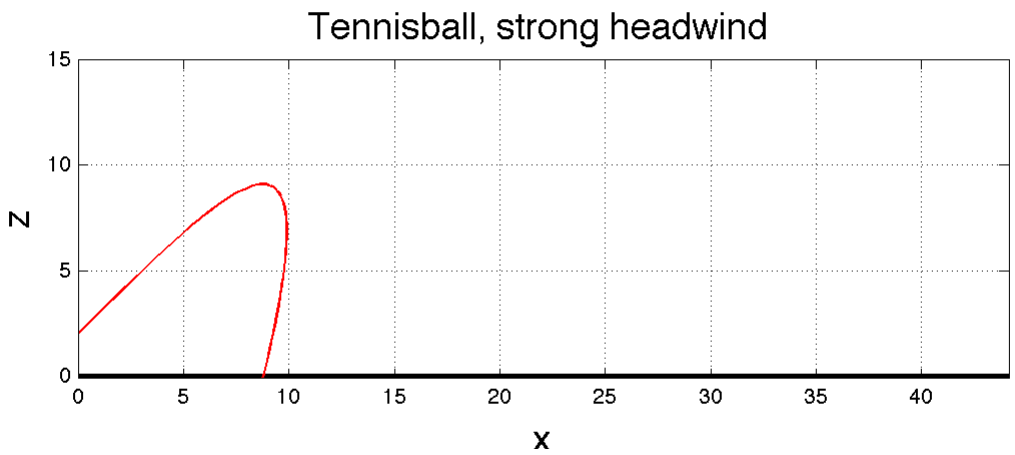
ODE example

Throwing a ball

ODE derived from Newton's second law:

$$m\frac{d^2\mathbf{r}}{dt^2} = -mg\mathbf{e}_z - c\left\|\frac{d\mathbf{r}}{dt} - \mathbf{w}\right\|\left(\frac{d\mathbf{r}}{dt} - \mathbf{w}\right), \qquad \mathbf{r}(0) = \mathbf{r}_0, \quad \frac{d\mathbf{r}(0)}{dt} = \mathbf{v}_0.$$

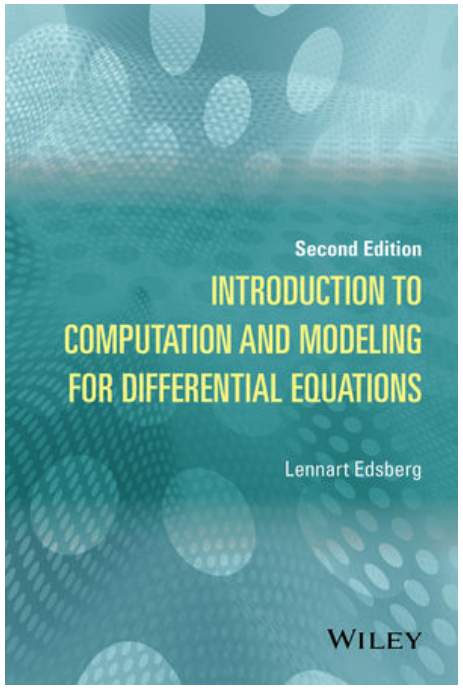
- $\mathbf{r}(t)$ – position at time t
- m – mass
- g – gravitational acceleration
- c – air resistance
- $\mathbf{w}$ – wind velocity
- $\mathbf{r}_0, \mathbf{v}_0$ – initial position and velocity



Course overview

- Structure
  - Lectures (9 + 7 in period 1/2)
  - Exercise lectures (2 + 2 in period 1/2)
  - Computer exercises (2.5 + 2.5 in period 1/2)
  - Project (period 2)
- Teachers
  - Olof Runborg (lectures period 1, examiner)
  - Anna Nissen (lectures period 2)
  - David Krantz (lab assistant, exercise lectures)
  - Sebastian Myrbäck (—"—)
- Literature
  - Lennart Edsberg, *Introduction to computation and modeling for differential equations*, 2nd ed
  - Matlab description (optional)
  - Some notes on various topics (PDF)
  - Numerical linear algebra material

    (PDF excerpts from two books)



Computer exercises

- Five exercises in total + refresher on ODE
- Implement/analyze methods using Matlab (and COMSOL for CE2).
- Work in groups of two students. (Can use Canvas forum to find people.)
  Sign up on Canvas to groups "SF2520-CE1 YYY".
- Mainly work on your own, but help sessions in computer lab scheduled ca once per week (not mandatory). (See Canvas for details.)
- Software available in computer labs, and for download via KTH. (Getting a license for COMSOL can take some time. Do it early!)
- Examination:
  - Exercise 1, 2: Written report + Matlab code
  - Exercise 3: Test in computer lab
  - Exercise 4, 5: Oral presentation in computer lab
  - Submit all material online (Matlab code and report) before deadline.
    First deadline Sep 18.
- Each exercise gives points that count towards course grade
- Max 26 points total

Computer exercises — group work

- Each group should solve the problems independently.
- You can discuss issues with other groups, but:
  - All code and reports must be written by yourself, from scratch (and e.g. not based on a stub given to you).
  - It is strictly forbidden to copy code from or to share code files with another group.
- Both group members should contribute equally to the work and understand all parts of the submitted solution.

KTH honor code

- *All members of a group are responsible for the group's work.*
- *In any assessment, every student shall honestly disclose any help received and sources used.*
- *Every student shall be able to present and answer questions about the entire assignment and solution.*

Project

- Larger exercises which require knowledge from several areas.
- Work in teams of 4 students.
- Teams are formed randomly.
- Oral presentation in December
- Slides from presentation and Matlab code submitted afterwards.

Examination

There are three parts of the examination. Each must be passed to finish the course:

- Computer exercises
  At least 21 points (out of 26), with at least 50% of max points on each exercise
- Project
  Oral presentation (and slides +Matlab code), pass/fail
- Exam 2023-01-12
  At least 13 points (out of 29).

Final grade determined by the sum of the points from parts one and three (max 26+29 = 55 points):

E	D	C	B	A
34-37	38-41	42-45	46-49	50-55

What advice would you give to future participants?

- Work hard!
- Try to really understand the labs, it will be worth when you study for the exam.
- Follow the **lectures** and start with the homework assignments early. Make sure to at least have looked through the whole assignment before the help sessions.
- Find a lab partner and start early on the labs. The report writing takes a long time.
- The assignments are interesting, provided you start in time. Then they really help to sum up what you've learned before
- Begin the assignments as soon as possible and don't hesitate to ask questions at the lab sessions Numerical stability is very important. This is at least in part why we choose different methods for different types of PDE problems (hyperbolic, parabolic, elliptic).
- Go to the **lectures** — the concepts are really easy to understand if you see the steps taken to get there. Go through your notes shortly after — see if there are any mistakes while you still remember the **lecture**, and if you have something that you didn't understand, now is the time to clear it.
- The workload for the computer exercises is alot. Start as soon as you get the assignments! Follow the **lectures**, you use them alot for the lab prep.
- Spend time doing the Computer exercises and make sure you understand them. Discuss with others and go to the **lectures**.
- Work in pairs! Actually make the effort to do the assignments regularly (which is kind of mandatory giving the grading system).
- Do the labs seriously and try to do some calculations with pen-paper during the course.
- Start the homeworks early and make sure you understand how to solve them as it will help a lot for the exams and it is nicer to hand them in early than stress in the last second.
- Keep up with the reading!

All info about the course is in Canvas.

Canvas announcements used for communication.

Make sure you are registered to the course
to access all Canvas features.

Contact the student office to register:

studentoffice@math.kth.se

Monitor your email that is registered in Canvas.

Ordinary differential equations (ODE)

- We consider the initial value problem (IVP) for the (first order) ODE

$$\frac{dy(t)}{dt} = f(t, y(t)), \qquad y(0) = y_0 \in \mathbb{R}.$$

- The unknown here is the scalar function $y(t)$. The initial value y_0 and the function f are given.
- More generally, we have higher order ODEs, e.g. p-th order ODE:

$$\frac{d^p y}{dt^p} = f(t, y, y', y'', \ldots, y^{(p-1)}),$$
$$y(0) = y_0, \quad y'(0) = y_1, \quad \ldots, \quad y^{(p-1)}(0) = y_{p-1},$$

- . . . and/or a system of ODEs

$$\frac{d\boldsymbol{y}(t)}{dt} = \boldsymbol{F}(t, \boldsymbol{y}(t)), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0 \in \mathbb{R}^d,$$

where $\boldsymbol{y} : \mathbb{R} \mapsto \mathbb{R}^d$ and $\boldsymbol{F} : \mathbb{R}^{d+1} \mapsto \mathbb{R}^d$ are vector valued functions.

Ordinary differential equations, cont.

- Many processes in nature and elsewhere are well described by ODEs (mechanical systems, chemical reactions, molecular dynamics, electrical circuits, infections and spreading of diseases, population dynamics, ...)
- Typically, when $y'(t) = f(y(t))$,

  $y(t) =$ some physical quantity (position, velocity, temperature, ...),

  $t =$ time,

  $f =$ physical law stating how y changes in time.

Example: Newton's second law

$$y'' = \frac{1}{m} f(y), \qquad y\text{=position, } f\text{=force, } m\text{=mass}$$

- Some (simple) ODEs can be solved analytically, e.g. scalar linear ODEs

$$y'(t) = a(t)y(t) \quad \Rightarrow \quad y(t) = y_0 e^{\int_0^t a(s)ds}.$$

  However, in general numerical methods are needed.

Ordinary differential equations, cont.

We focus on the initial value problem for systems of ODEs

$$\frac{d\boldsymbol{y}(t)}{dt} = \boldsymbol{F}(t, \boldsymbol{y}(t)), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0 \in \mathbb{R}^d.$$

- Dimension d can be large.
- Need initial data for each component (d data).
- Note: This form includes higher order ODEs upon rewriting them as a system for $\boldsymbol{u}(t) := [y(t),\ y'(t),\ \ldots,\ y^{(p-1)}(t)]^T$.

Example

The second order scalar ODE

$$y'' = f(t, y, y'), \qquad y(0) = y_0, \quad y'(0) = y_1,$$

can, with $\boldsymbol{u} = (y, y')^T =: (u_0, u_1)^T$, be written as the system

$$\frac{d\boldsymbol{u}}{dt} = \begin{pmatrix} u_1 \\ f(t, u_0, u_1) \end{pmatrix} =: \boldsymbol{F}(t, \boldsymbol{u}), \qquad \boldsymbol{u}(0) = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}.$$

Ordinary differential equations, cont.

Closely related equations

- Two-point boundary value problem (BVP)

$$\frac{d^2y(x)}{dx^2} = f(t, y(x)), \qquad y(0) = \alpha, \qquad y(1) = \beta.$$

(y given in two different points, instead of y, y' in the same point.)

- Differential algebraic systems (DAE)

$$\begin{aligned} \frac{d\boldsymbol{y}}{dt} &= \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{z}), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0, \\ 0 &= \boldsymbol{G}(\boldsymbol{y}, \boldsymbol{z}), \end{aligned}$$

with $\boldsymbol{z}_0 = \boldsymbol{z}(0)$ solving $\boldsymbol{G}(\boldsymbol{y}_0, \boldsymbol{z}_0) = 0$.

(ODE coupled to an algebraic equation.)

Properties of ODEs

- **Existence and uniqueness**

  The ODE $$\frac{d\boldsymbol{y}(t)}{dt} = \boldsymbol{F}(t, \boldsymbol{y}(t)), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0 \in \mathbb{R}^d,$$

  has a unique solution $\boldsymbol{y} \in C^1$ at least in some finite time interval $[0, T]$ if the function $\boldsymbol{F}$ is
  - continuous with respect to t, and
  - Lipschitz continuous with respect to $\boldsymbol{y}$ (satisfied e.g. if $\frac{\partial \mathbf{F}}{\partial \mathbf{y}}$ exists),
  - in a neighborhood of $(t = 0, \boldsymbol{y} = \boldsymbol{y}_0)$.
- This means that in most practical situations a unique solution exists (since $\boldsymbol{F}$ typically has the required regularity)...
- ...but typically *only for finite time*.

Example

The scalar ODE $y' = y^2$ with $y(0) = y_0 > 0$ has the solution

$$y(t) = \frac{y_0}{1 - y_0 t}$$

which blows up when $t \to 1/y_0$. (I.e. existence only for $t < t_0$.)

Properties of linear ODEs

- Linear systems of ODEs (called LCC in the book if A constant)

$$\frac{d\boldsymbol{y}(t)}{dt} = A\boldsymbol{y}(t), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0 \in \mathbb{R}^d, \qquad A \in \mathbb{R}^{d\times d},$$

  has a unique solution globally in time (!).
- If A can be *diagonalized* (has full set of eigenvectors)

$$\boldsymbol{y}(t) = \alpha_1 \boldsymbol{v}_1 e^{\lambda_1 t} + \alpha_2 \boldsymbol{v}_2 e^{\lambda_2 t} + \cdots + \alpha_d \boldsymbol{v}_d e^{\lambda_d t},$$

  where $(\boldsymbol{v}_k, \lambda_k)$ are the eigenvectors/-values of A and $\{\alpha_j\}$ are given from initial data by solving the linear system

$$\alpha_1 \boldsymbol{v}_1 + \alpha_2 \boldsymbol{v}_2 + \cdots + \alpha_d \boldsymbol{v}_d = \boldsymbol{y}_0.$$

- For general A the solution is given by

$$\boldsymbol{y}(t) = e^{At}\boldsymbol{y}_0, \qquad e^A := \sum_{k=0}^{\infty} \frac{A^k}{k!} \in \mathbb{R}^{d\times d},$$

  where e^{At} is the *matrix exponential*.

  (Can be computed with the `expm(A)` command in Matlab.)

Properties of linear ODEs, cont.

- Note: In general eigenvalues λ_k are complex even if A is real. This gives oscillatory solutions

$$\lambda = a + i\theta \quad \Rightarrow \quad y \sim e^{at+it\theta}.$$

- Solutions of the inhomogeneous linear problem

$$\frac{d\boldsymbol{y}(t)}{dt} = A\boldsymbol{y}(t) + \boldsymbol{g}(t), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0 \in \mathbb{R}^d, \qquad A \in \mathbb{R}^{d\times d},$$

is given by "Duhamel's principle" as

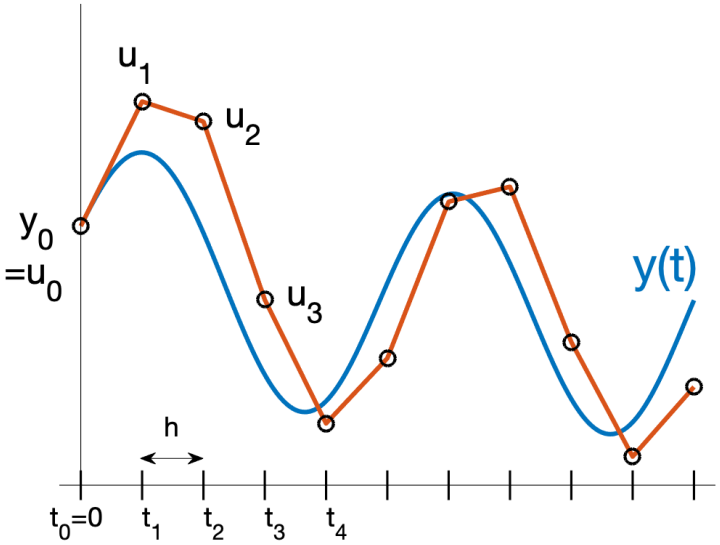
$$\boldsymbol{y}(t) = e^{At}\boldsymbol{y}_0 + \int_0^t e^{A(t-s)}\boldsymbol{g}(s)ds.$$

Numerical methods for ODEs

Want to solve IVP numerically:

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{F}(t, \boldsymbol{y}), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0 \in \mathbb{R}^d,$$

(Find $\boldsymbol{y}$ when $\boldsymbol{y}_0$ and $\boldsymbol{F}$ given.)



General approach:

0. **Rewrite as a first order system**
   (if the ODE is of higher order, as e.g. $\boldsymbol{y}'' = \boldsymbol{F}(\boldsymbol{y})$)
1. **Discretize**
   Introduce discrete points in time $t_n = nh$ where $h \ll 1$ is a small step length, and approximate $\boldsymbol{u}_n \approx \boldsymbol{y}(t_n)$.
2. **Iterate** *("time step")*
   Use recursion where $\boldsymbol{u}_{n+1}$ computed from $\boldsymbol{u}_n$ (and sometimes $\boldsymbol{u}_{n-1}, \dots$)
   Ex: Explicit Euler

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h\boldsymbol{F}(t_n, \boldsymbol{u}_n), \qquad \boldsymbol{u}_0 = \boldsymbol{y}_0.$$

Examples of numerical ODE methods

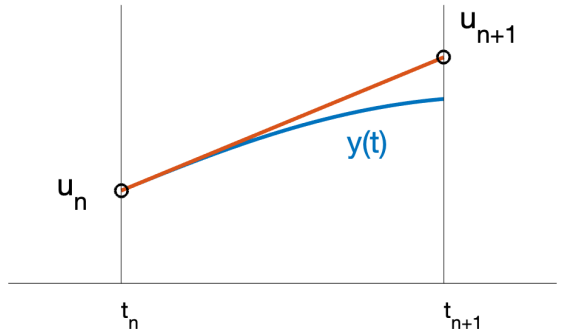
- **Explicit Euler (Forward Euler)**

$$u_{n+1} = u_n + hf(t_n, u_n),$$

Given by approximating y'

$$y'(t_n) = f(t_n, y(t_n)) \approx \frac{y(t_{n+1}) - y(t_n)}{h},$$

i.e. by a forward difference.



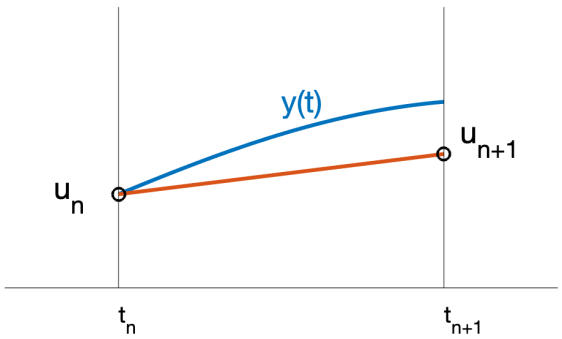
- **Implicit Euler (Backward Euler)**

$$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}),$$

Given by approximating y'

$$y'(t_n) = f(t_n, y(t_n)) \approx \frac{y(t_n) - y(t_{n-1})}{h},$$

i.e. by a backward difference.



Examples of numerical ODE methods

- **Trapezoidal method**

$$u_{n+1} = u_n + \frac{1}{2}h\Big(f(t_n, u_n) + f(t_{n+1}, u_{n+1})\Big)$$

Given by taking the average of Explicit and Implicit Euler.

- **Heun's method (Runge–Kutta 2)**

$$u_{n+1} = u_n + \frac{1}{2}h\Big(f(t_n, u_n) + f(t_{n+1}, u_n + hf(t_n, u_n))\Big)$$

Given by replacing u_{n+1} in the Trapezoidal method by one step of Explicit Euler.

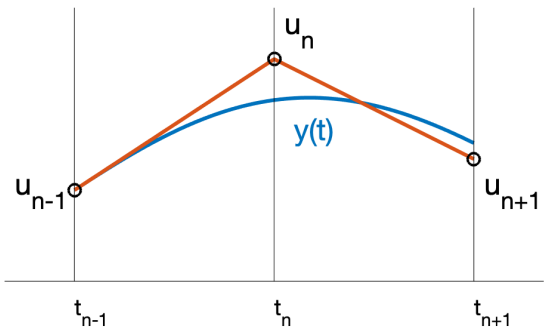
- **Midpoint method**

$$u_{n+1} = u_{n-1} + 2hf(t_n, u_n),$$

Given by approximating y'

$$y'(t_n) = f(t_n, y(t_n)) \approx \frac{y(t_{n+1}) - y(t_{n-1})}{2h},$$

i.e. by a central difference.



ODE methods — classification

1. Explicit Euler, $u_{n+1} = u_n + hf(t_n, u_n)$,
2. Implicit Euler, $u_{n+1} = u_n + hf(t_{n+1}, u_{n+1})$,
3. Trapezoidal method, $u_{n+1} = u_n + \frac{1}{2}h\Big(f(t_n, u_n) + f(t_{n+1}, u_{n+1})\Big)$
4. Heun's method, $u_{n+1} = u_n + \frac{1}{2}h\Big(f(t_n, u_n) + f(t_{n+1}, u_n + hf(t_n, u_n))\Big)$
5. Midpoint method, $u_{n+1} = u_{n-1} + 2hf(t_n, u_n)$,

- (For systems $\boldsymbol{y}' = \boldsymbol{F}(t, \boldsymbol{y})$, just replace $u \to \boldsymbol{u}$ and $f \to \boldsymbol{F}$ above.)
- **Explicit (1,4,5) vs Implicit (2,3)**
  A method is *explicit* if u_{n+1} is not an argument of f anywhere in the method. Otherwise it is an *implicit* method (and a nonlinear equation must typically be solved in each step!)
- **One-step (1,2,3,4) vs Multi-step (5 with $q = 2$)**
  A *one-step* method only uses u_n to compute u_{n+1}. A *multi-step* (q-step) method uses u_n, u_{n-1}, ..., u_{n-q+1}.

ODE – implicit methods

- In an implicit method u_{n+1} is an argument of f, and an equation (in general nonlinear) must be solved in each step.
- **Example:** Implicit Euler for linear system $\boldsymbol{y}' = A\boldsymbol{y} + \boldsymbol{g}(t)$

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h\Big[A\boldsymbol{u}_{n+1} + \boldsymbol{g}(t_{n+1})\Big] \quad \Rightarrow \quad (I - hA)\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h\boldsymbol{g}(t_{n+1})$$

Solve linear system of eqs. in each step. (I is the identity matrix.)

- **Example:** Implicit Euler for general nonlinear ODE $\boldsymbol{y}' = \boldsymbol{F}(t, \boldsymbol{y})$

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h\boldsymbol{F}(t_{n+1}, \boldsymbol{u}_{n+1})$$

Solve $G(\boldsymbol{u}) = 0$ where

$$G(\boldsymbol{u}) = \boldsymbol{u} - h\boldsymbol{F}(t_{n+1}, \boldsymbol{u}) - \boldsymbol{u}_n.$$

Set $\boldsymbol{u}_{n+1} = \boldsymbol{u}^*$ = solution.

- Use e.g. Newton's method to solve $G(\boldsymbol{u}) = 0$, with start value $\boldsymbol{u}_n$.

Gives two levels of iterations: time stepping (outer) and Newton (inner).

One-step methods

The general form of a one-step method is

$$u_{n+1} = u_n + h\phi(h, t_n, u_n, u_{n+1}), \qquad u_0 = y_0,$$

where ϕ depends on f.

- If ϕ does not depend on u_{n+1} the method is explicit.
- Examples:
  - **Explicit Euler**:
  $$\phi(h, t_n, u_n, u_{n+1}) = f(t_n, u_n)$$
  - **Implicit Euler**:
  $$\phi(h, t_n, u_n, u_{n+1}) = f(t_n + h, u_{n+1})$$
  - **Heun's method**:
  $$\phi(h, t_n, u_n, u_{n+1}) = \frac{1}{2}\Big(f(t_n, u_n) + f(t_n + h, u_n + hf(t_n, u_n))\Big)$$

Numerical errors

- Introduce the (global) error e_n in step n

$$e_n := u_n - y(t_n).$$

- For convergence we want $\lim_{h\to 0} e_n \to 0$. (Note: e_n, u_n and t_n depend on h.)
- More precisely: Consider the solution in a fixed interval $t \in [0, T]$. If we use $h = T/N$, i.e. N time steps, then we want to bound

$$\text{maximum global error} = \max_{0\le n\le N} |e_n| \le Ch^p,$$

such that C does not depend on h and $p \ge 1$.

(Note: N, e_n and t_n depend on h.)

- When this holds the method has order of accuracy p. Higher p means faster convergence.

(Order of accuracy is a central general concept. See notes if you need to catch up on this.)

- Error estimates typically also hold pointwise for the methods:

$$e_n \approx C(t_n)h^p,$$

where C depends on the (fixed) time $t_n = nh$.

Numerical errors – examples

Example

We solve the ODE

$$y' = \sin(y) - y^2 + \cos(2\pi t), \qquad y(0) = 1,$$

in the interval $t \in [0, 2]$ using Explicit Euler and the Trapezoidal method with time steps $h = 0.2, 0.1, 0.05, 0.025,$ i.e. N=10, 20, 40, 80.

- Matlab examples.
- Empirically we get order of accuracy $p = 1$ for Explicit Euler and $p = 2$ for the Trapezoidal method.