III. Ordinary Differential Equations

Goals:
- solve IVPs numerically
- basic methods (explicit/implicit) and their characteristics
- stability and stiff equations

Why? E.g. time evolution of CSTR
" " " tubular reactor

MATLAB: ode45, ode23s, ...

III.1 Problem statement and examples

Def.: A scalar first order Initial Value Problem (IVP) is given by

$$\frac{dy}{dt} = \dot{y}(t) = f(t, y(t))$$

because only first derivative

scalar first order Ordinary Differential Equation (ODE)

$$y(t_0) = y_0$$

Initial Value (IV)

and an interval $I = [t_0, T]$ on which the solution $y(t)$ is seeked.

initial, final time

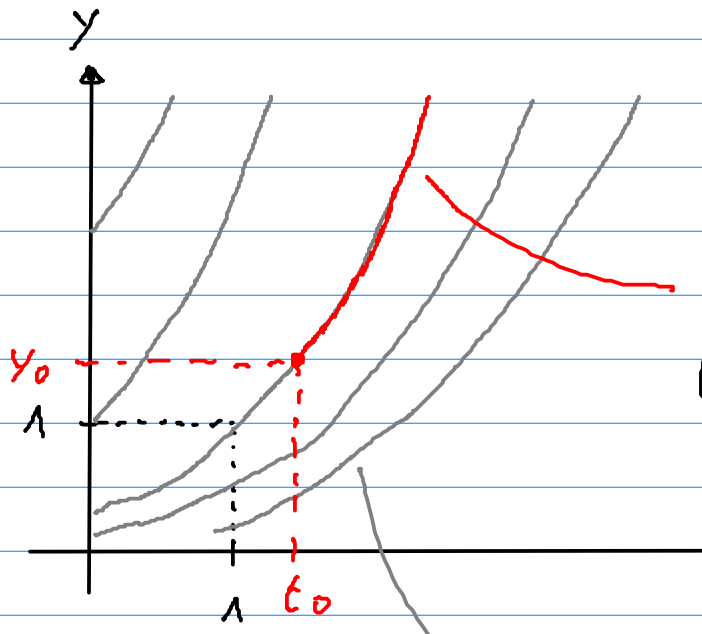
Rem.: (i) Chosen time t as the independent variable for convenience ... But it could be anything else x ...

(ii) We seek an entire function

Ex.: (1) $\left.\begin{aligned} \dot{y}(t) &= y(t) \\ y(t_0) &= y_0 \end{aligned}\right\}$ IVP

For each value of t and y the ODE prescribes the derivative $\dot{y}$ of the solution

⇝ slope fields (also direction field or vector field)



$y(t) = y_0 e^{t-t_0}$ (analytical solution to IVP)

$y(t) = C \cdot e^t$ general solution of ODE

Often not only one ODE, i.e. scalar, but a whole system

Def.: A (general) first order IVP is given by

$$\dot{\vec{y}}(t) = \vec{f}(t, \vec{y}(t)) \qquad \text{system of ODEs}$$

$$\vec{y}(t_0) = \vec{y}_0 \qquad \text{IVs}$$

and an interval $I = [t_0, T]$ on which the solution $\vec{y}(t)$ is seeked.

Ex.: (2) CSTR

(3) Linear system of ODEs

$$\dot{\vec{y}}(t) = A\,\vec{y}(t)$$

$$\vec{y}(t_0) = \vec{y}_0$$

A matrix

Analytical solution $\vec{y}(t) = e^{A(t-t_0)}\,\vec{y}_0$

$$= \left(\sum_{k=0}^{\infty} \frac{1}{k!} A^k (t-t_0)^k \right) \vec{y}_0$$

matrix exponential

What about n-th order IVPs?

Def.: A scalar n-th order IVP is given by

$$y^{(n)}(t) = f(t, y(t), \dot{y}(t), \dots, y^{(n-1)}(t))$$

(n-th derivative; scalar n-th order ODE)

$$y(t_0) = y_0$$
$$\dot{y}(t_0) = \dot{y}_0 \qquad \text{IVs}$$
$$\vdots$$
$$y^{(n-1)}(t_0) = y_0^{(n-1)}$$

and an interval $I = [t_0, T]$ on which the solution $y(t)$ is seeked.

Ex.: (4) Equations of motion (Newton's 2nd law)

$$m\ddot{x}(t) = F$$

(mass, acceleration, force)

$$x(t_0) = x_0 \quad \text{initial position}$$
$$\dot{x}(t_0) = v_0 \quad \text{initial velocity}$$

In the following we will only see numerical methods for first order IPV.

Reduction to first order system

Given an n-th order ODE

$$y^{(n)}(t) = f\left(t, y(t), \dot{y}(t), \ddot{y}(t), \dots, y^{(n-1)}(t)\right)$$

We define

$$\begin{aligned} z_0(t) &= y(t) \\ z_1(t) &= \dot{y}(t) &&= \dot{z}_0(t) \\ z_2(t) &= \ddot{y}(t) &&= \dot{z}_1(t) \\ &\vdots \\ z_{n-1}(t) &= y^{(n-1)}(t) &&= \dot{z}_{n-2}(t) \end{aligned}$$

$n-1$ first order ODEs

Now

$$\begin{aligned} z_n(t) = y^{(n)}(t) &= \dot{z}_{n-1}(t) \\ &= f\left(t, y(t), \dot{y}(t), \ddot{y}(t), \dots, y^{(n-1)}(t)\right) \\ &= f\left(t, z_0(t), z_1(t), z_2(t), \dots, z_{n-1}(t)\right) \end{aligned}$$

$\rightsquigarrow$ n first order ODEs !

By defining

$$\vec{z}(t) = \begin{pmatrix} z_0(t) \\ z_1(t) \\ \vdots \\ z_{n-2}(t) \\ z_{n-1}(t) \end{pmatrix} \quad \text{and} \quad \vec{g}(t, \vec{z}) = \begin{pmatrix} z_1(t) \\ z_2(t) \\ \vdots \\ z_{n-1}(t) \\ f(t, z_0(t), \dots, z_{n-1}(t)) \end{pmatrix}$$

we get a first order system of ODEs

$$\dot{\vec{z}}(t) = \vec{g}(t, \vec{z}(t))$$

Together with the IVs we get a first order IPV.

Ex.: (S) $\ddot{y} = \dot{y} + y^2 - e^t$, $t \in [t_0, T]$

$y(t_0) = 1$, $\dot{y}(t_0) = 0$

Reduction to first order:

$$z_0(t) = y(t)$$
$$z_1(t) = \dot{y}(t) = \dot{z}_0(t)$$
$$z_2(t) = \ddot{y}(t) = \dot{z}_1(t)$$
$$= \dot{y}(t) + y^2(t) - e^t$$
$$= z_1(t) + z_0^2(t) - e^t$$

With

$$\vec{z}(t) = \begin{pmatrix} z_0(t) \\ z_1(t) \end{pmatrix}, \quad \vec{g}(t, \vec{z}(t)) = \begin{pmatrix} z_1(t) \\ z_1(t) + z_0^2(t) - e^t \end{pmatrix}$$

and IV

$$\vec{z}(t_0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

we have a first order IVP ✓.

Solving IVP analytically is in general difficult or even impossible:

$$\frac{dy(t)}{dt} = f(t, y(t)) \qquad \Big| \int_{t_0}^{t} \cdot \, dt$$

$$\vdots$$

$$y(t) = y(t_0) + \int_{t_0}^{t} f(\tau, y(\tau)) \, d\tau$$

"solution depends on solution..."

⇝ Numerical methods

$$\approx y(t_0) + \sum_{\substack{\text{steps} \\ \text{until} \\ t}} f(\tau, y_\tau) \cdot \Delta\tau$$

approx. sol. at τ ; (time) step size

III.2 The Euler methods

Consider the scalar IVP

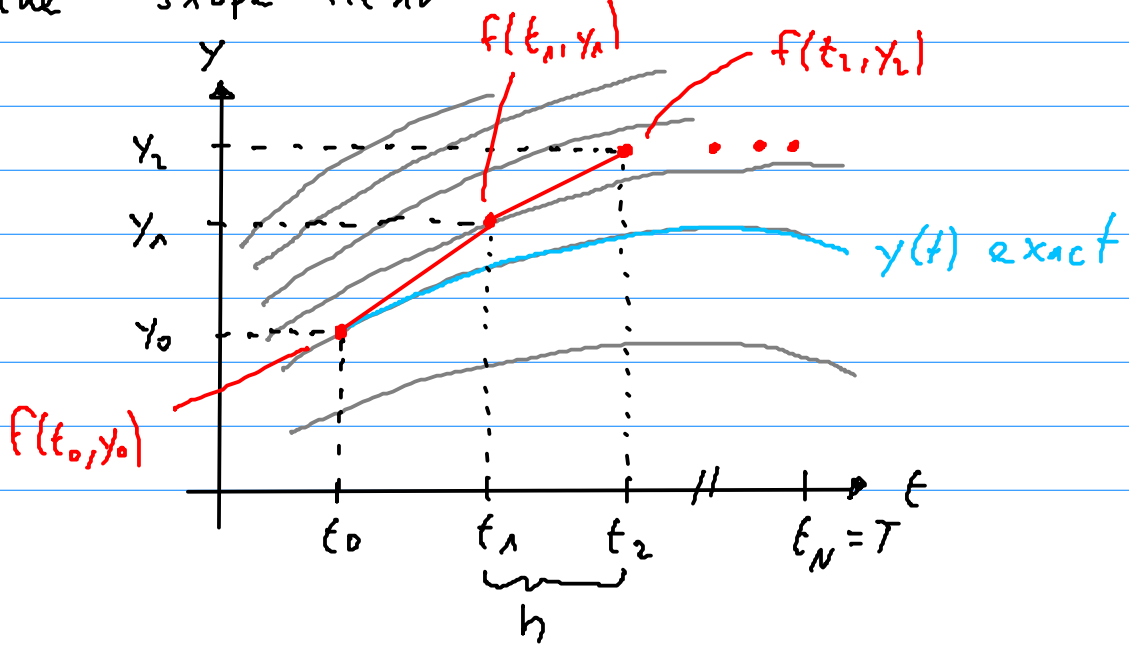
$$\dot{y}(t) = f(t, y(t))$$
$$y(t_0) = y_0$$

Goal: Find approximation of the solution $y(t)$ for $t_0 \le t \le T$.

Idea: Partition the interval $[t_0, T]$ regularly with

$$t_j = t_0 + j \cdot h \qquad \text{(step size)}$$
$$= t_0 + j \cdot \frac{T - t_0}{N} \qquad j = 0, 1, \dots, N-1$$

and approx. the solution by "following" the slope field



$$\rightsquigarrow \quad y_1 = y_0 + h \cdot f(t_0, y_0)$$

$$y_2 = y_1 + h \cdot f(t_1, y_1)$$

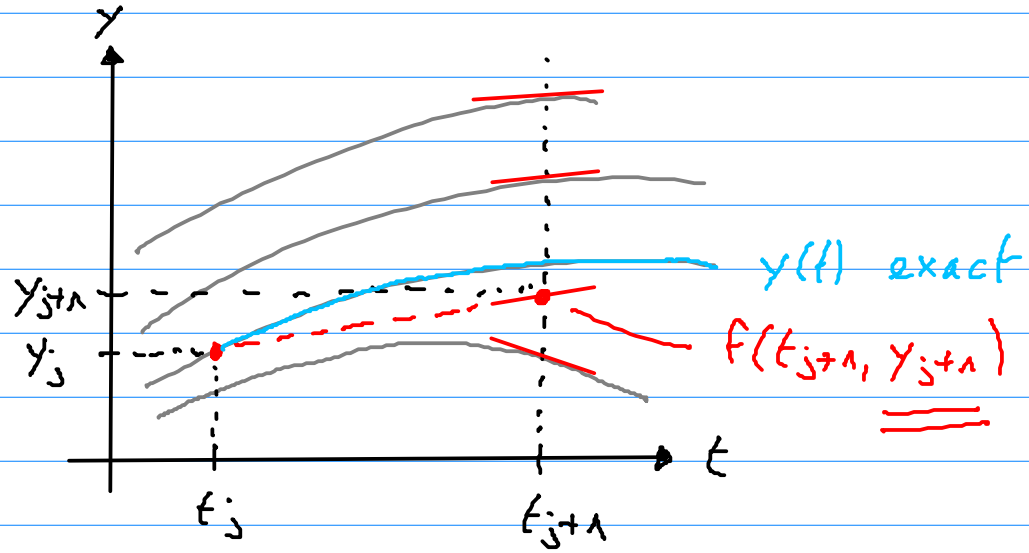
$$\vdots$$

We obtain the (explicit) Euler method (EE)

$$y_{j+1} = y_j + h \cdot f(t_j, y_j) \,, \quad j = 0, 1, \ldots, N-1$$

a.k.a. forward Euler

A different possibility



We obtain the implicit Euler method (IE)

$$y_{j+1} = y_j + h \cdot f(t_{j+1}, y_{j+1})$$

Need to solve for y_{j+1} !
⇝ implicit

03.10.24

A.k.a. backward Euler

Ex.: (6) $\dot{y}(t) = -y(t) + 2 \cdot \cos(t)$

$y(0) = 1$

⇝ Slides (Euler Methods)

Rem.: For systems of ODEs, just replace y, f by $\vec{y}, \vec{f}$

III.3 Error estimation and convergence

To analyze the accuracy of so-called one-step methods we consider the general expression

$$y_{j+1} = y_j + h \cdot \phi(t_j, y_j, y_{j+1}, h)$$

and call ϕ the increment function.

Rem.: (i) $\phi(t_j, y_j, y_{j+1}, h) = f(t_j, y_j)$ EE

(ii) $\phi(t_j, y_j, y_{j+1}, h) = f(t_{j+1}, y_{j+1})$ IE

Def.: the local truncation error (LTE) at t_j is defined by

$$e_j = y(t_j) - \left(y(t_{j-1}) + h \cdot \phi(t_{j-1}, y(t_{j-1}), y(t_j), h) \right)$$

exact sol. at t_j ; t_{j-1}

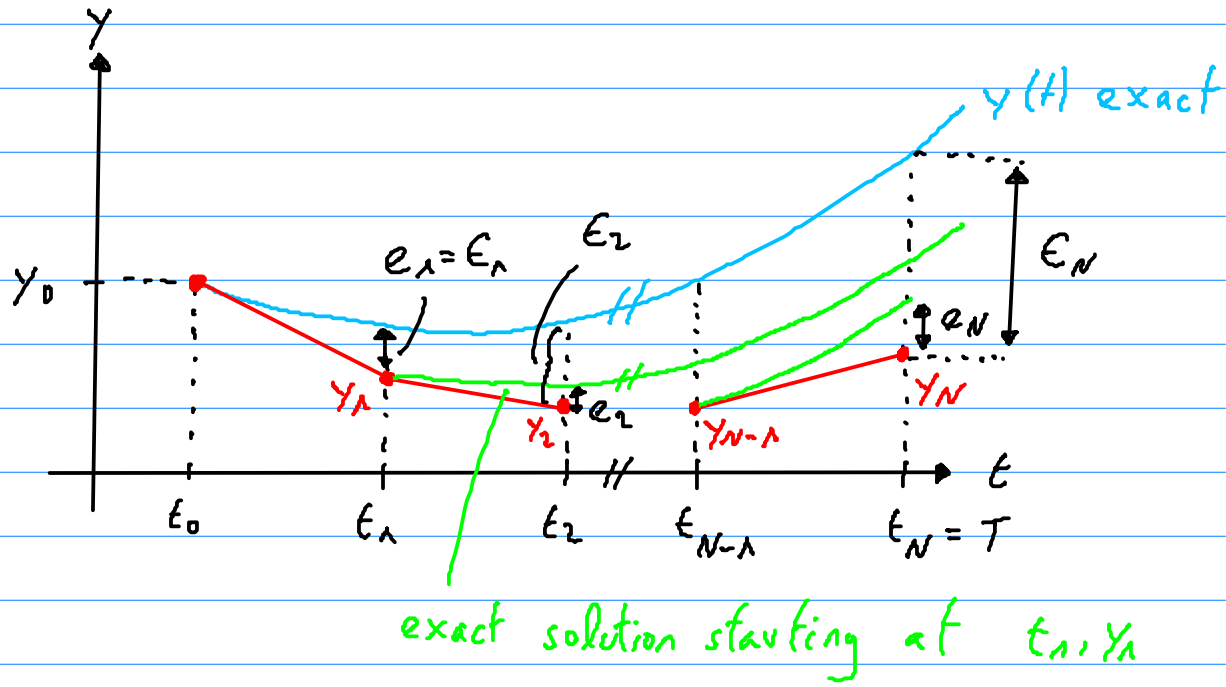
The LTE is the error after one step of a method executed with the exact solution.

Def.: the global truncation error (GTE) at t_j is defined by

$$\epsilon_j = y(t_j) - y_j$$

exact ; approx. solution at t_j

Graphically:



The LTE can "easily" be obtained by a Taylor expansion:

Ex.: (1) LTE of EE $\phi(t_j, y_j, y_{j+1}, h) = f(t_j, y_j)$

$$e_j \overset{\text{def.}}{=} y(t_j) - \left(y(t_{j-1}) + h \cdot \phi(t_{j-1}, y(t_{j-1}), y(t_j)) \right)$$

$$= y(t_j) - y(t_{j-1}) - h \cdot f(t_{j-1}, y(t_{j-1}))$$

$$= y(t_{j-1} + h) - y(t_{j-1}) - h \cdot f(t_{j-1}, y(t_{j-1}))$$

Taylor

$$= y(t_{j-1}) + h \cdot \dot{y}(t_{j-1}) + \frac{h^2}{2} \ddot{y}(t_{j-1}) + \frac{h^3}{6} \dddot{y}(t_{j-1}) + \dots$$

ODE $f(t_{j-1}, y(t_{j-1}))$

$$- y(t_{j-1}) - h \cdot f(t_{j-1}, y(t_{j-1}))$$

$$= \frac{h^2}{2} \ddot{y}(t_{j-1}) + h^3 \cdot (\dots) + h^4 \cdot (\dots) + \dots = \mathcal{O}(h^2)$$

Ex.: (8) LTE of IE $\phi(t_j, y_j, y_{j+1}, h) = f(t_{j+1}, y_{j+1})$

$$e_j \overset{\text{Def.}}{=} y(t_j) - y(t_{j-1}) - h \cdot f(t_j, y(t_j))$$

$$= \dots$$

$y(t_j - h) = \dots$ Taylor...

$$= \mathcal{O}(h^2)$$

Rem.: The LTE depends on the smoothness of the solution (e.g. $\ddot{y}$ above!) and therefore on the smoothness of the right-hand side function $f(t, y(t))$:

$$\ddot{y}(t) = \frac{d}{dt}\left(\dot{y}(t)\right)$$

$$\overset{\text{ODE}}{=} \frac{d}{dt}\left(f(t, y(t))\right)$$

$$= \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{dy}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} f$$

ODE $\dot{y} = f(t, y)$

So we have seen that the LTE error is easy to compute with Taylor series...

How is the LTE related to the practically more meaningfull GTE?

Ex.: (9) EE & IE GTE

⇝ Slides (Euler Methods)

We observe that although the LTE behaves as $O(h^2)$, the GTE is $O(h)$

Actually, one can show (under requirements that are anyway necessary for the existence and uniqueness of solutions to an IVP) that the LTEs accumulate in each step:

$$|\epsilon_N| \leq N \cdot \max_{1 \leq j \leq N} |e_j| = O(h)$$

$N \sim \frac{1}{h}$, $\max |e_j| = O(h^2)$

And we have convergence: $y_N \xrightarrow{h \to 0} y(T)$

This motivates the following definition

Def.: We say that a method has order of accuracy p if the LTE is

$$|e_j| = O(h^{p+1})$$

$p+1$!

or equivalently

$$|E_j| = O(h^p)$$

Rem.: EE and IE have order of accuracy $p=1$.

III.4 Runge-Kutta methods

So far: Euler methods with error $\mathcal{O}(h)$

$\rightsquigarrow$ want 100 x smaller error

↳ ? (100) x smaller step size h

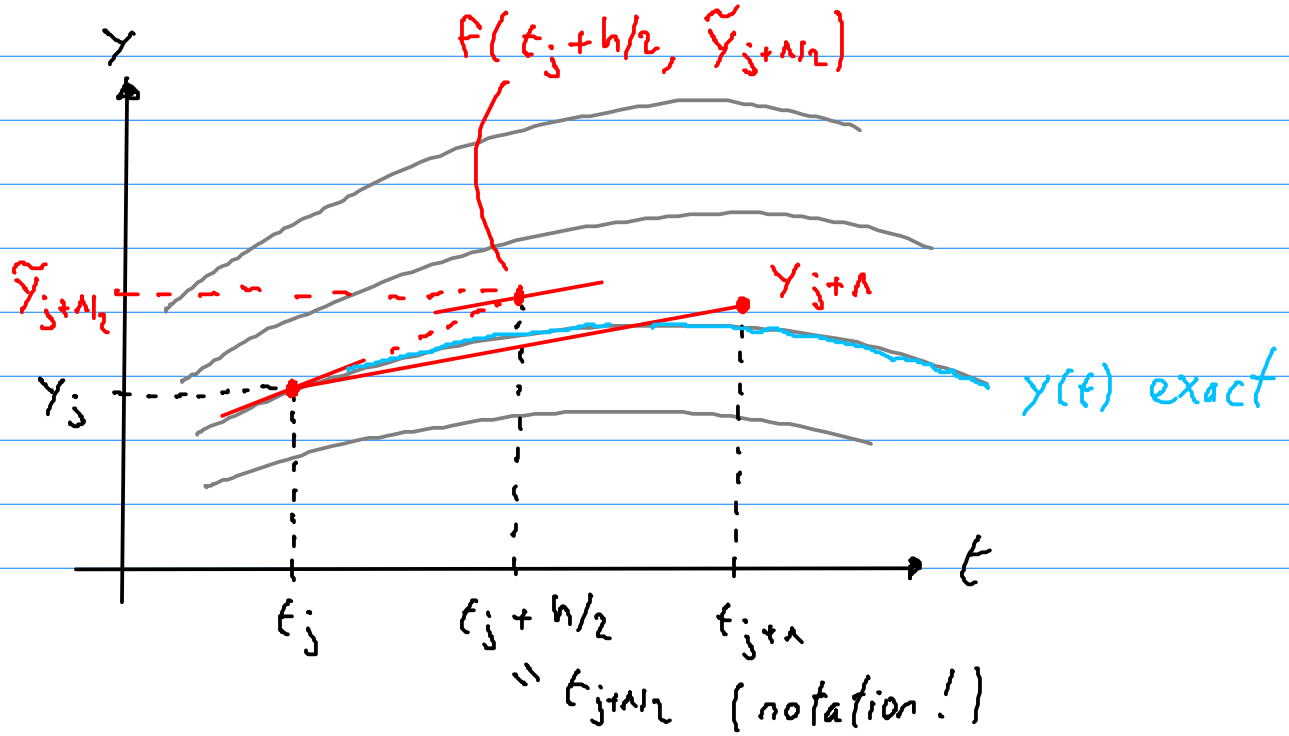
? (100) x more (computational) work

Can we do better? YES!

Try: $y_{j+1} = y_j + h \cdot f\left(t_j + \frac{h}{2}, \tilde{y}_{j+1/2}\right)$

$\tilde{y}_{j+1/2}$: Approx. of sol. at $t_j + h/2$?

Let's go back to the slope field:



Idea: approx. $\tilde{y}_{j+1/2}$ with half EE step

Is this better?

Ex.: (10) Approx. IVP of Ex. (6) with above scheme

⟶ Slides (Runge's Method)

We observe in the above numerical experiment that the GTE of this scheme is $O(h^2)$. Hence it's order of accuracy is $p=2$ (and it is indeed more accurate than the Euler methods).

We could compute the LTE.

$$e_j = y(t_j) - \Big[y(t_{j-1}) + h \cdot f\Big(t_{j-1} + \frac{h}{2}, y(t_{j-1}) + \frac{h}{2} f(t_{j-1}, y(t_{j-1}))\Big)\Big]$$

$$= \dots \text{ easy, but tedious } \dots$$

$$= O(h^{3}) \quad (2+1)$$

The scheme we just derived is known as Runge's (~1895) or modified Euler's or explicit midpoint method (EM).

We write it as

$$k_1 = f(t_j, y_j)$$

← slope approx.!

$$k_2 = f\left(t_j + h/2,\ \underbrace{y_j + h/2 \cdot k_1}_{\tilde{y}_{j+1/2}}\right)$$

$$y_{j+1} = y_j + h \cdot k_2$$

16.10.24

Another possibility is Heun's method

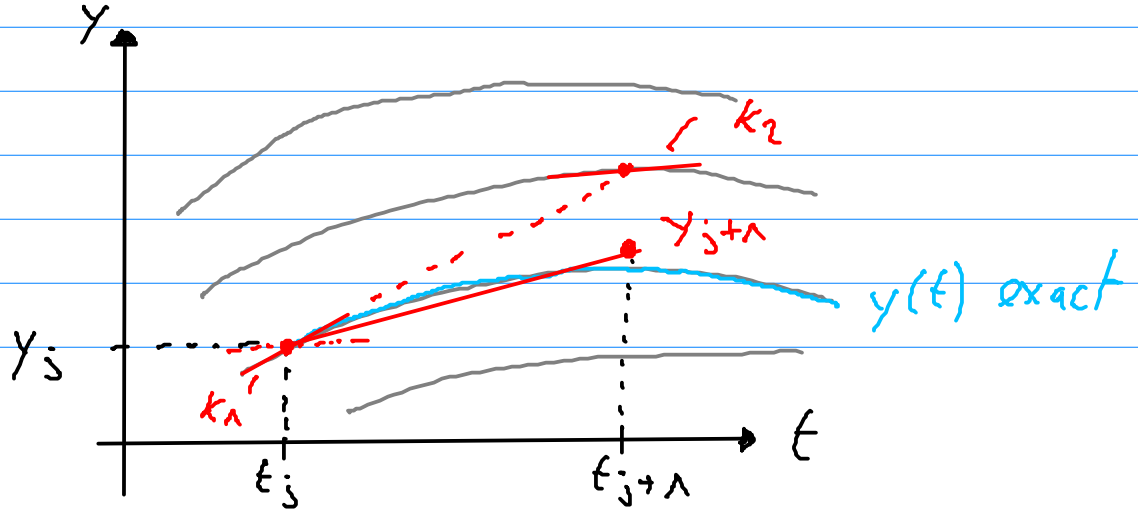
$$k_1 = f(t_j, y_j)$$

$$k_2 = f(t_j + h,\ y_j + h \cdot k_1)$$

$$y_{j+1} = y_j + \frac{h}{2}(k_1 + k_2)$$

or the explicit trapezoidal method (ET)

Graph.:



One can (easily) show, that Heun's method has order of accuracy $p=2$.

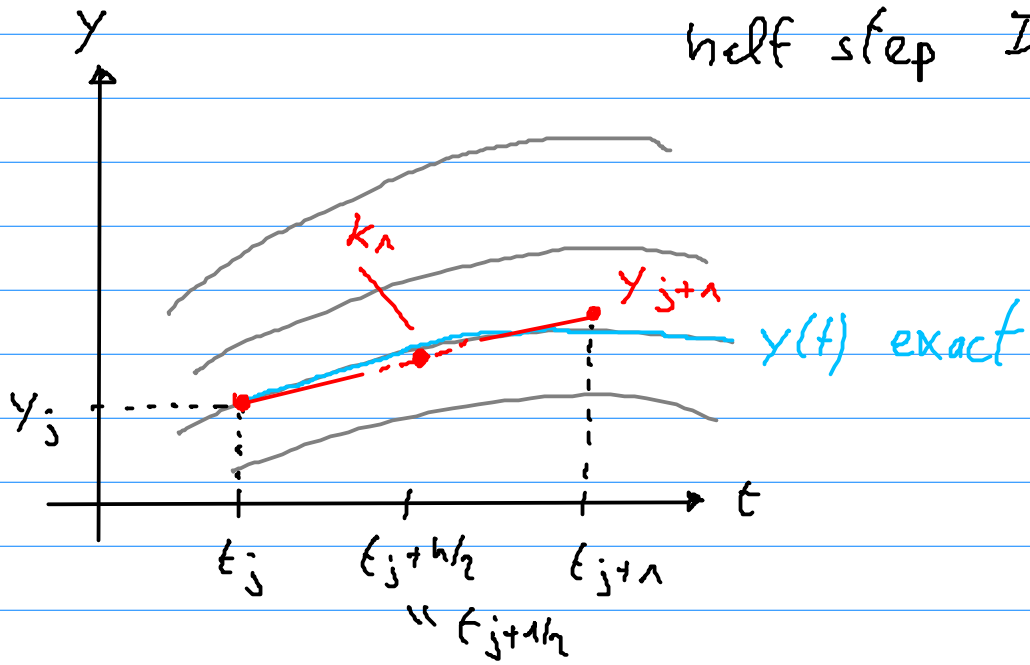
So far we have used the EE to estimate the slopes...

Let's try with IE:

$$y_{j+1} = y_j + h \cdot f\left(t_j + h/2,\ \tilde{y}_{j+1/2}\right)$$

with $\tilde{y}_{j+1/2} = y_j + \frac{h}{2} \cdot f\left(t_j + h/2,\ \tilde{y}_{j+1/2}\right)$

half step IE!



This method is known as the implicit midpoint method (IM):

$$k_1 = f\left(t_j + h/2,\ y_j + \frac{h}{2} k_1\right)$$

$$y_{j+1} = y_j + h \cdot k_1$$

Another popular scheme is the implicit trapezoidal method (IT):

$$k_1 = f(t_j, y_j)$$

$$k_2 = f\left(t_j + h,\ y_j + \frac{h}{2}(k_1 + k_2)\right)$$

$$y_{j+1} = y_j + \frac{h}{2}(k_1 + k_2)$$

Sometimes written as

$$y_{j+1} = y_j + \frac{h}{2}\left(f(t_j, y_j) + f(t_{j+1}, y_{j+1})\right)$$

Clearly looks like the trapezoidal rule and hence the name!

Both IM and IT have order of accuracy $p = 2$.

The methods we have seen so far are all representatives of a large class of one-step methods known as Runge-Kutta (RK) methods:

$$k_i = f\left(t_j + c_i \cdot h,\ y_j + h \cdot \sum_{l=1}^{s} a_{il} \cdot k_l\right)$$

$$y_{j+1} = y_j + h \cdot \sum_{i=1}^{s} b_i \cdot k_i$$

where:

- s ... number of stages
- c_i ... nodes
- b_i ... weights
- a_{il} ... RK matrix
- k_i ... slopes

It's convenient to write RK methods with a so-called Butcher tableau (BT):

$$\begin{array}{c|ccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} = \begin{array}{c|c} \vec{c} & A \\ \hline & \vec{b}^T \end{array}$$

Ex.: (11) BTs for methods seen so far
⇝ Slides (RK Methods)

(12) Comparison of RK methods on IVP of Ex. (6) ⇝ Slides (RK Methods)

Rem.: (i) For explicit methods the RK matrix A is lower triangular

$$A = \begin{pmatrix} 0 & & \\ a_{21} & & 0 \\ \vdots & \ddots & \\ a_{s1} & \cdots & 0 \end{pmatrix}$$

(ii) For system: replace y, f by $\vec{y}, \vec{f}$

There is indeed a tight connection between RK methods and quadrature. To see this, let's have a look at the LTE:

$$\begin{aligned} e_{j+1} &\overset{\text{Def.}}{=} y(t_{j+1}) - \left(y(t_j) + h \cdot \sum_{i=1}^{s} b_i \cdot k_i \right) \\ &= y(t_{j+1}) - y(t_j) - h \cdot \sum_{i=1}^{s} b_i \cdot k_i \\ &= \int_{t_j}^{t_{j+1}} \dot{y}(t)\, dt - h \cdot \sum_{i=1}^{s} b_i \cdot k_i \qquad \text{(ODE)} \\ &= \int_{t_j}^{t_{j+1}} f(t, y(t))\, dt - h \cdot \sum_{i=1}^{s} b_i \cdot k_i \end{aligned}$$

(b_i: QW; k_i: $f(\dots)$)

III.5 Absolute stability

Let's consider the very simple IVP

$$\dot{y}(t) = \lambda\, y(t) \qquad (\lambda \in \mathbb{R} \text{ or } \mathbb{C})$$
$$y(0) = y_0$$

which is known as the Dahlquist test equation (DTE) (in the present context).

The analytical solution is simply

$$y(t) = y_0\, e^{\lambda t}$$

Let's apply EE to this IVP

$$\begin{aligned} y_{j+1} &= y_j + h \cdot f(t_j, y_j) \\ &= y_j + h\lambda\, y_j \\ &= (1 + h\lambda)\, y_j \\ &= (1 + h\lambda)^2\, y_{j-1} \\ &\vdots \\ &= (1 + h\lambda)^{j+1}\, y_0 \end{aligned}$$

$y_j = (1+h\lambda)\, y_{j-1}$

$y_{j-1} = (1+h\lambda)\, y_{j-2}$

Now let's qualitatively compare the exact to the approx. solution:

(i) $\lambda > 0$: $y(t)$ increases

y_j ? increases ✓.

(ii) $\lambda < 0$: $y(t)$ decreases

y_j ? depends on the step size h: $0 < h < \frac{2}{|\lambda|}$

Let's apply IE to the same IVP

$$y_{j+1} = y_j + h \cdot f(t_{j+1}, y_{j+1})$$

$$= y_j + h\lambda \cdot y_{j+1}$$

(solve for y_{j+1} IMPLICIT!)

$$y_{j+1} = \frac{1}{1 - h\lambda} y_j$$

$y_j = \frac{1}{1-h\lambda} y_{j-1}$

$$= \left(\frac{1}{1-h\lambda}\right)^2 y_{j-1}$$

$$\vdots$$

$$= \left(\frac{1}{1-h\lambda}\right)^{j+1} y_0$$

How does it look here for $\lambda < 0$?

$y(t)$ decreases

y_j ? decreases (does not depend on the step size h!)

30.10.24

Def.: A one-step method applied to the DTE can be written as

$$y_{j+1} = g(z)\, y_j$$

where $z = h\lambda$ and $g(z)$ is called the stability function (SF) of the method

Ex.: (13) EE: $g(z) = 1 + z$

IE: $g(z) = 1/(1 - z)$

(14) EM: $g(z) = 1 + z + \frac{z^2}{2}$

We want at least that the approx. solution follows the trend of the exact one!

decay!

For the practically meaningfull case $\lambda < 0$ we demand the approx. solution decreases in magnitude:

$$|y_{j+1}| < |y_j| \quad (\underline{\text{Absolute stability}})$$

or simply

$\underline{A\text{-stability}}$ (AS)

So with y_{j+1} (SF)

$$|y_{j+1}| = |g(z)\, y_j| = |g(z)|\,|y_j| < |y_j|$$

we get a requirement on the SF

$$|g(z)| < 1$$

This motivates the following definition

Def.: Given a one-step method with SF $g(z)$

For $\lambda \in \mathbb{R}$ we call

$$SI = \{ x = h\lambda \in \mathbb{R} \mid |g(x)| < 1 \}$$

the method's **stability interval** (SI)

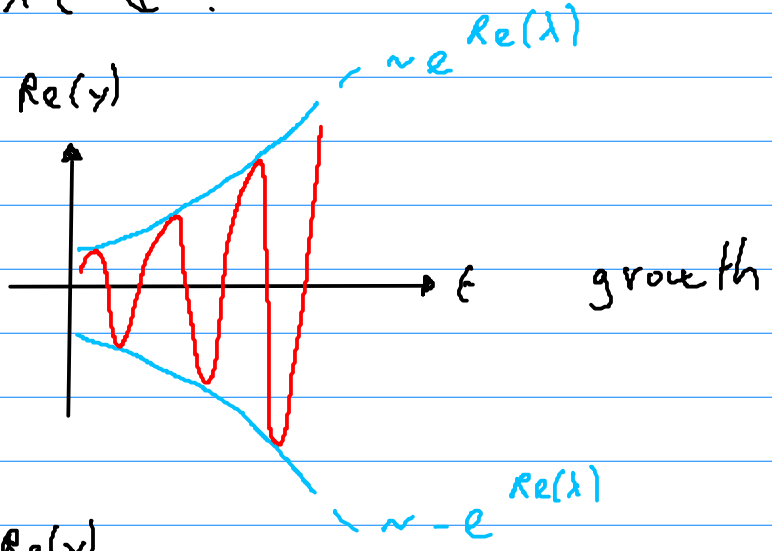
For $\lambda \in \mathbb{C}$ we call

$$SR = \{ z = h\lambda \in \mathbb{C} \mid |g(z)| < 1 \}$$

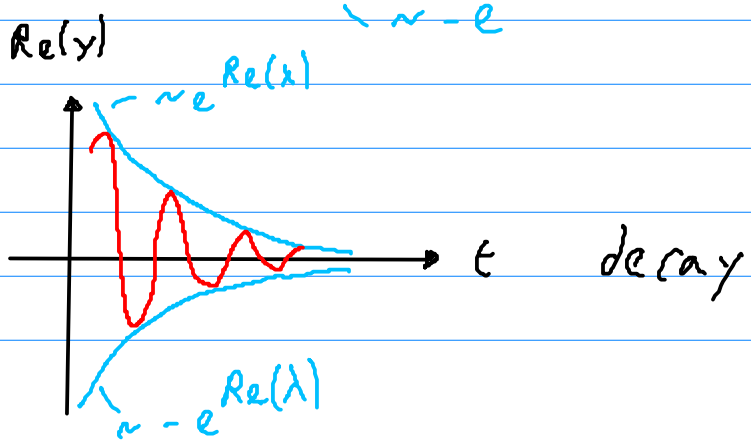
the methods **stability region** (SR)
(or ~ domain)

In general $\lambda \in \mathbb{C}$:

(i) $Re(\lambda) > 0$



(ii) $Re < 0$



Def.: A method is called **A-stable** if the whole left complex half plane is contained in the SR:

$$\{ z \in \mathbb{C} \mid Re(z) < 0 \} \subset SR$$

Hence, for A-stable methods there is no time step restriction for $\lambda < 0$!

Ex.: (AS) A-stability of EE, IE, EM, ET, IM and RK4 methods

$\rightsquigarrow$ slides (Stability regions)

Rem.: In general, explicit RK methods are not A-stable

III.6 Stiff equations

Basically: stiff problems are problems for which "explicit methods don't work"

The step size has to be chosen much smaller for stability rather than from accuracy considerations

Ex.: (16) Stiff linear system

⇝ Slides (Stiff linear IVP)

A linear inhomogeneous system of ODEs

$$\dot{\vec{y}}(t) = A\vec{y}(t) + \vec{b}(t)$$

(A: $n \times n$ Matrix)

is called stiff if the eigenvalues of A, λ_i $(i = 1, 2, \dots, n)$, have very different negative real parts:

$$S = \frac{\max_j |Re(\lambda_j)|}{\min_j |Re(\lambda_j)|}, \quad Re(\lambda_j) < 0$$

That is: the stiffness ratio S is "large".

In ex. (16) $S = \frac{1000}{1/2} = 2000$

Stiffness is also very common in nonlinear systems of ODEs

$$\dot{\vec{y}}(t) = \vec{f}(t, \vec{y}(t))$$

A <u>local</u> measure of stiffness is obtained by linearizing the system of ODEs at some point (of interest!) $t_1, \vec{y}_1$:

$$\vec{f}(t, \vec{y}) \approx \vec{f}(t_1, \vec{y}_1) + \frac{\partial \vec{f}}{\partial t}(t_1, \vec{y}_1) \cdot (t - t_1) + \underbrace{\frac{\partial \vec{f}}{\partial \vec{y}}}_{\text{Jacobian matrix } J = \frac{\partial \vec{f}}{\partial \vec{y}}} (t_1, \vec{y}_1) \cdot (\vec{y} - \vec{y}_1)$$

So one obtains the following inhomogeneous linear system of ODEs

$$\dot{\vec{y}}(t) = \underbrace{J(t_1, \vec{y}_1)}_{A} \vec{y}(t) + \underbrace{\left(\vec{f}(t_1, \vec{y}_1) + \frac{\partial \vec{f}}{\partial t}(t_1, \vec{y}_1) \cdot (t - t_1) - J(t_1, \vec{y}_1) \vec{y}_1 \right)}_{\vec{b}(t)}$$

If this is stiff, then one says that the nonlinear system of ODEs is locally stiff around $(t_n, \vec{y}_n)$.

Ex.: (17) Stiff nonlinear system

$\rightsquigarrow$ slides (Stiff nonlinear IVP)

We conclude from Ex. (16) and (17) that explicit methods are inefficient for the numerical treatment of stiff problems:

because the step size is dictated by stability and NOT accuracy considerations

explicit	implicit
cheap per step	expensive per step
step size limited by fastest decaying component	step size only limited by desired accuracy

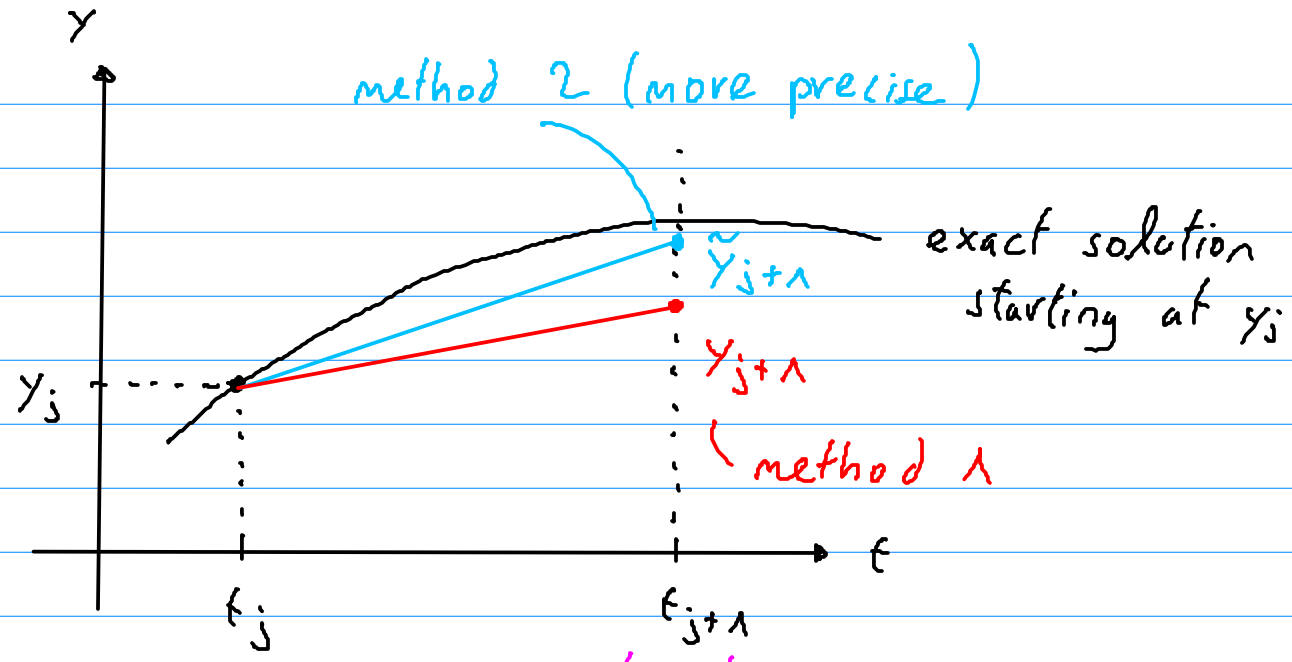
III.7 Error control and adaptive step size

In practice, one wants to choose the step size (the h!) sufficiently small to ensure some required precision and sufficiently large to avoid unnecessary computational work.

To achieve this, we need some local error estimate to indicate if the step size should be "small" or "large".

Idea: perform a step with two methods, where a first method is compared to the result of a second, more precise, method

↳ the digits that match are assumed to be correct !



error estimate

Now: If: $| y_{j+1} - \tilde{y}_{j+1} | \leq$ tolerance

AbsTol or RelTol in MATLAB

accept step

Else:

repeat step with smaller step size and check again

Rem.: (i) the more precise method (method 2) could be of higher order then method 1

or method 2 could be the same as method 1 but with a smaller step size h (substeps!)

(ii) the above pseudo-code is far from complete as one also wishes to increase the step size if possible

(iii) no guarantee that the error estimate reflects the true error.
however, often works well in practice.

Ex.: (18) Adaptive step size method (MATLAB ode45) for the van der Pol ODE:

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = \mu\left(1 - y_1^2\right) y_2 - y_1$$

with $\mu = 1$ and $\vec{y} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$, $t \in [0, 20]$

$\rightsquigarrow$ 5 slides

(19) " with $\mu = 1000$ and $t \in [0, 3000]$ (locally) makes the problem very stiff!

$\rightsquigarrow$ 5 slides & exercises

06.11.24