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

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

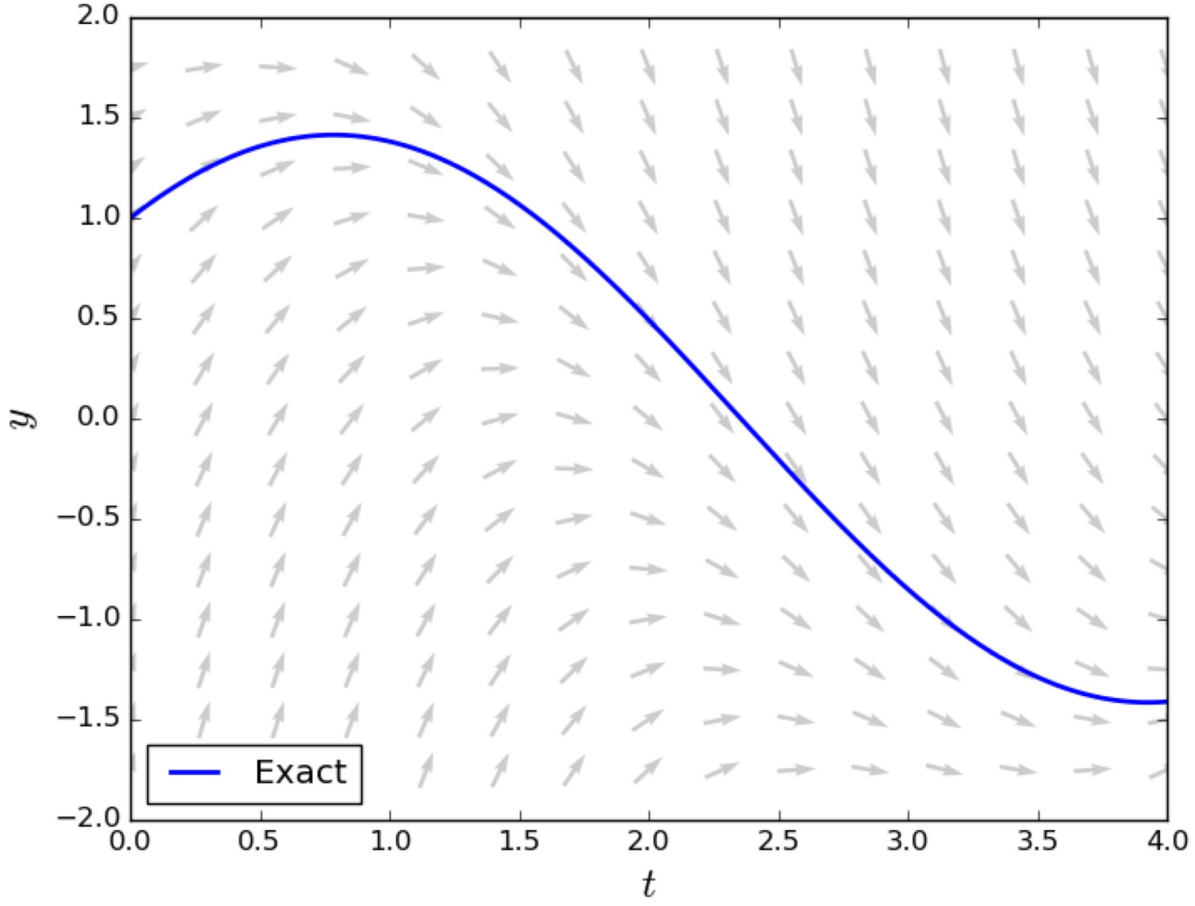
Together with the IVs we get a first order IPV.

Ex. (6)

The Euler Methods

IVP $\begin{cases} \dot{y}(t) = -y(t) + 2\cos(t) \\ y(0) = 1 \end{cases}$

$y(t) = \sin(t) + \cos(t)$

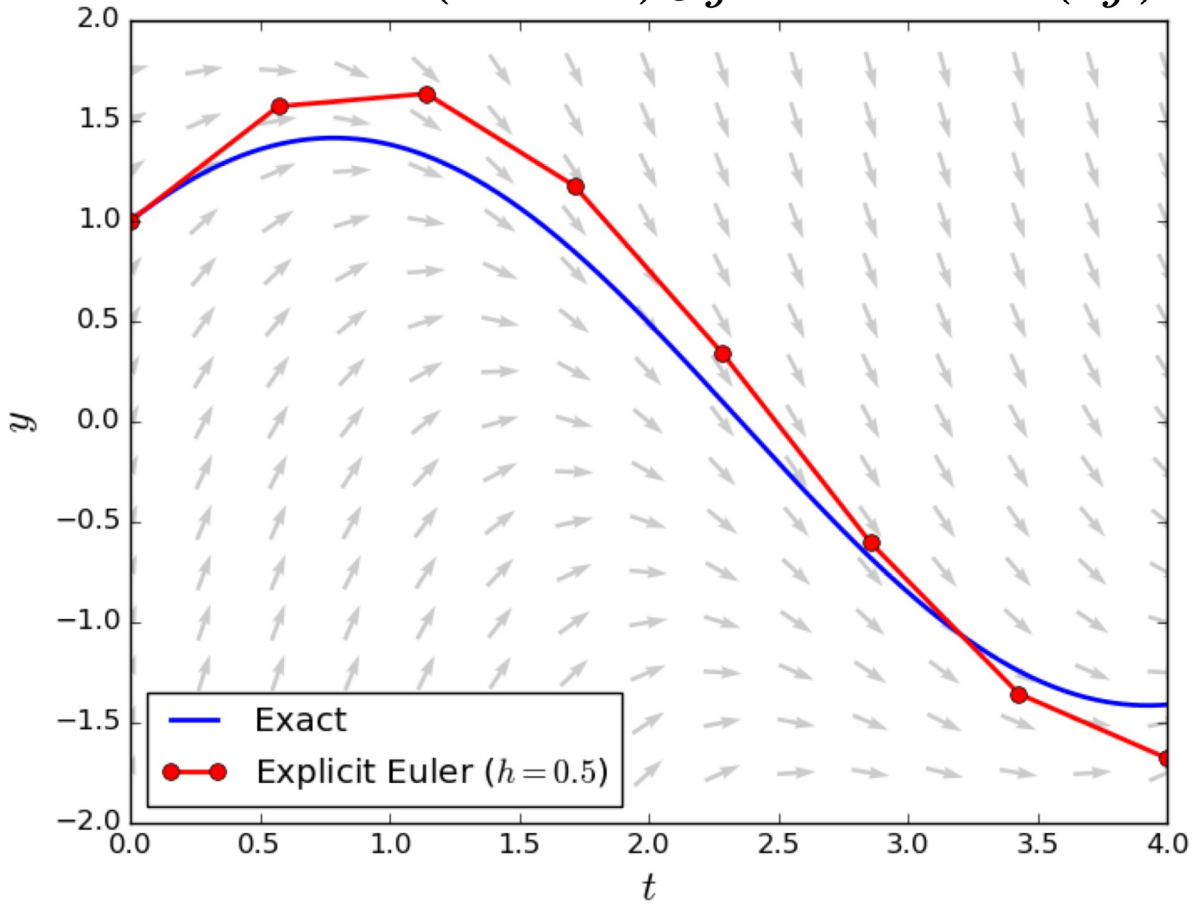


Ex. (6)

The Euler Methods

Explicit

$$y_{j+1} = y_j + hf(t_j, y_j)$$
$$= y_j + h(-y_j + 2\cos(t_j))$$
$$= (1-h)y_j + 2h\cos(t_j)$$

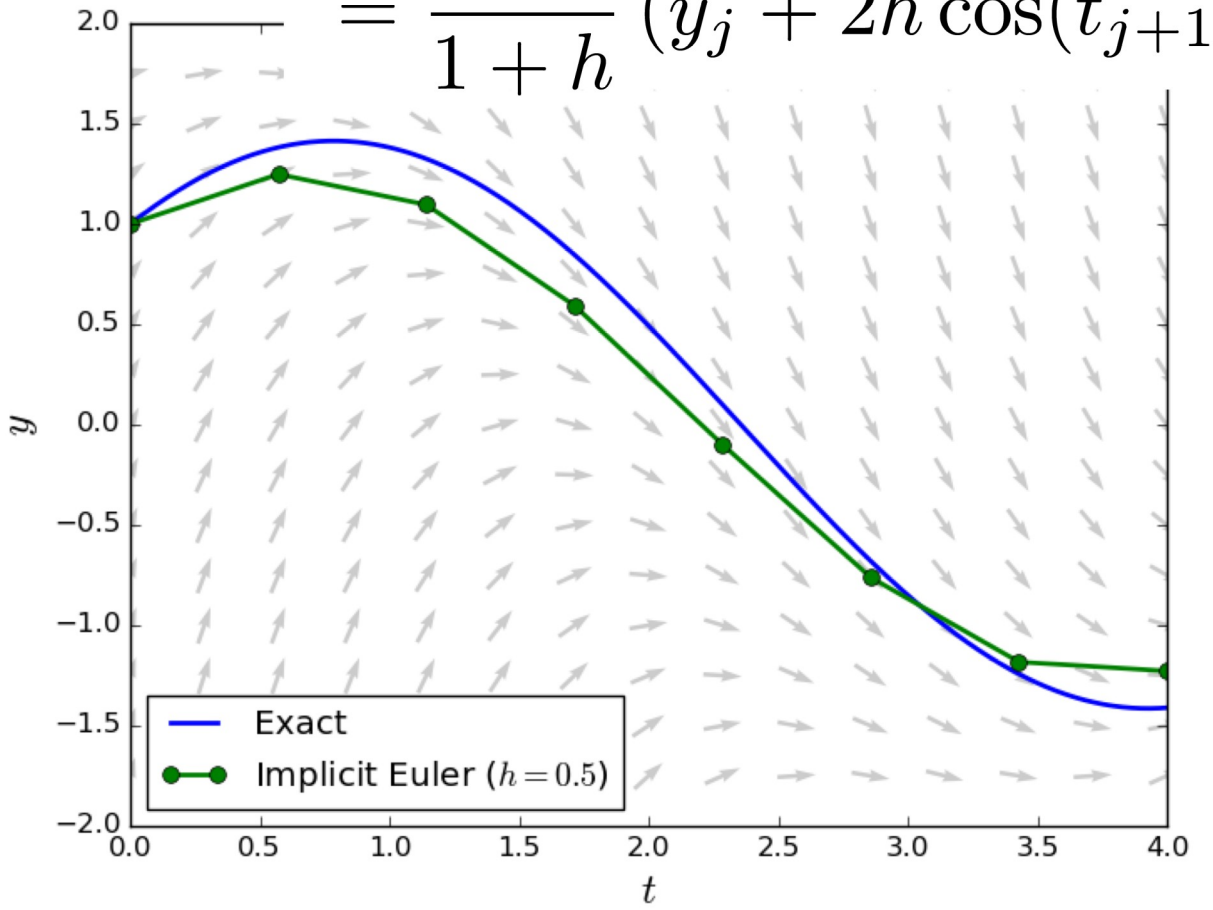


Ex. (6)

The Euler Methods

Implicit

$$y_{j+1} = y_j + hf(t_{j+1}, y_{j+1})$$
$$= y_j + h(-y_{j+1} + 2\cos(t_{j+1}))$$
$$= \frac{1}{1+h}\left(y_j + 2h\cos(t_{j+1})\right)$$

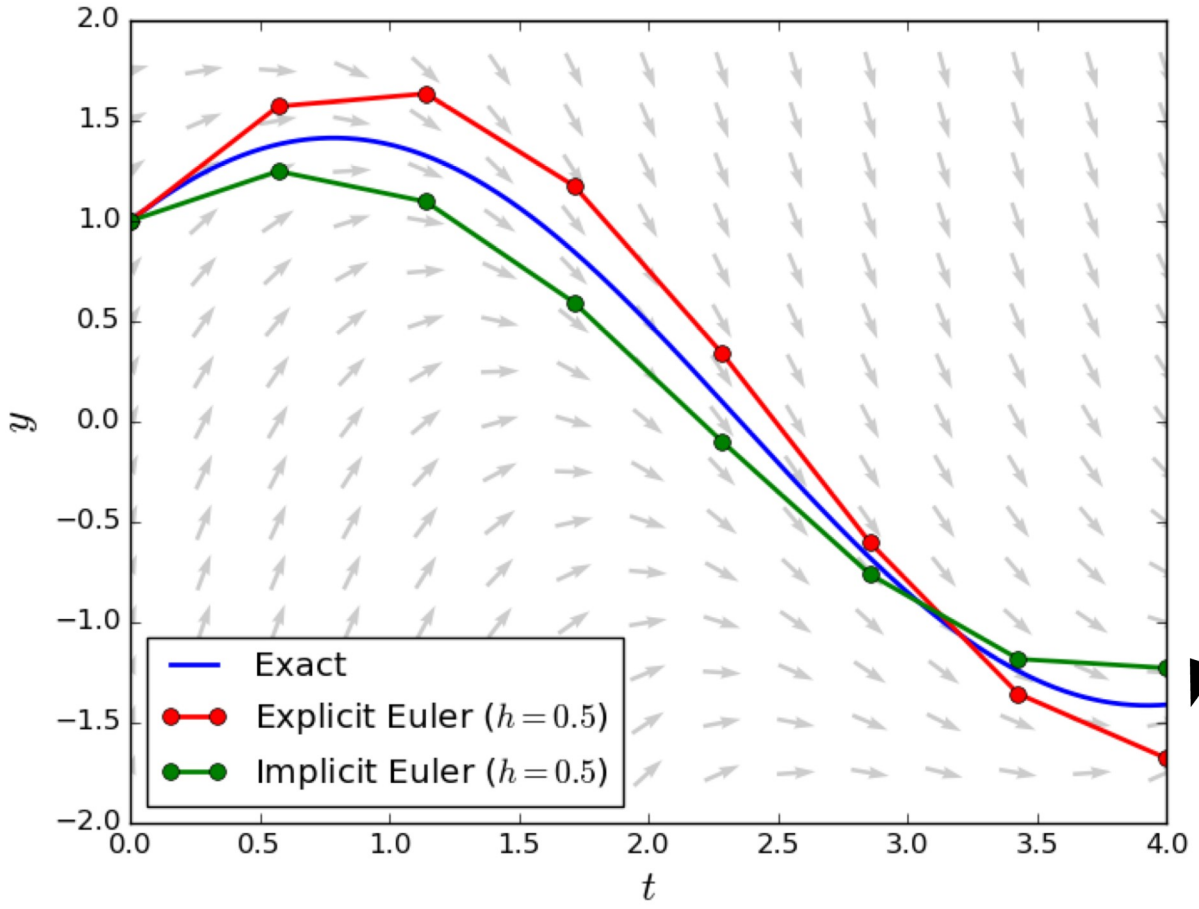


Ex. (6)

The Euler Methods

$$E_N = |y(\bar{T}) - y_N|$$

Error



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

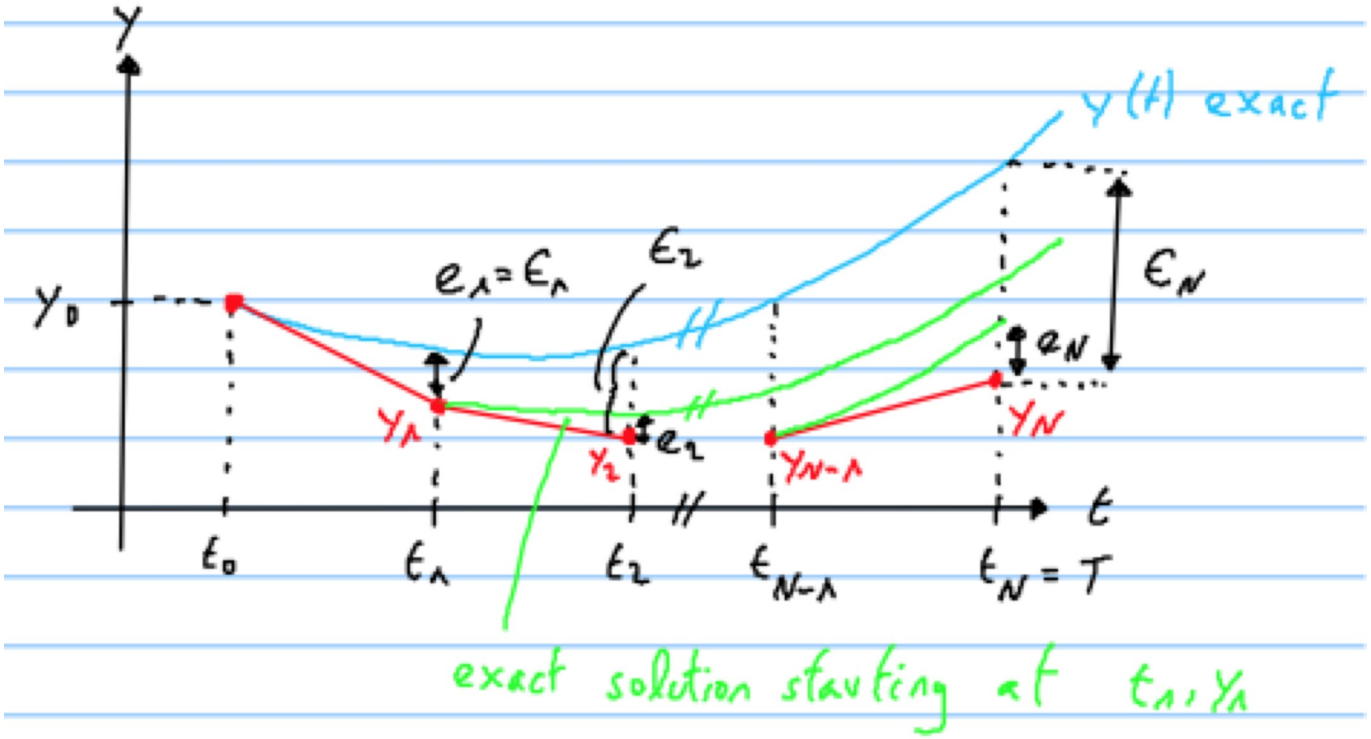
$$e_j = y(t_j) - \left(y(t_{j-1}) + h \cdot \phi(t_{j-1}, y(t_{j-1}), y(t_j), h) \right)$$

$y(t_j)$: exact sol. at t_j; $y(t_{j-1})$: t_{j-1}

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

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

$y(t_j)$: exact; y_j: approx. solution at t_j



Ex. (9)

The Euler Methods

N	h	EE $\lvert y_N - y(\bar{T})\rvert$	IE $\lvert y_N - y(\bar{T})\rvert$
8	2^{-1}	2.666E-01	1.830E-01
16	2^{-2}	1.105E-01	9.295E-02
32	2^{-3}	5.097E-02	4.688E-02
64	2^{-4}	2.453E-02	2.354E-02
128	2^{-5}	1.204E-02	1.180E-02

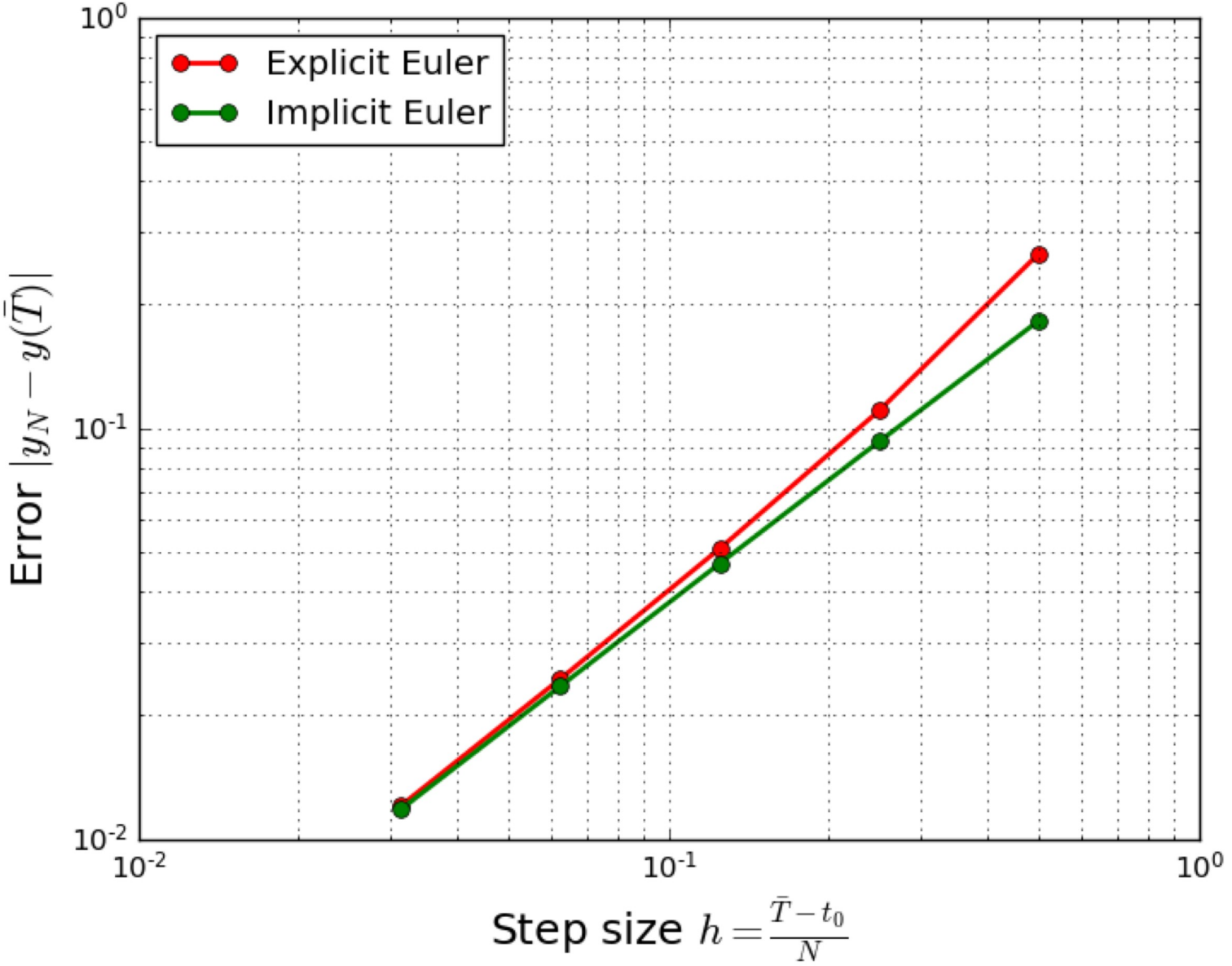
$O(h)$

We observe that the GTE is directly proportional to the stepsize h, although the LTE is proportional to its square....

$O(h^2)$

Ex. (9)

The Euler Methods



LTE, GTE and order of accuracy

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:

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

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

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

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

$p+1$!

or equivalently

$$|E_j| = \mathcal{O}(h^p)$$

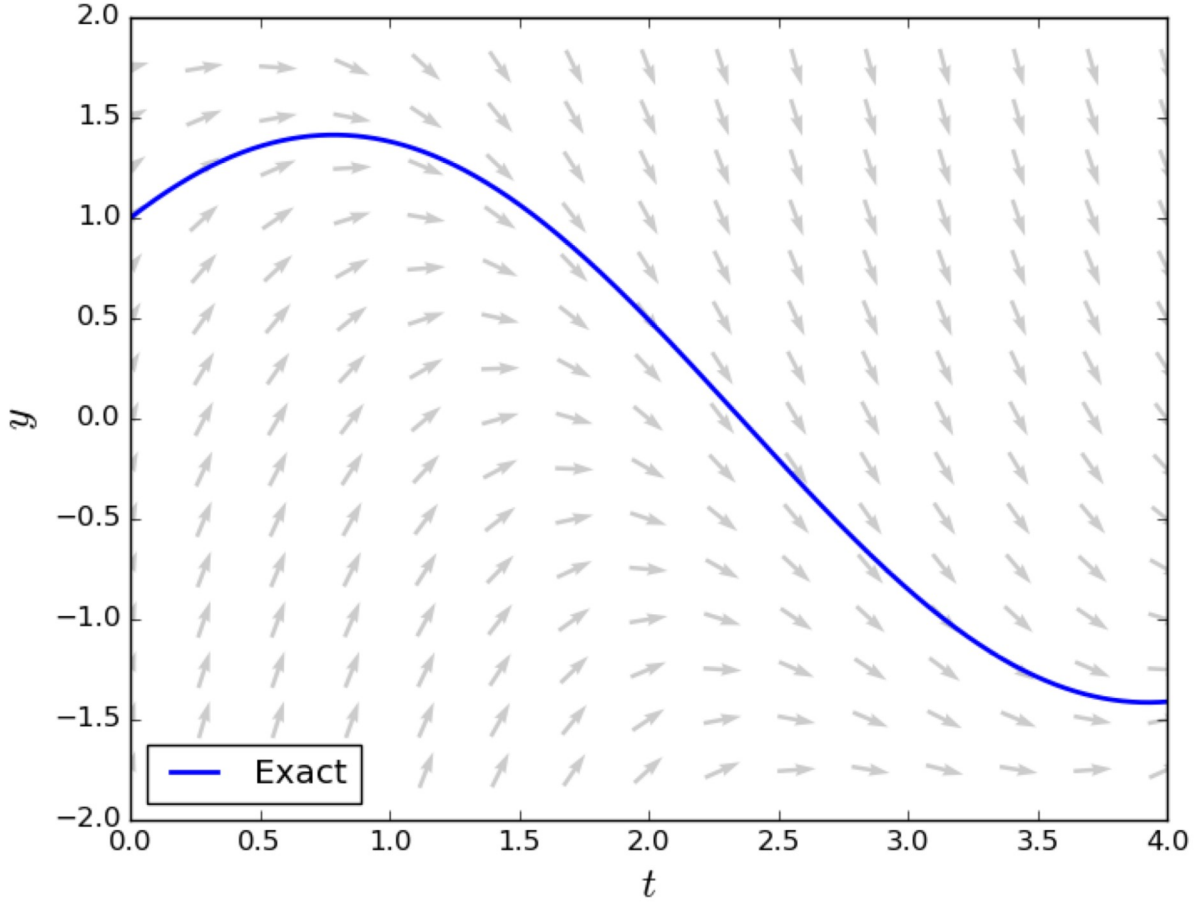
Rem.: EE and IE have order of accuracy $p=1$.

Ex. (10)

Runge‘s Method

IVP $\begin{cases} \dot{y}(t) = -y(t) + 2\cos(t) \\ y(0) = 1 \end{cases}$

$$y(t) = \sin(t) + \cos(t)$$

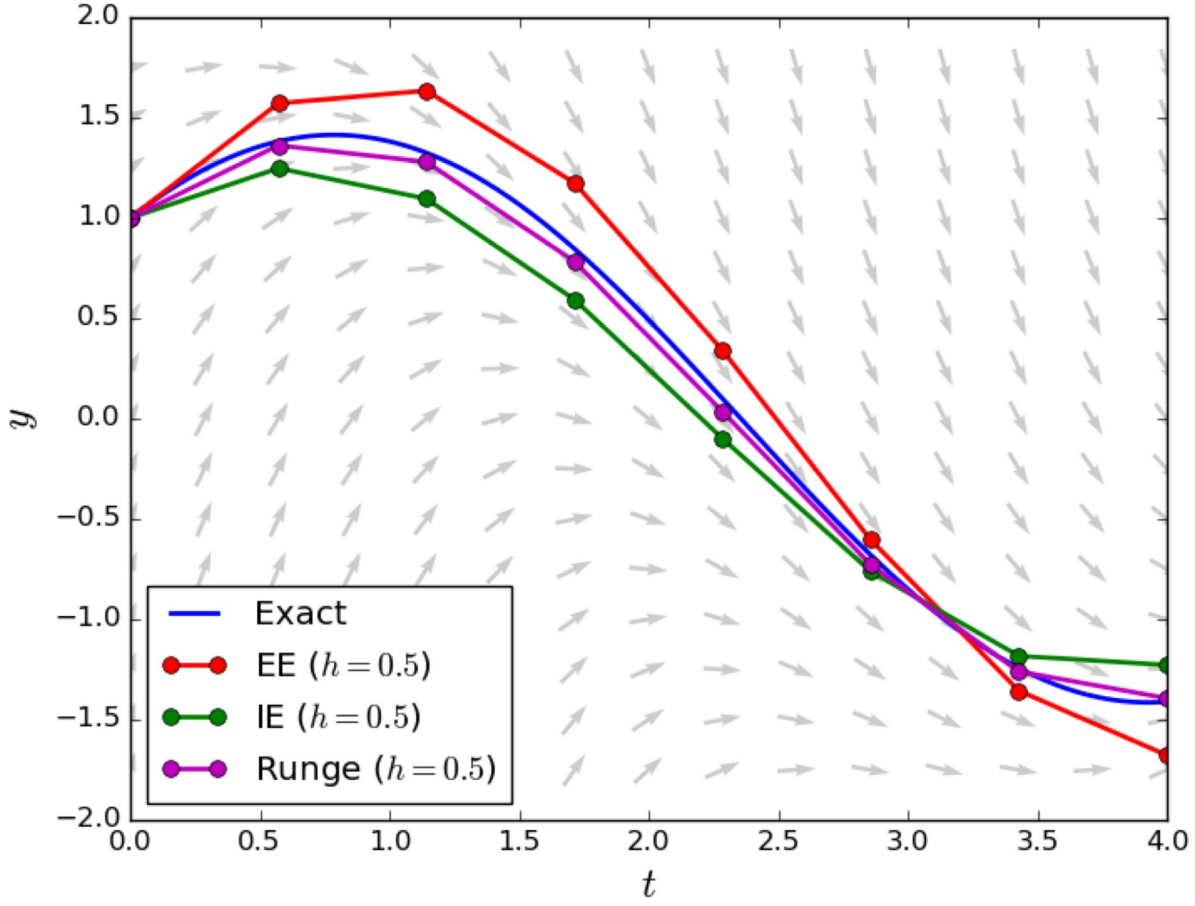


Ex. (10)

Runge‘s Method

IVP $\begin{cases} \dot{y}(t) = -y(t) + 2\cos(t) \\ y(0) = 1 \end{cases}$

$$y(t) = \sin(t) + \cos(t)$$



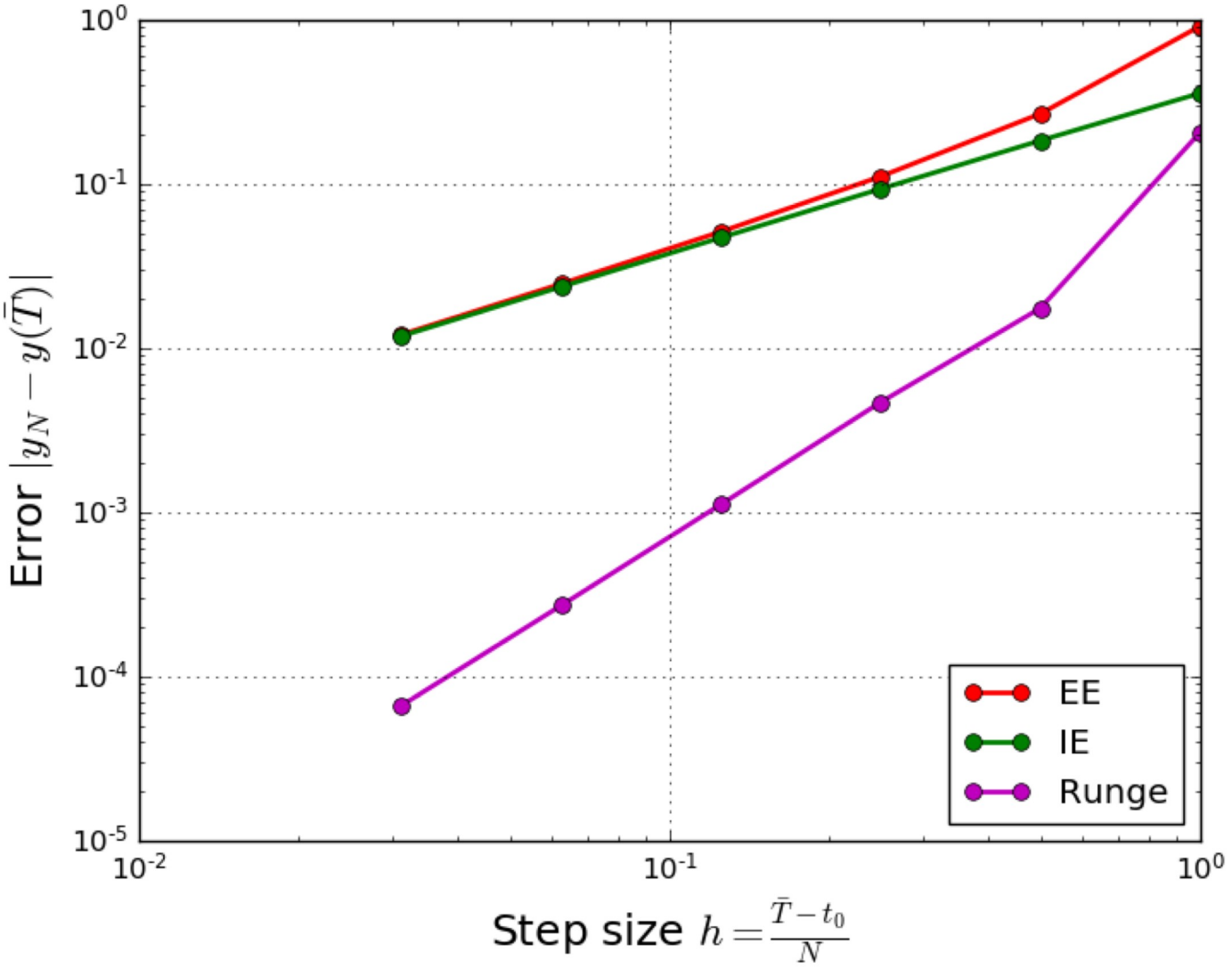
Ex. (10)

Runge‘s Methods

N	h	EE $\lvert y_N - y(\bar{T})\rvert$	IE $\lvert y_N - y(\bar{T})\rvert$	EM $\lvert y_N - y(\bar{T})\rvert$
4	2^{-0}	9.109E-01	3.559E-01	2.056E-01
8	2^{-1}	2.666E-01	1.830E-01	1.745E-02
16	2^{-2}	1.105E-01	9.295E-02	4.662E-03
32	2^{-3}	5.097E-02	4.688E-02	1.112E-03
64	2^{-4}	2.453E-02	2.354E-02	2.694E-04
128	2^{-5}	1.204E-02	1.180E-02	6.621E-05

Ex. (10)

Runge‘s Methods



Ex. (11)

Runge-Kutta Methods

Number of statges

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

Stages

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

Nodes

Runge-Kutta matrix

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

Weights

Runge-Kutta Methods

- Explicit Euler

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

- Implicit Euler

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

- Runge‘s

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

Zeros usually not written!

- Heun‘s

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Ex. (11)

Runge-Kutta Methods

- Implicit Midpoint

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

- Implicit Trapezoidal

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

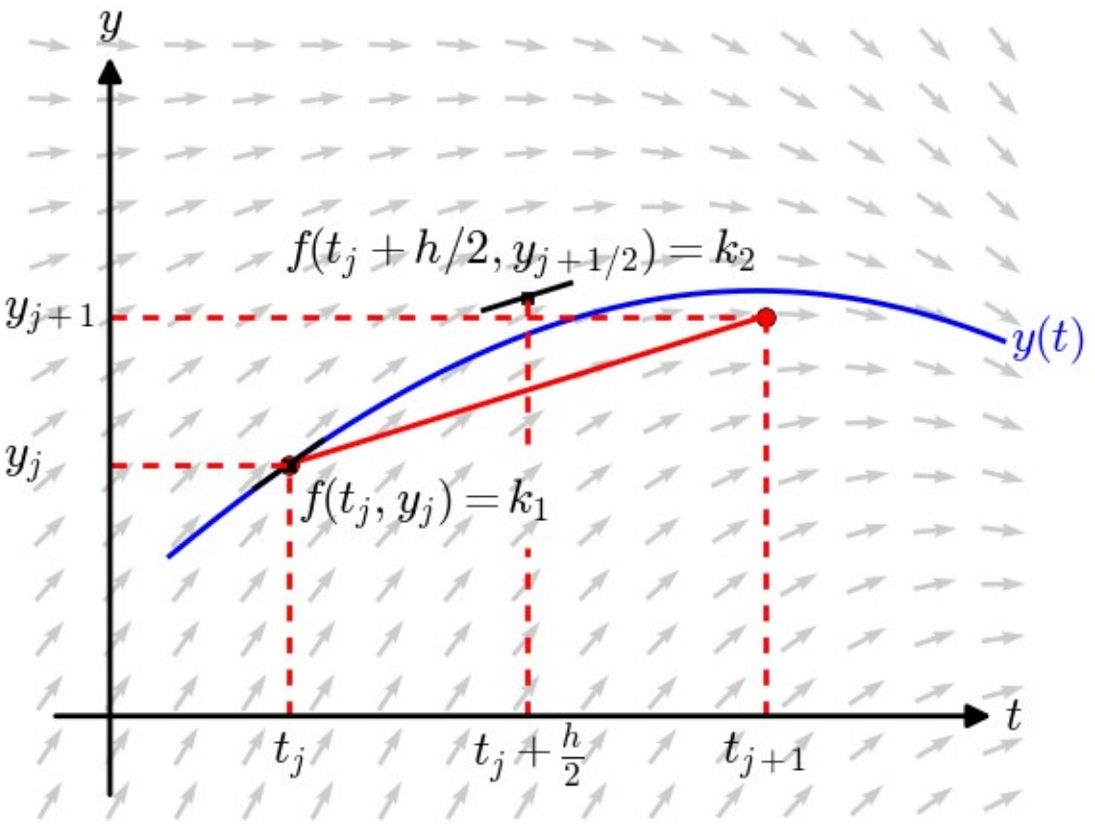
Zeros usually not written!

- Classic Runge-Kutta

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

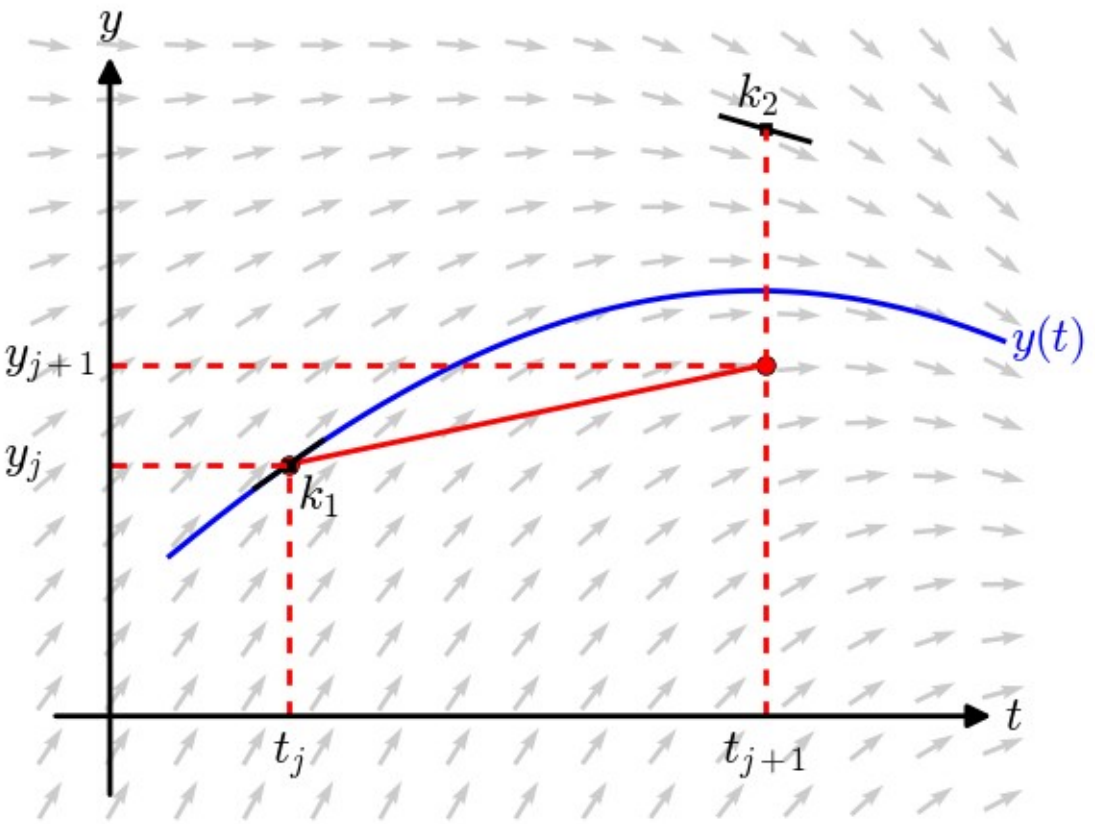
Ex. (11)

Runge‘s method



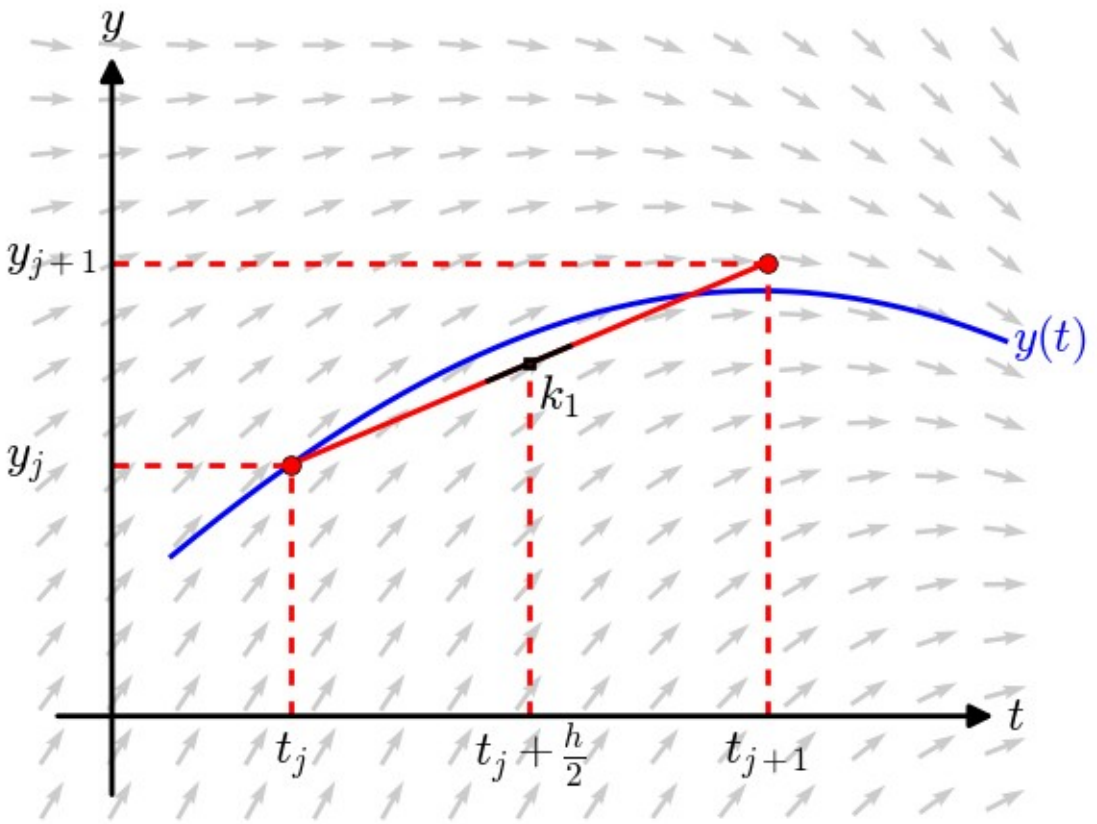
Ex. (11)

Heun‘s method



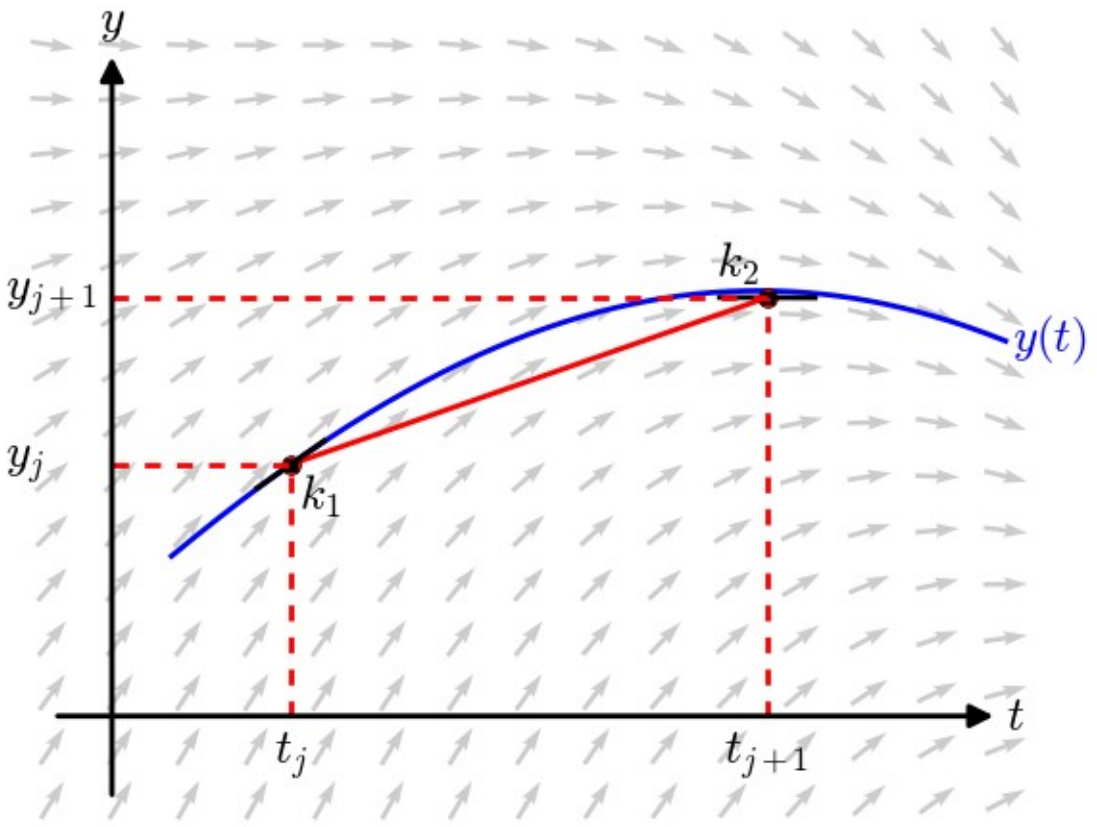
Ex. (11)

Implicit midpoint method



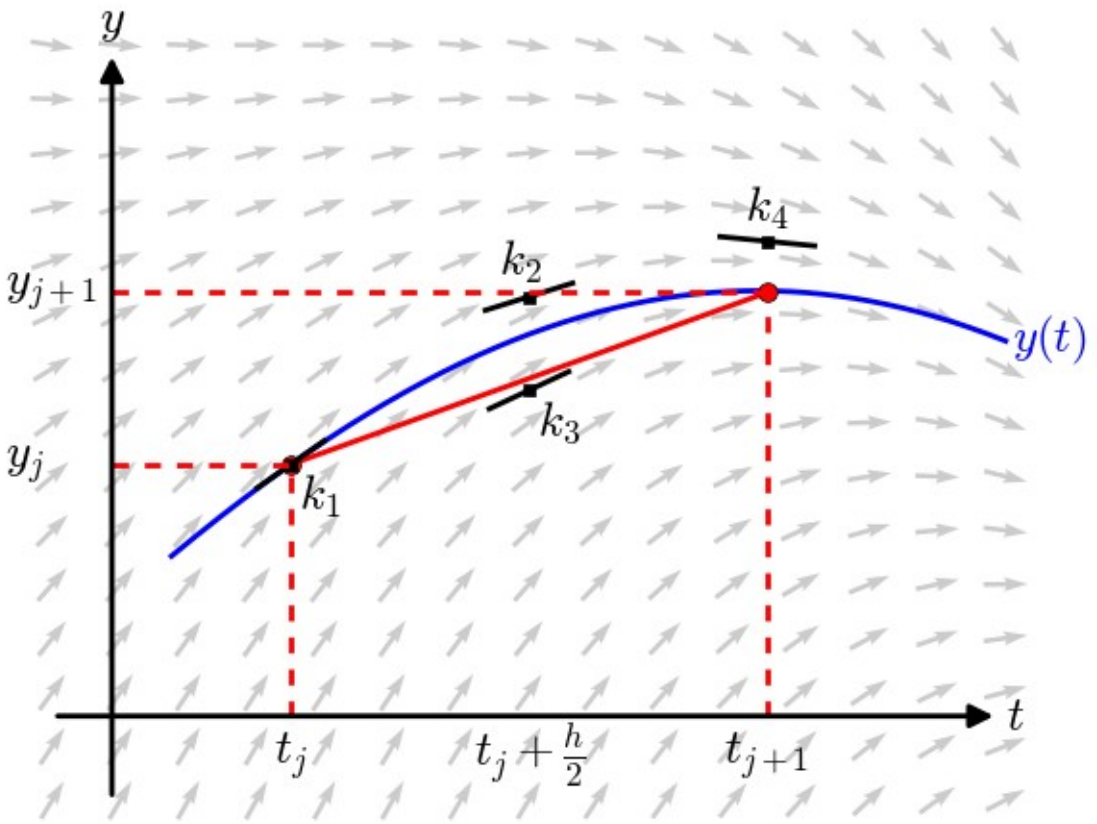
Ex. (11)

Implicit trapezoidal method



Ex. (11)

Classical Runge-Kutta (RK4)

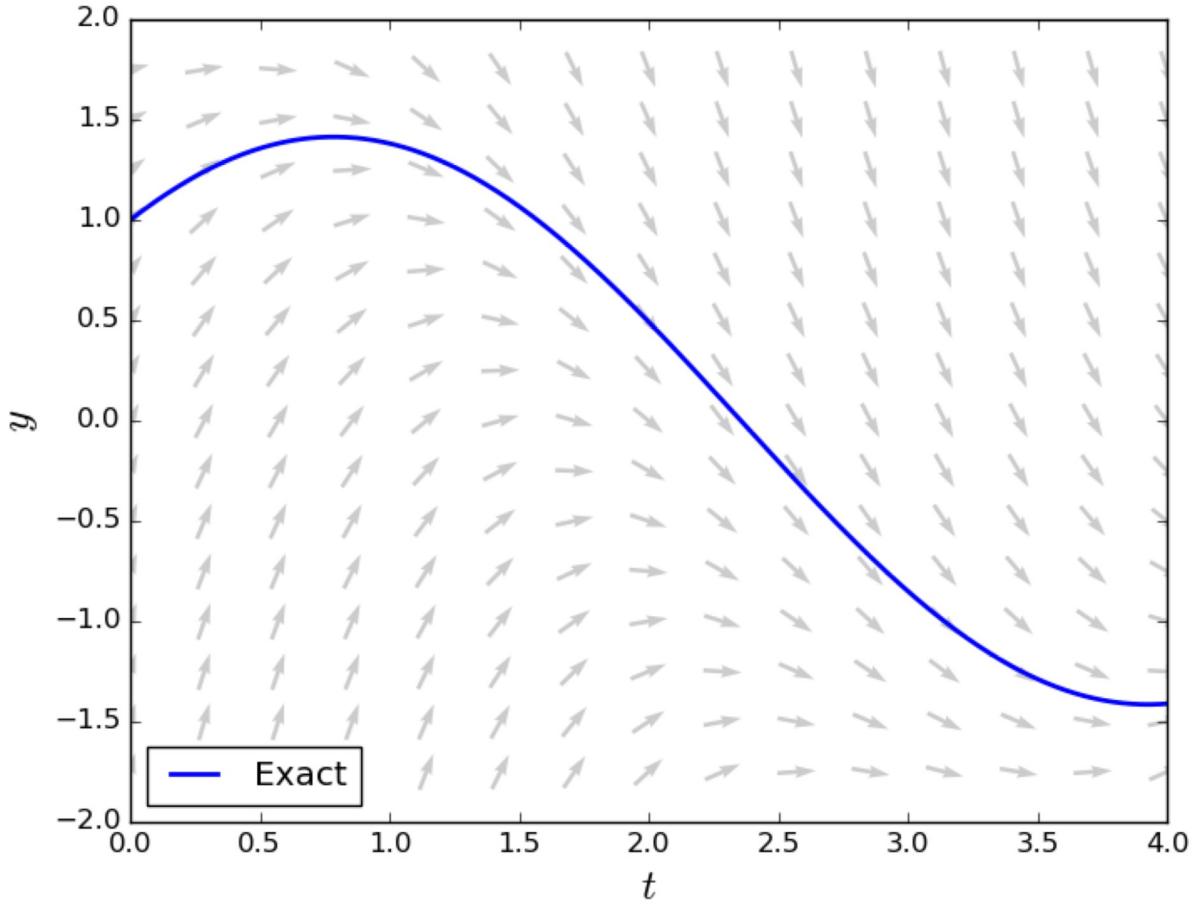


Ex. (12)

Example

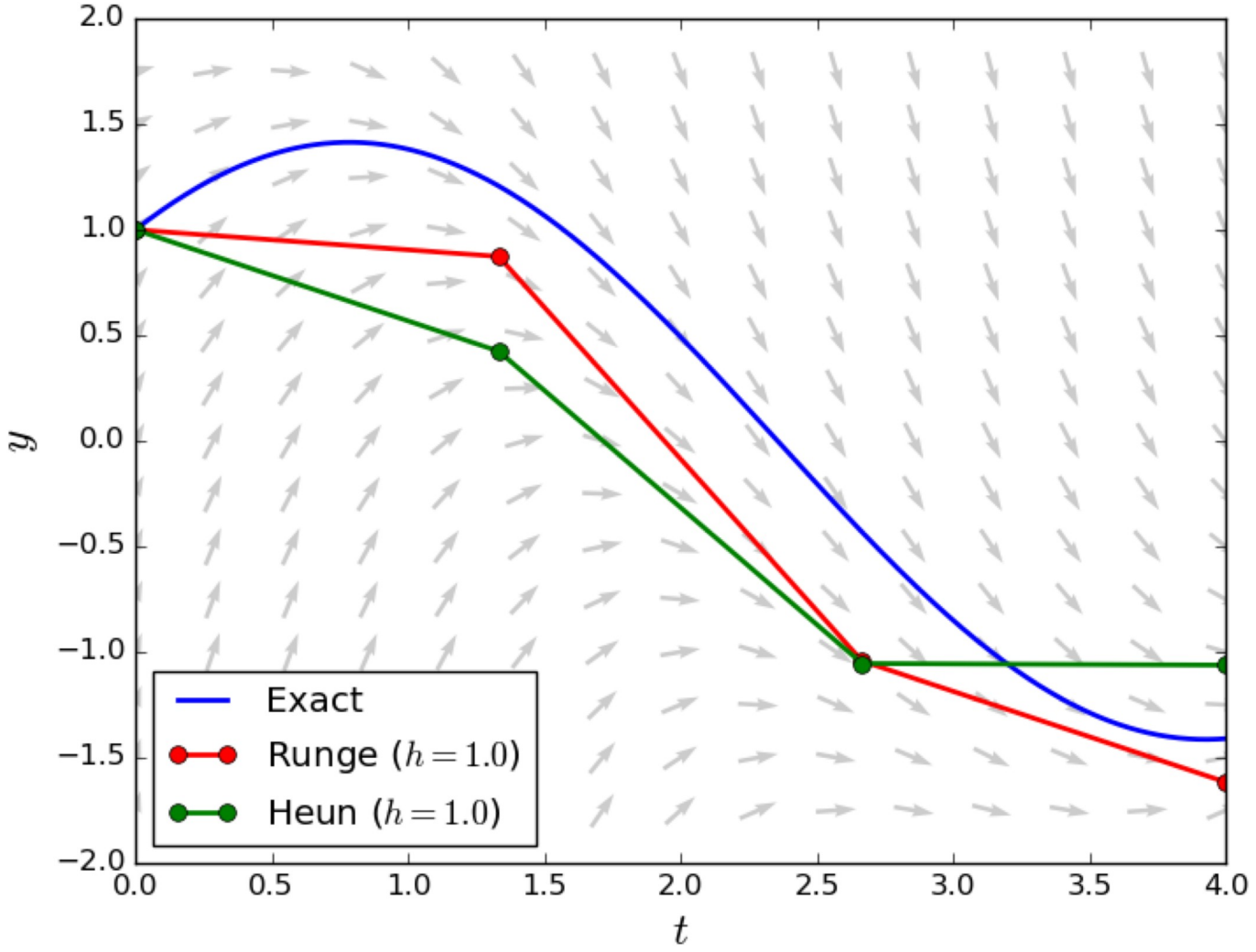
IVP $\begin{cases} \dot{y}(t) = -y(t) + 2\cos(t) \\ y(0) = 1 \end{cases}$

$$y(t) = \sin(t) + \cos(t)$$



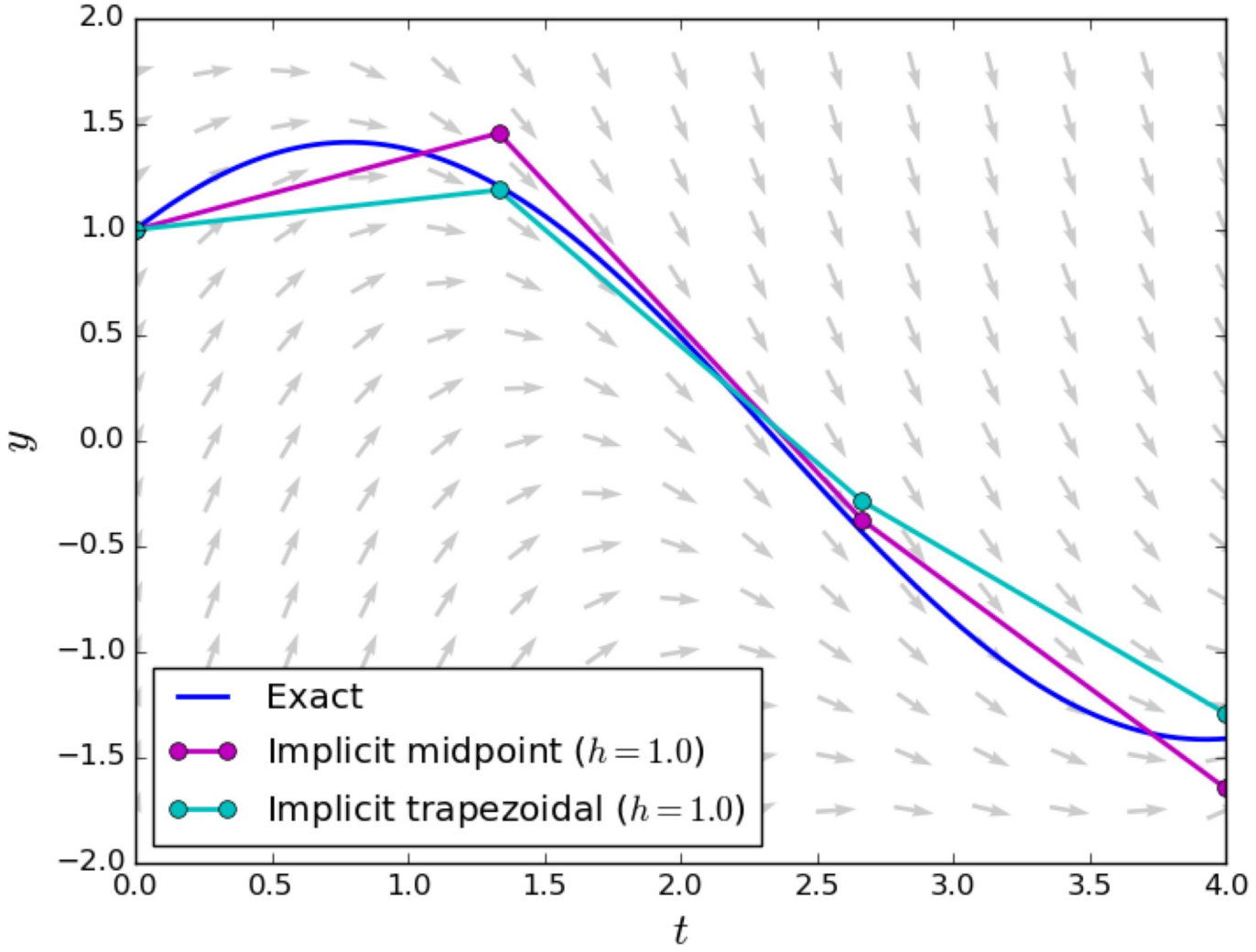
Ex. (12)

Runge‘s & Heun‘s method



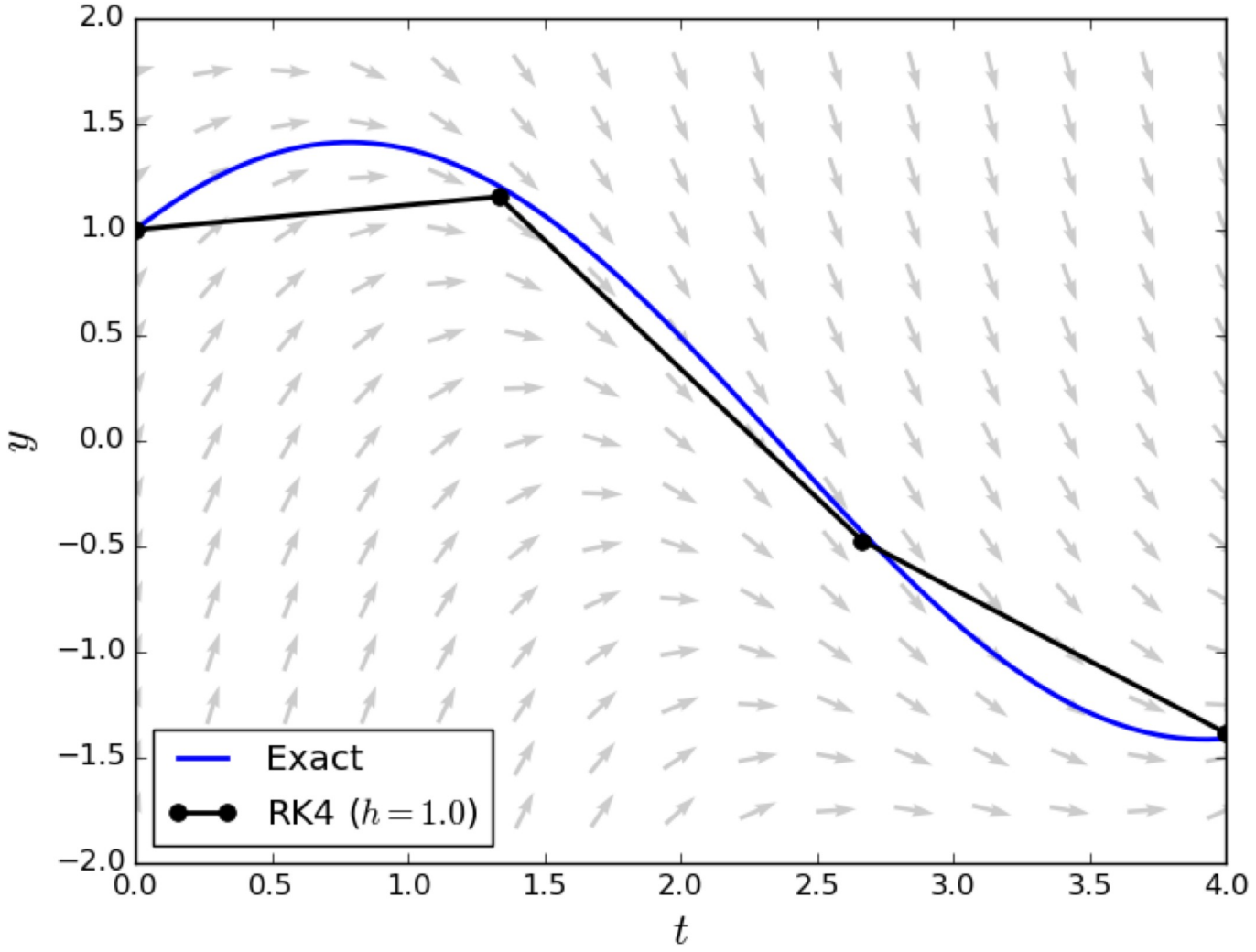
Ex. (12)

Implicit midpoint & trapezoidal



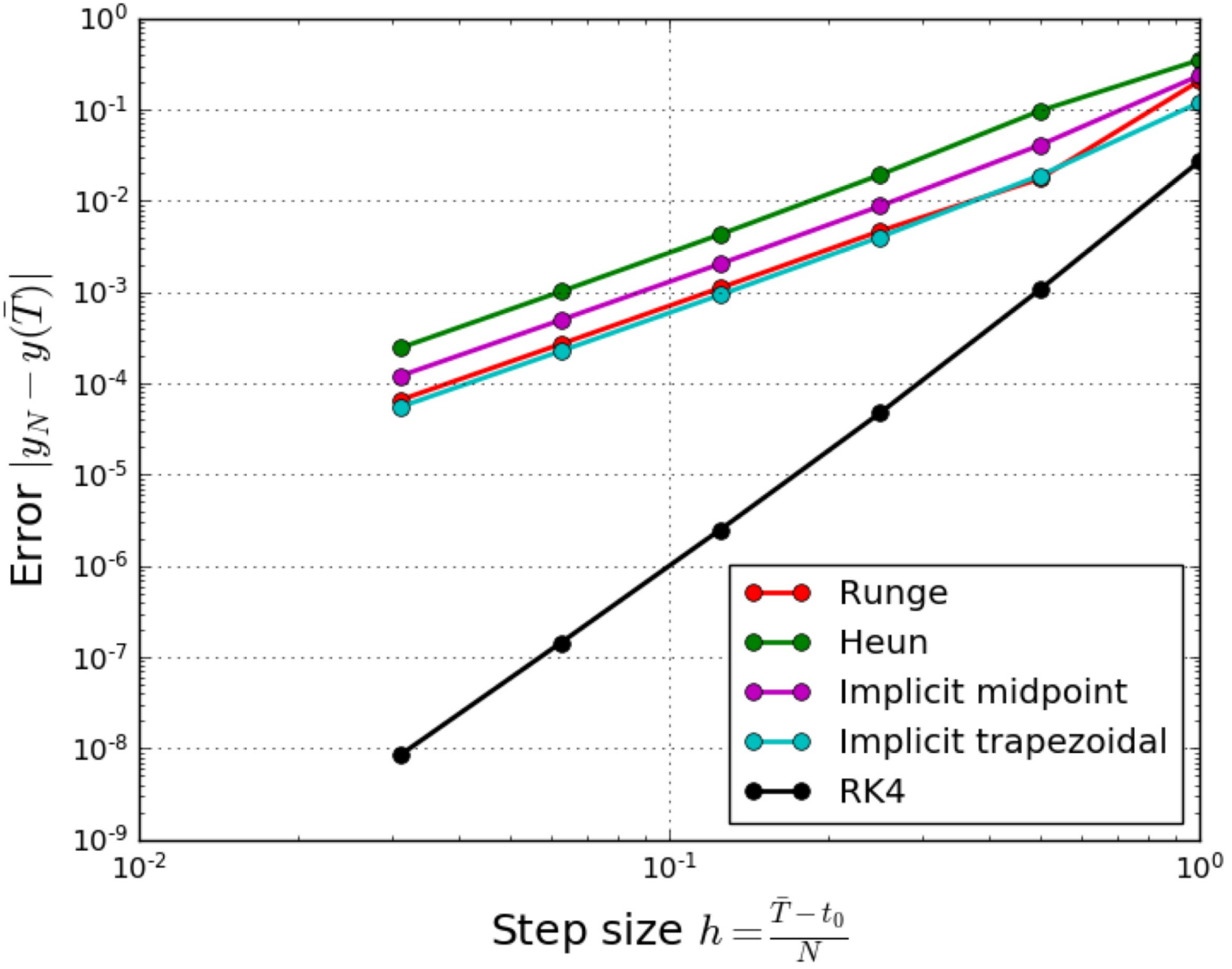
Ex. (12)

Runge-Kutta 4



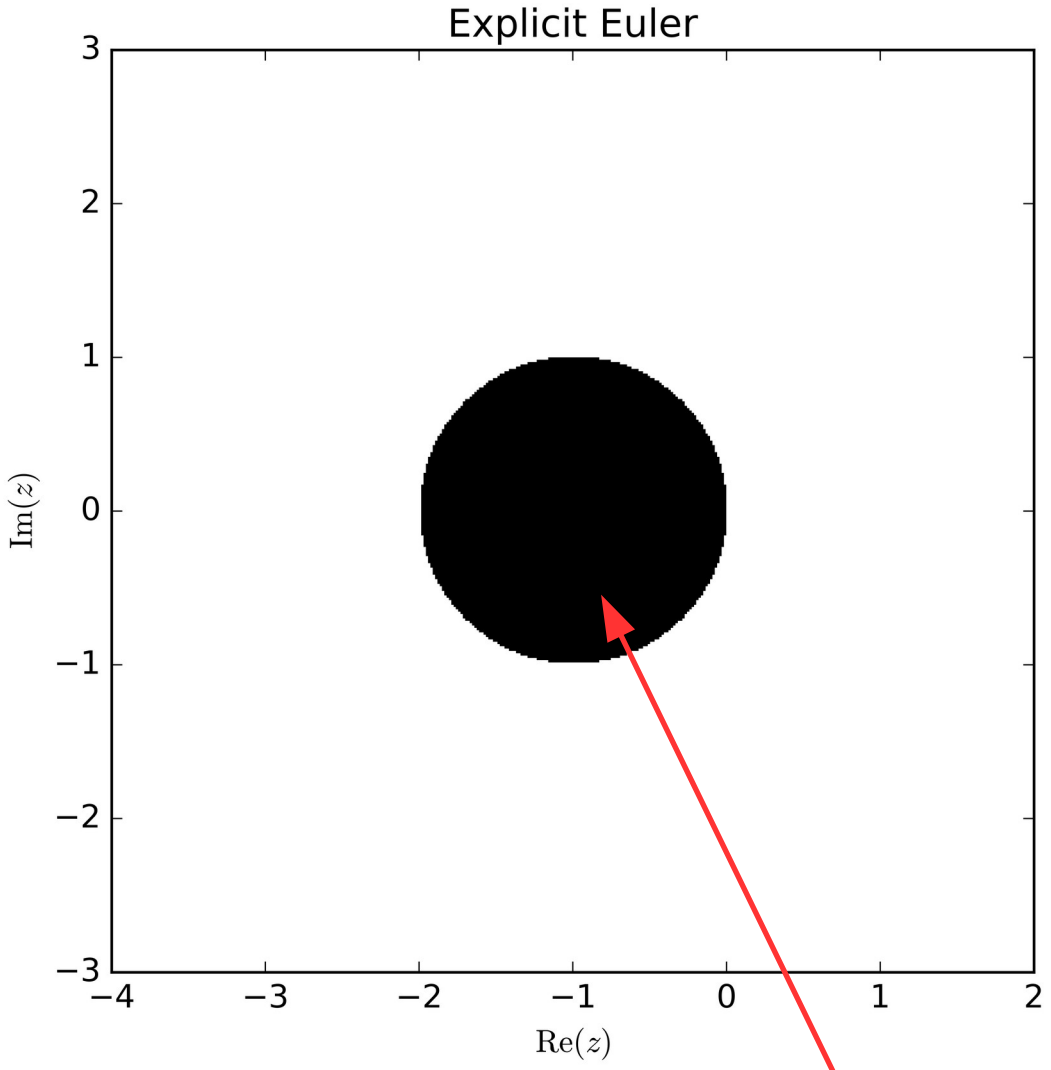
Ex. (12)

Errors


Error $|y_N - y(\bar{T})|$
Step size $h = \frac{\bar{T} - t_0}{N}$
Runge
Heun
Implicit midpoint
Implicit trapezoidal
RK4

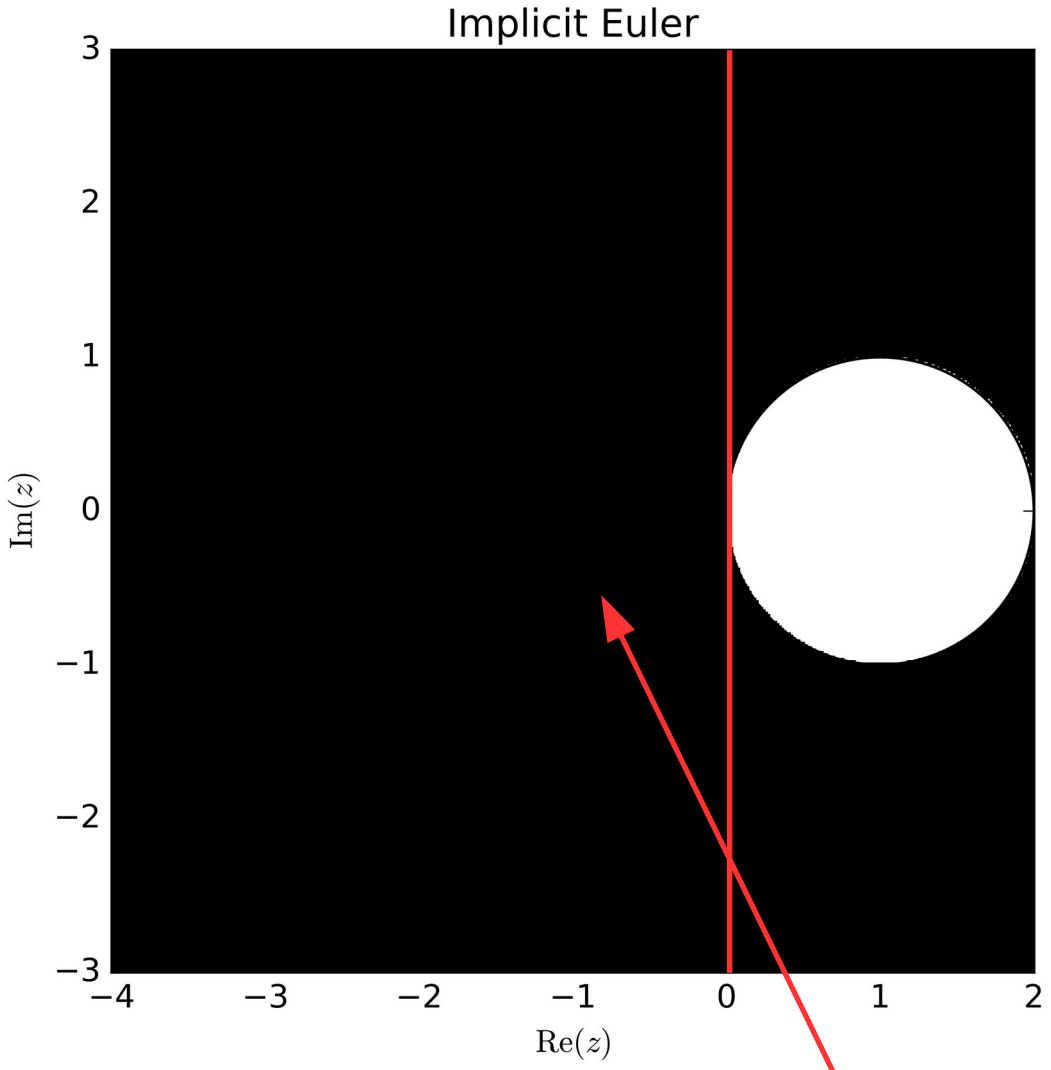
Ex. (15)

Stability Regions



Ex. (15)

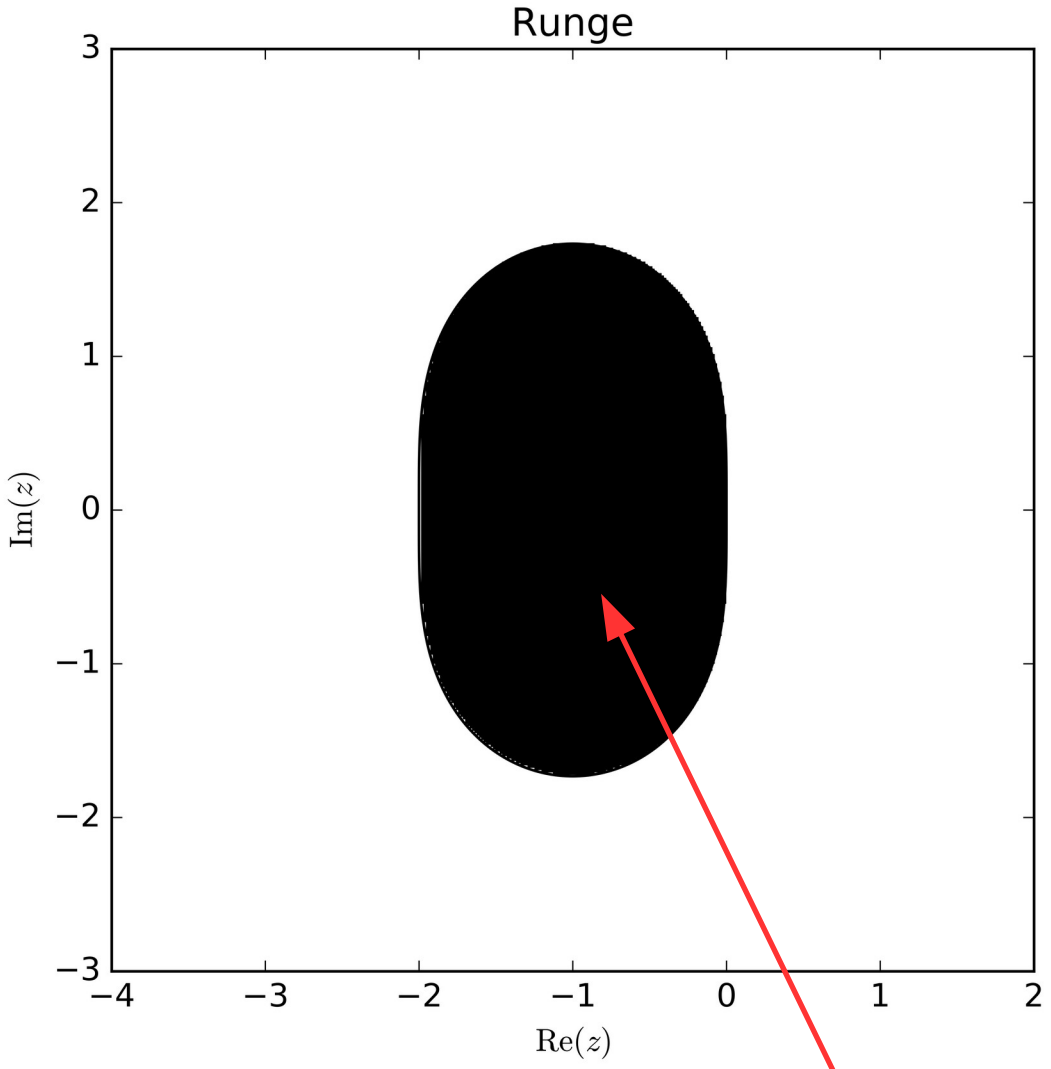
Stability Regions



Stability region

Ex. (15)

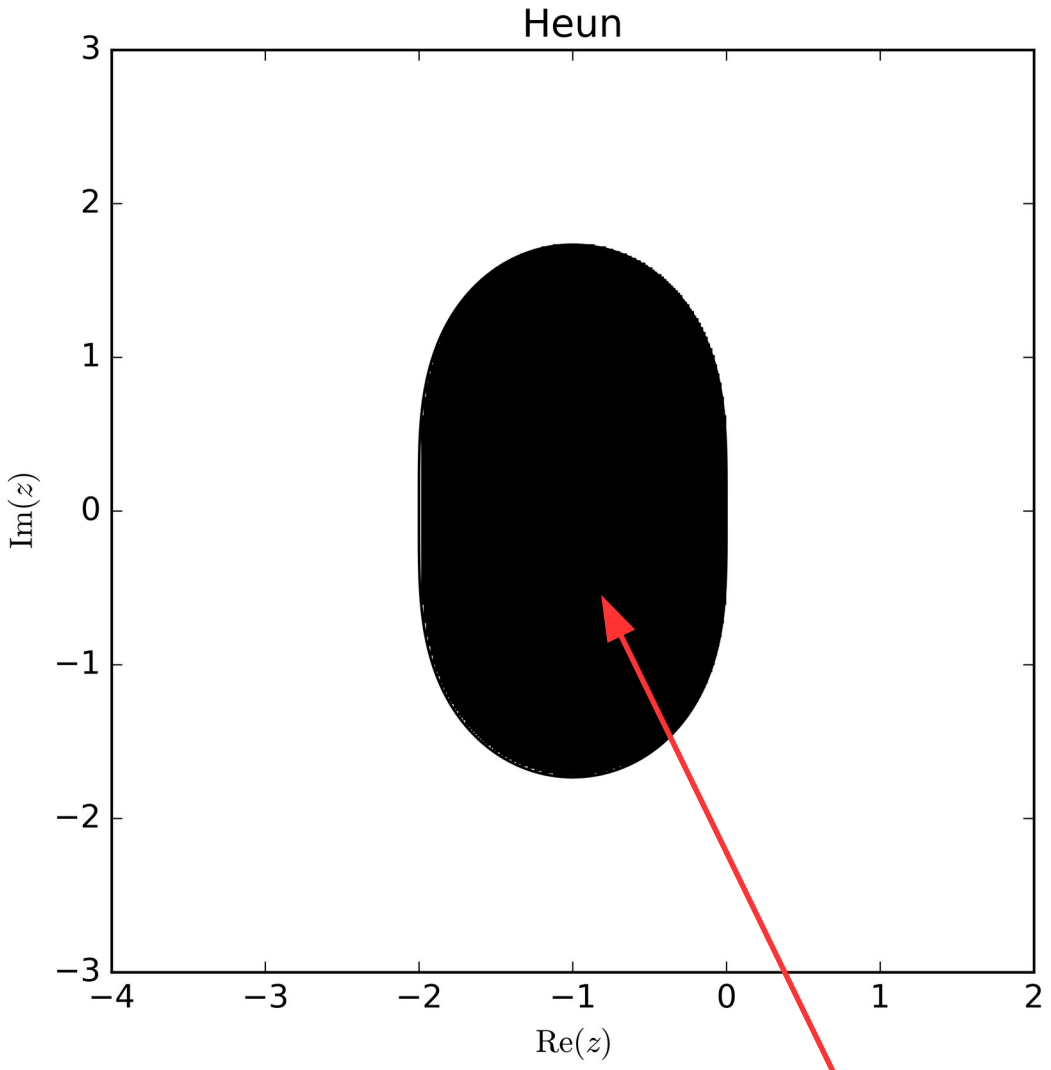
Stability Regions



Stability region

Ex. (15)

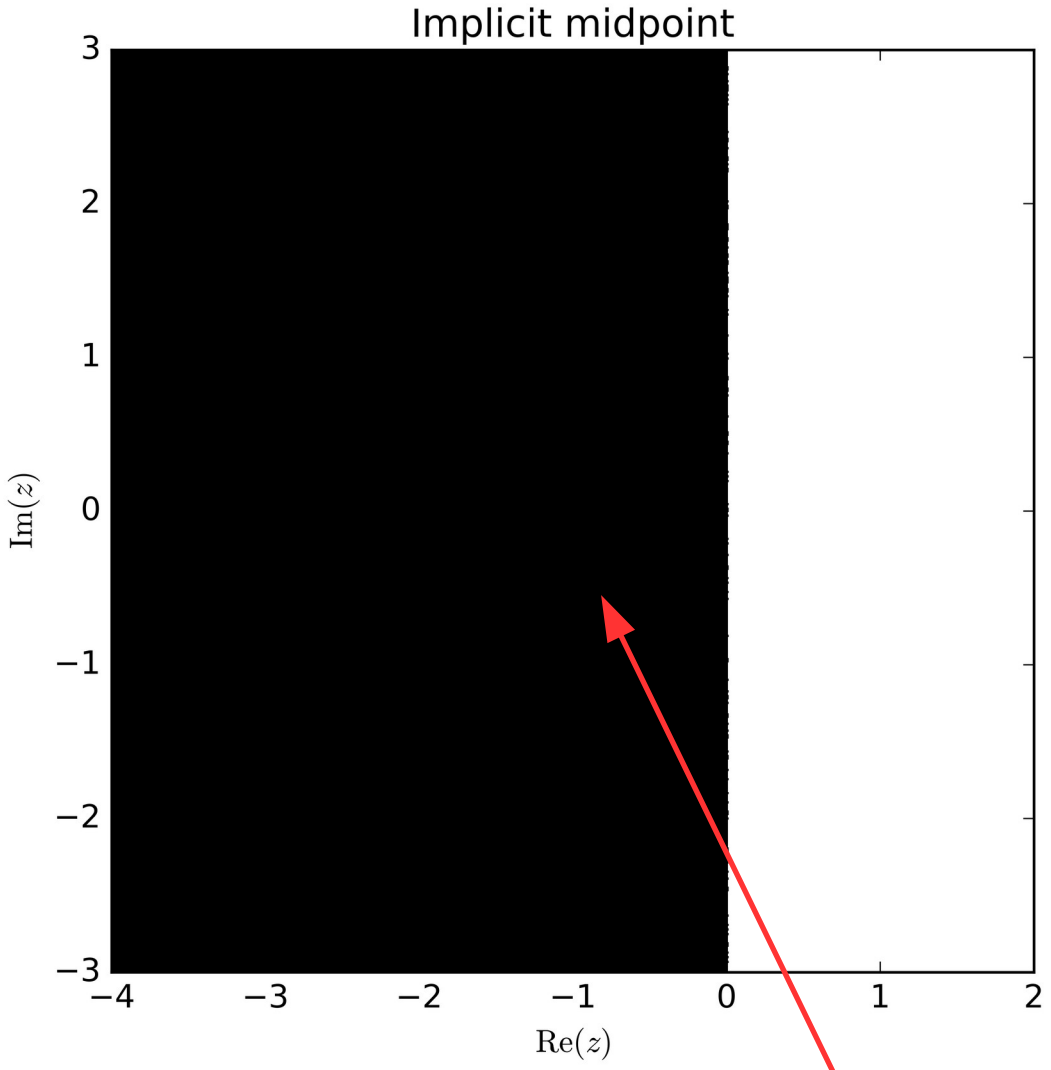
Stability Regions



Stability region

Ex. (15)

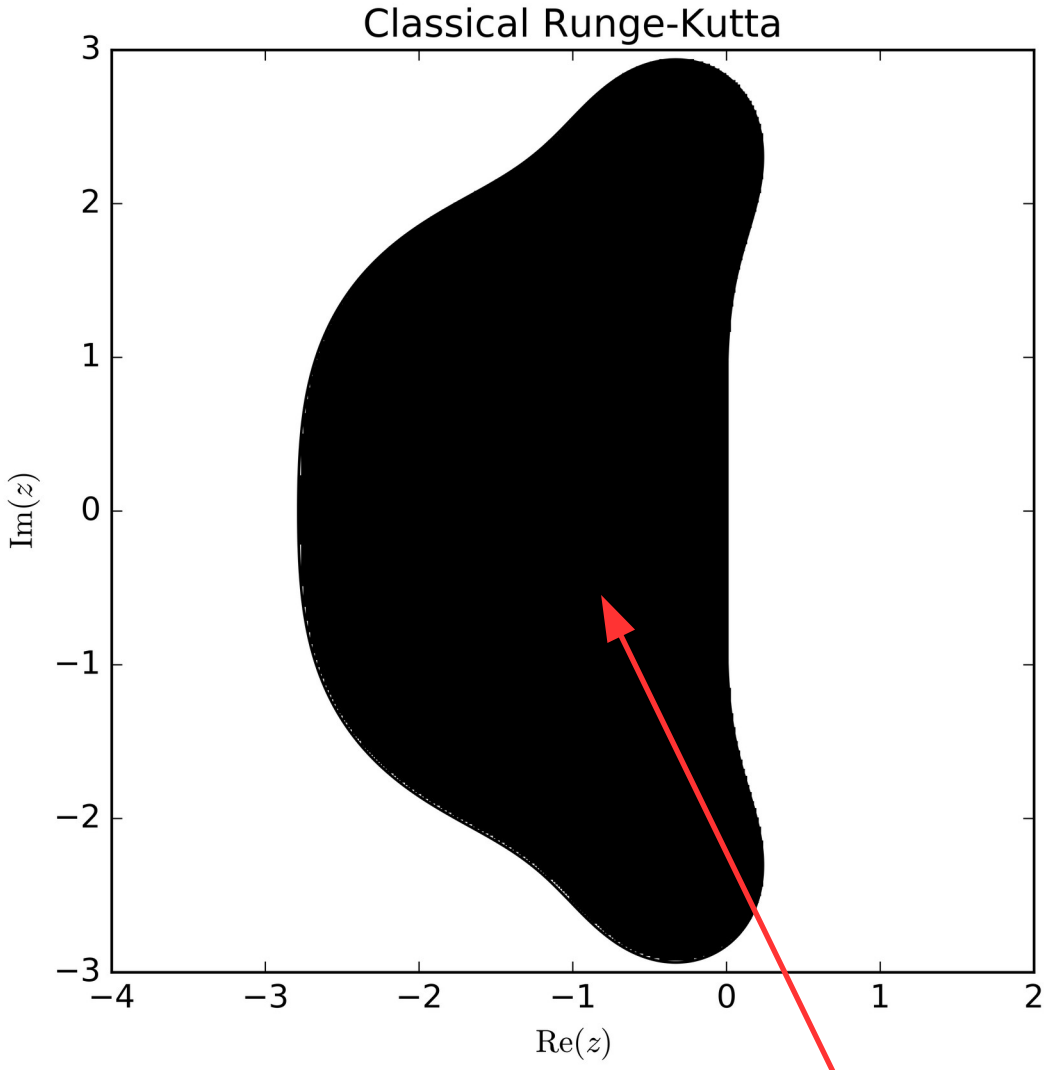
Stability Regions



Stability region

Ex. (15)

Stability Regions



Stability region

Ex. (16)

Stiff linear IVP

$$\dot{\mathbf{y}}(t) = A\mathbf{y}(t)$$

$$\mathbf{y} = \begin{pmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{pmatrix} \qquad A = \begin{pmatrix} -\frac{1}{2} & -\frac{869}{10} & \frac{1521}{5} \\ 0 & -\frac{227}{2} & \frac{591}{2} \\ 0 & \frac{591}{2} & -\frac{1803}{2} \end{pmatrix}$$

$$\mathbf{y}_0 = \begin{pmatrix} 1 \\ 6 \\ 2 \end{pmatrix} \qquad 0 \leq t \leq 2$$

Ex. (16)

Stiff linear IVP

$$\dot{\mathbf{y}}(t) = A\mathbf{y}(t)$$

$$= PDP^{-1}\mathbf{y}(t)$$

Diagonal!

I.e. „from the left“

$$\times P^{-1}$$

$$\mathrm{P}^{-1}\dot{\mathbf{y}}(t) = D\underbrace{P^{-1}\mathbf{y}(t)}_{\mathbf{z}(t)}$$

$$\dot{\mathbf{z}}(t) = D\mathbf{z}(t)$$

Ex. (16)

Stiff linear IVP

Simple computations (e.g. with CAS!):

$$D = \begin{pmatrix} -\frac{1}{2} & 0 & 0 \\ 0 & -15 & 0 \\ 0 & 0 & -1000 \end{pmatrix}$$

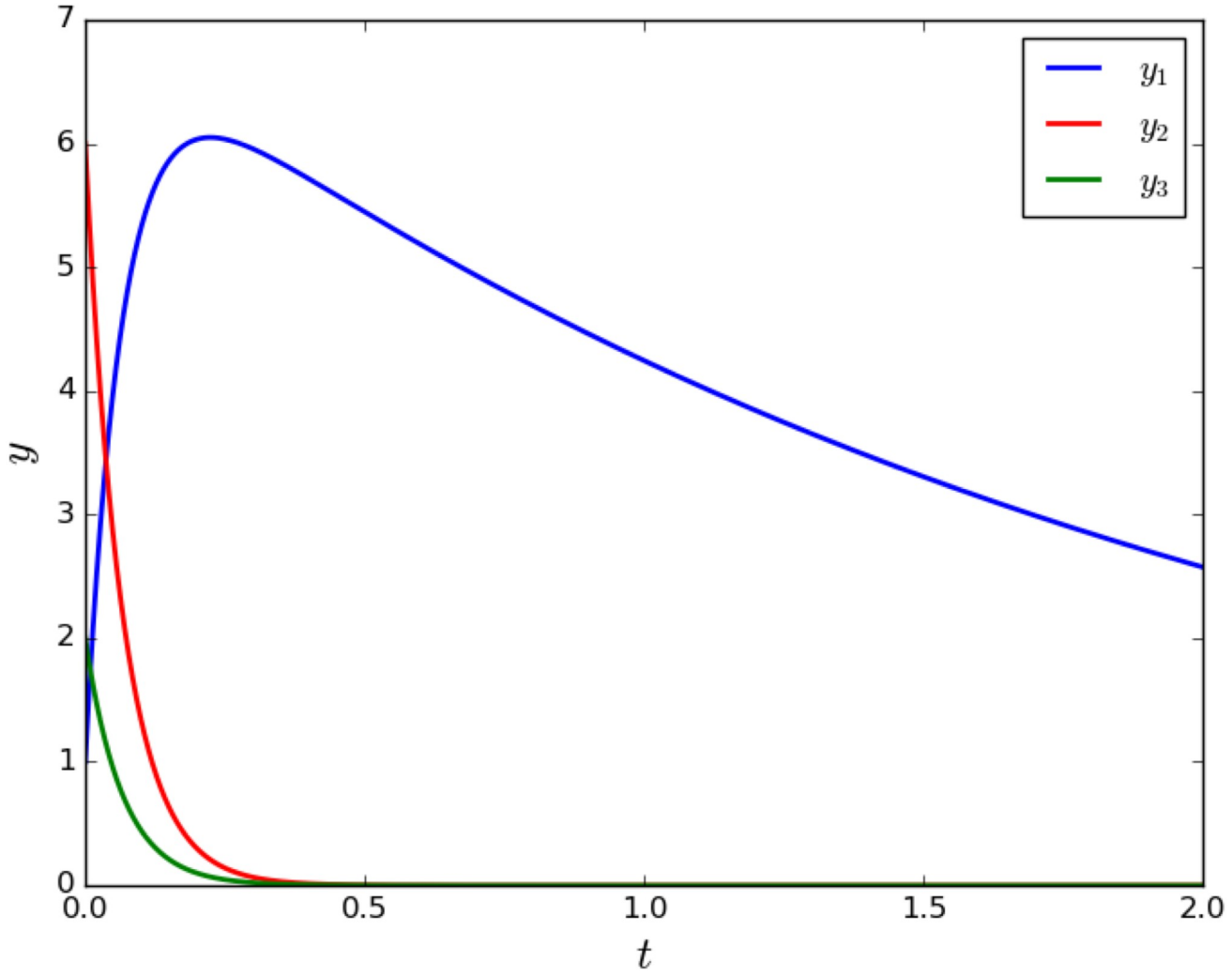
$$P = \begin{pmatrix} 15 & -12 & 1 \\ 0 & 12 & 1 \\ 0 & 4 & -3 \end{pmatrix} \qquad P^{-1} = \begin{pmatrix} \frac{1}{15} & \frac{4}{75} & \frac{1}{25} \\ 0 & \frac{3}{40} & \frac{1}{40} \\ 0 & \frac{1}{10} & -\frac{3}{10} \end{pmatrix}$$

Plugging in the IV gives:

$$\mathbf{y}(t) = \begin{pmatrix} 7 \\ 0 \\ 0 \end{pmatrix} e^{-\frac{1}{2}t} + \begin{pmatrix} -6 \\ 6 \\ 2 \end{pmatrix} e^{-15t} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} e^{-1000t}$$

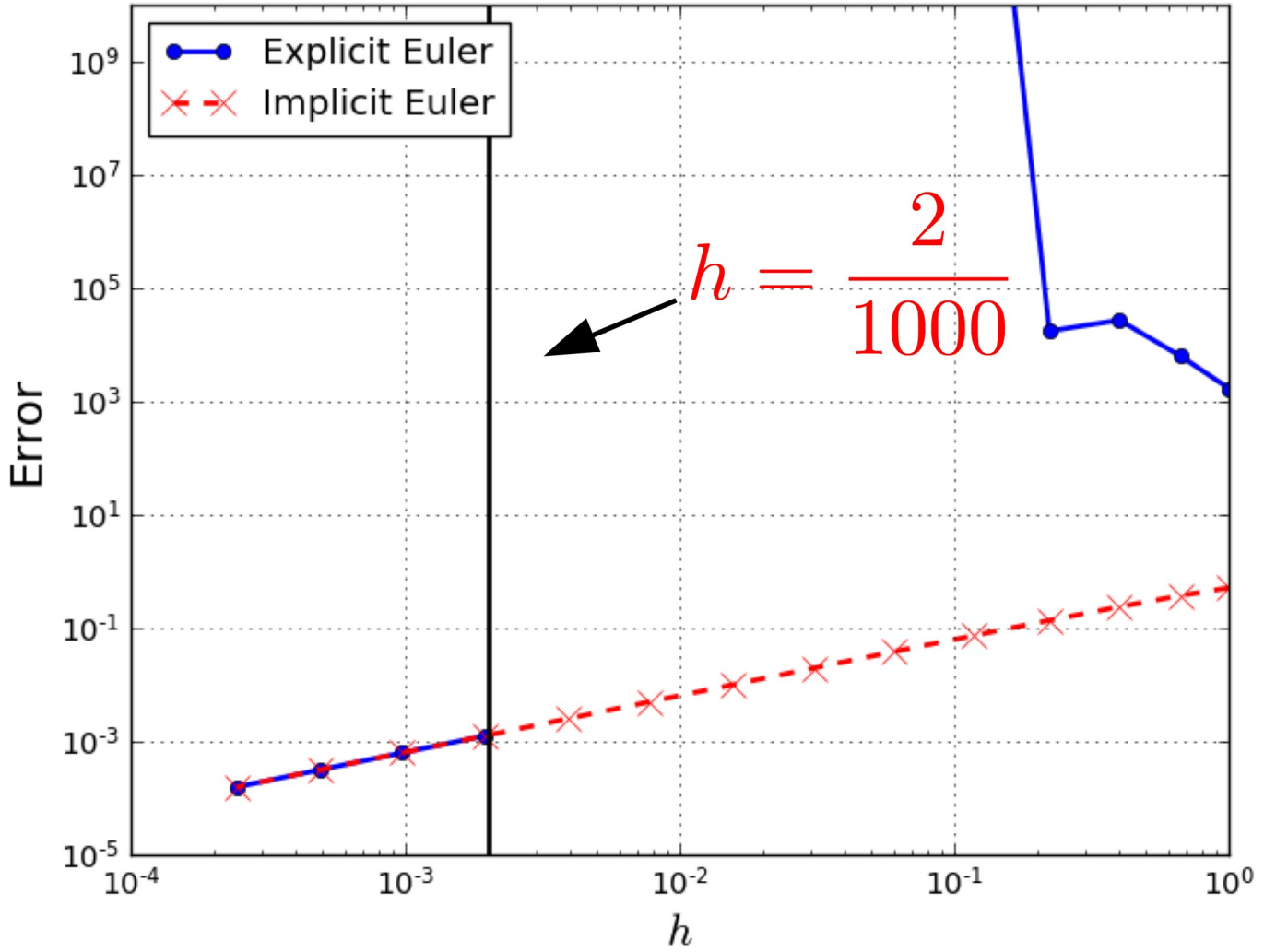
Ex. (16)

Stiff linear IVP



Ex. (16)

Stiff linear IVP



Ex. (16)

Stiff equations

A linear inhomogeneous system of ODEs

$$\dot{\vec{y}}(t) = \underbrace{A}_{n \times n \text{ Matrix}} \vec{y}(t) + \vec{b}(t)$$

is called stiff if the eigenvalues of A, λ_i $(i = 1, 2, \ldots, n)$, 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".

Ex. (17)

Stiff Nonlinear IVP

$$
\begin{aligned}
\dot{y}_A &= -0.1y_A + 100y_By_C \\
\dot{y}_B &= +0.1y_A - 100y_By_C - 500y_B^2 \\
\dot{y}_C &= + 500y_B^2 - 0.5y_C
\end{aligned}
$$

$$y_A(0) = 0.5 \; y_B(0) = 0.5 \; y_C(0) = 0.5$$

$$0 \leq t \leq 1$$

Ex. (17)

Stiff Nonlinear IVP

$$\begin{aligned} \dot{y}_A &= -0.1y_A + 100y_By_C \\ \dot{y}_B &= +0.1y_A - 100y_By_C - 500y_B^2 \\ \dot{y}_C &= +500y_B^2 - 0.5y_C \end{aligned}$$

NON-LINEAR!!!

$$\mathbf{y} = \begin{pmatrix} y_A \\ y_B \\ y_C \end{pmatrix} \qquad \mathbf{f}(t,\mathbf{y}) = \begin{pmatrix} -0.1y_A + 100y_By_C \\ +0.1y_A - 100y_By_C - 500y_B^2 \\ +500y_B^2 - 0.5y_C \end{pmatrix}$$

$$y_A(0) = 0.5 \; y_B(0) = 0.5 \; y_C(0) = 0.5 \qquad \mathbf{y}_0 = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

Ex. (17)

Local measure of stiffness

A local measure of stiffness is obtained by linearizing the system of ODEs at some point (of interest!) $t_n, \vec{y}_n$:

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

So one obtains the following inhomogeneous linear system of ODEs

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

Ex. (17)

Local measure of stiffness

Constant Matrix **Constant Vector**

$$\dot{\mathbf{y}}(t) = \overbrace{A(t_1, \mathbf{y}_1)}\mathbf{y}(t) + \overbrace{\mathbf{b}(t_1, \mathbf{y}_1)}$$

If this is stiff, then one says that the nonlinear system of ODEs is locally stiff around $(t_1, \vec{y}_1)$.

Stiff Stiff Nonlinear IVP

Stiff?

Let‘s linearize the RHS:

$$\mathbf{f}(t,\mathbf{y}) = \begin{pmatrix} -0.1y_A + 100y_By_C \\ +0.1y_A - 100y_By_C - 500y_B^2 \\ +500y_B^2 - 0.5y_C \end{pmatrix}$$

$$\mathbf{f}(t,\mathbf{y}) \approx \mathbf{f}(t_1,\mathbf{y}_1) + \underbrace{\frac{\partial \mathbf{f}}{\partial t}(t_1,\mathbf{y}_1)}_{0} + \underbrace{\frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_1,\mathbf{y}_1)}$$

$$J(t,\mathbf{y}) = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t,\mathbf{y}) = \begin{pmatrix} -0.1 & 100y_C & 100y_B \\ +0.1 & -1000y_B - 100y_C & -100y_B \\ 0 & 1000y_B & -0.5 \end{pmatrix}$$

Ex. (17)

Stiff Nonlinear IVP

Locally **Stiff?**

Let‘s linearize the RHS at the initial value: $\mathbf{y}_0 = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$

$$J(t_0, \mathbf{y}_0) = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_0, \mathbf{y}_0) = \begin{pmatrix} -0.1 & 50 & 50 \\ +0.1 & -550 & -50 \\ 0 & 500 & -0.5 \end{pmatrix}$$

MATLAB: eig ⟶

$$\lambda_1 \approx -5.00 \times 10^2$$
$$\lambda_2 \approx -9.87 \times 10^{-4}$$
$$\lambda_3 \approx -5.07 \times 10^1$$

$$S = \frac{\max_j |\operatorname{Re}(\lambda_j)|}{\min_j |\operatorname{Re}(\lambda_j)|} \approx 5.06 \times 10^5$$

Yes!!!

Error control and adaptive step size

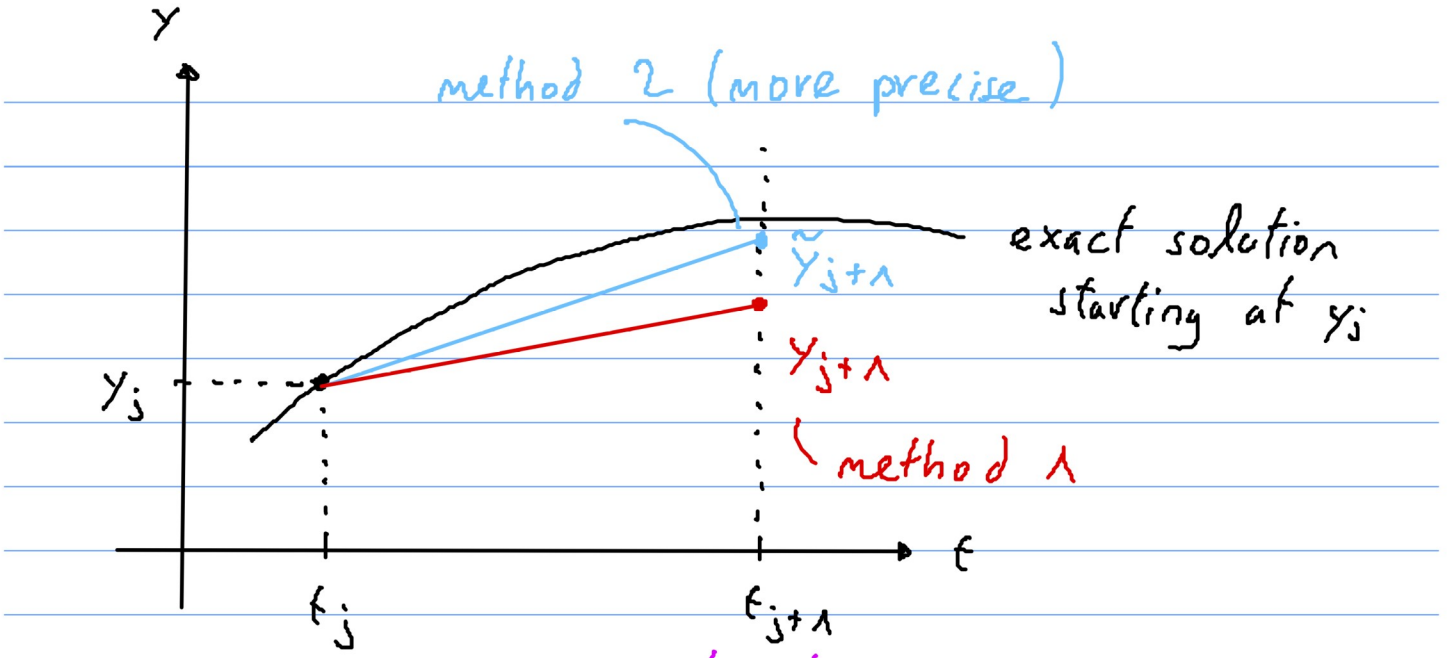
In practice, one wants to choose the step size (the h!) sufficiently small to ensure some required precision and sufficiently large to avoid unnecessary computational work.

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 control and adaptive step size



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

Error control and adaptive step size

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

Error control and adaptive step size

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

⇝ slides

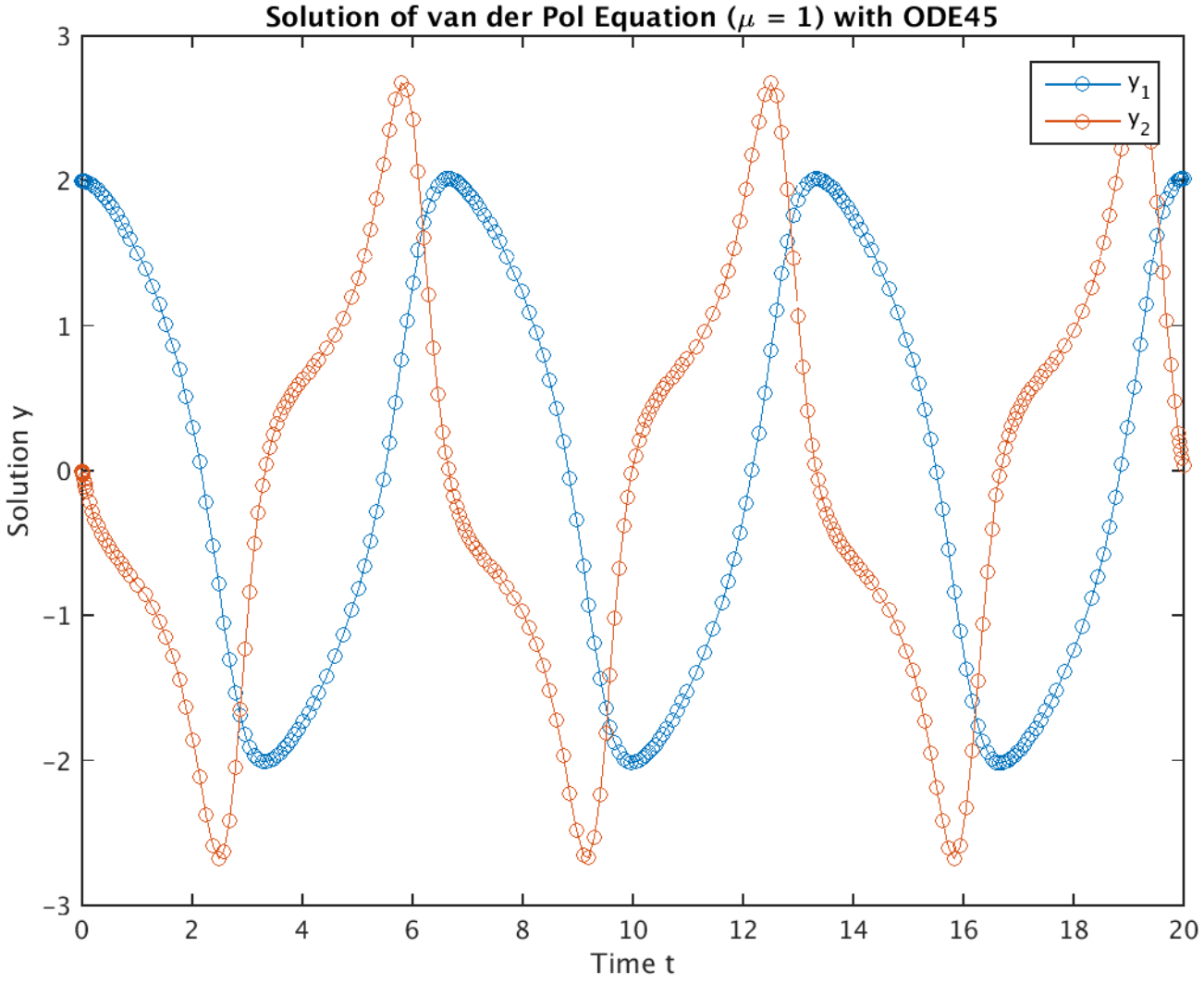
(19) " with $\mu = 1000$ and $t \in [0, 3000]$ (locally) makes the problem very stiff!

⇝ slides & exercises

Ex. (18)

Adaptive step size

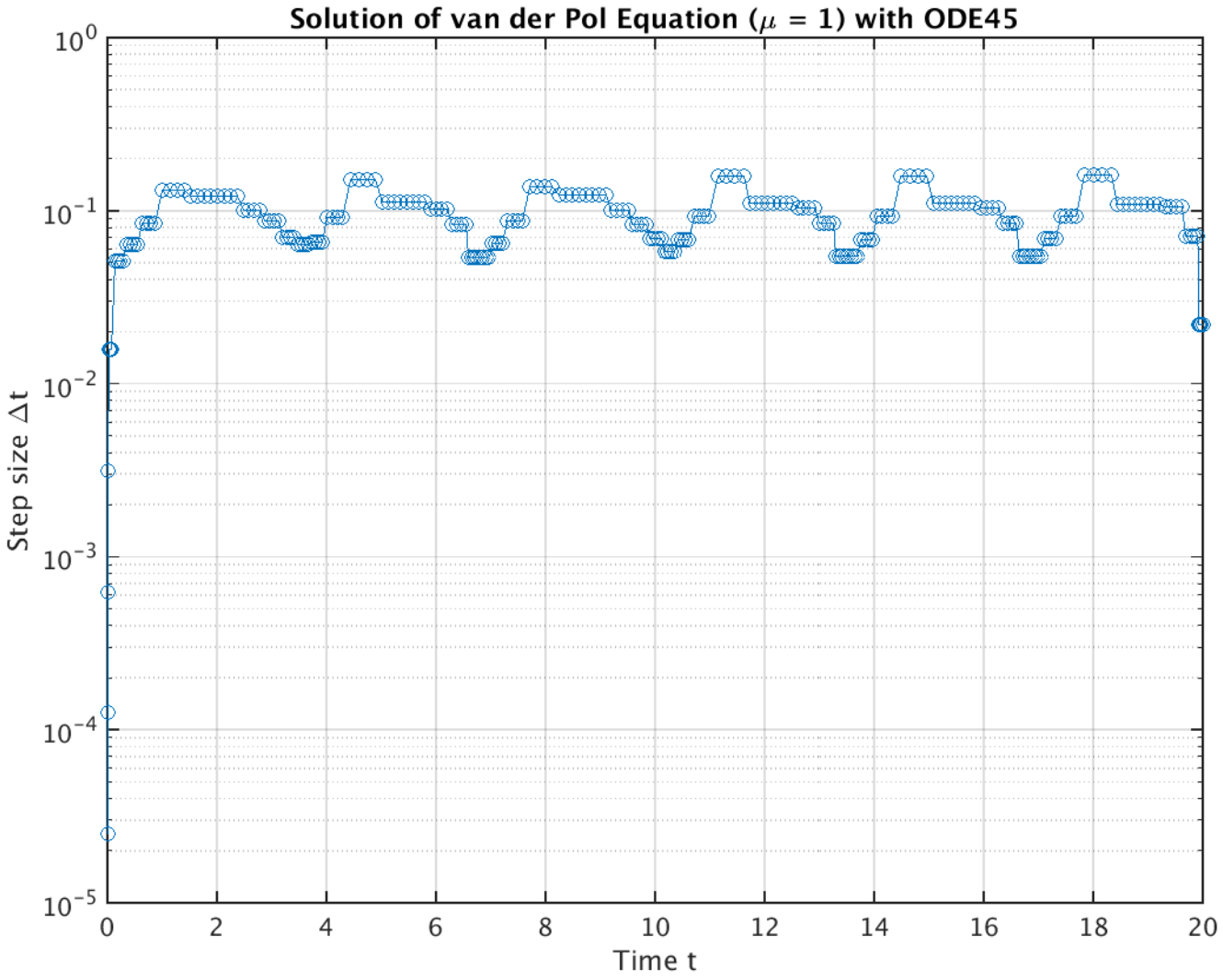
$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = \mu(1 - y_1^2)y_2 - y_1 \qquad \mu = 1 \qquad \mathbf{y}_0 = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \qquad t \in [0, 20]$$



Ex. (18)

Adaptive step size

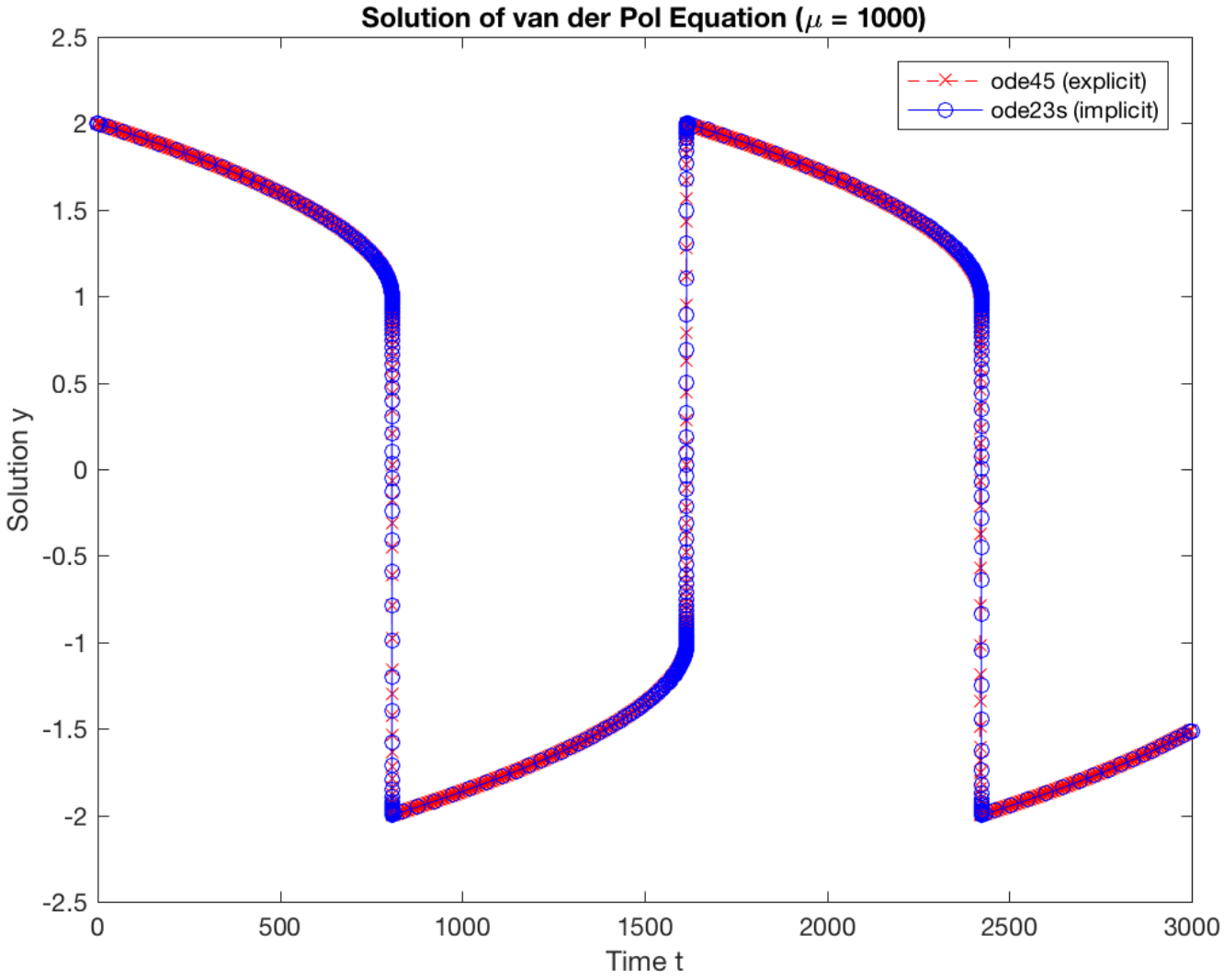
$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = \mu(1 - y_1^2)y_2 - y_1 \qquad \mu = 1 \qquad \mathbf{y}_0 = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \qquad t \in [0, 20]$$



Ex. (19)

Adaptive step size

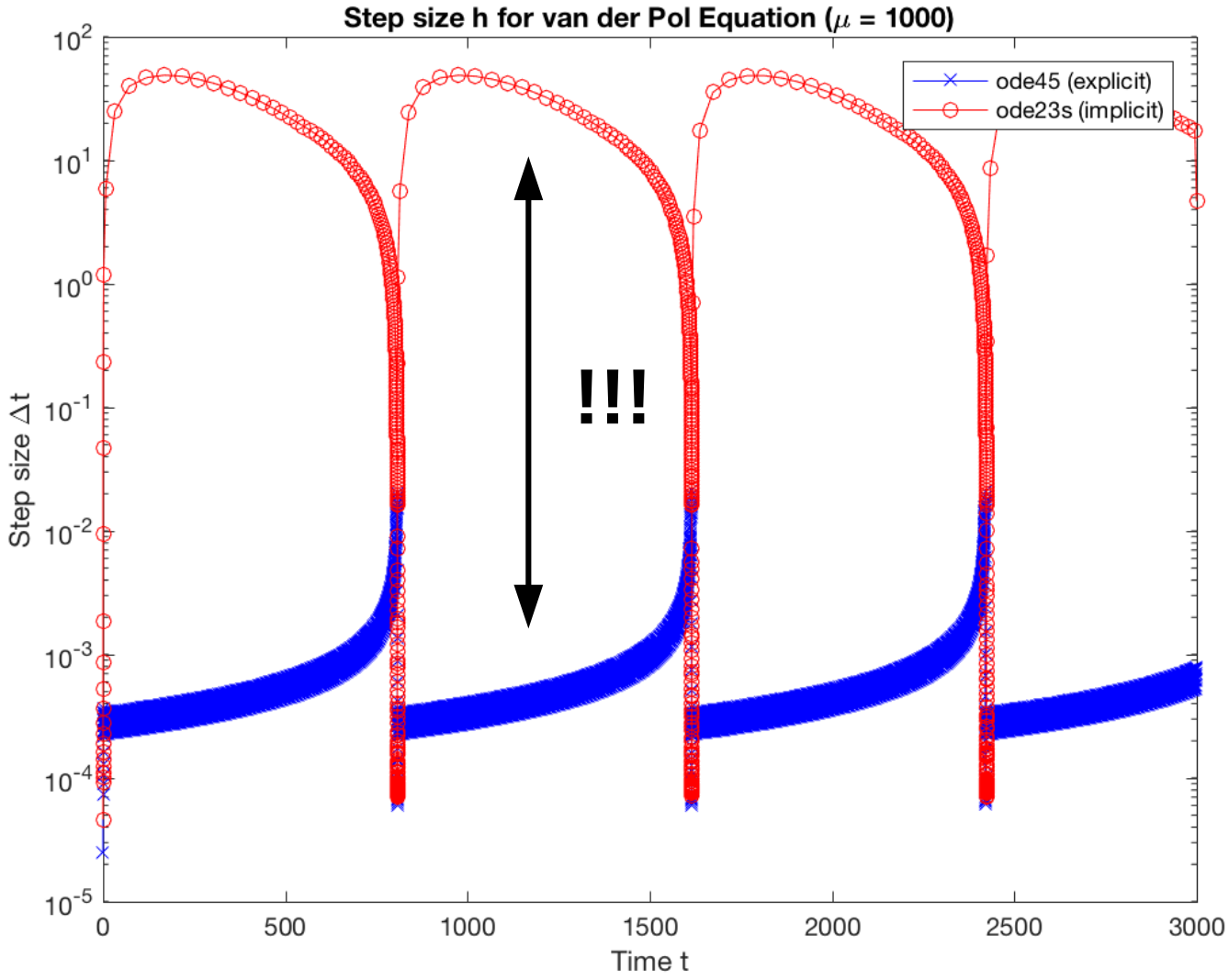
$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = \mu(1 - y_1^2)y_2 - y_1 \qquad \mu = 1000 \qquad \mathbf{y}_0 = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \qquad t \in [0, 3000]$$



Ex. (19)

Adaptive step size

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = \mu(1 - y_1^2)y_2 - y_1 \qquad \mu = 1000 \qquad \mathbf{y}_0 = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \qquad t \in [0, 3000]$$



Ex. (19)

Adaptive step size

$$\dot{y}_1 = y_2$$
$$\dot{y}_2 = \mu(1 - y_1^2)y_2 - y_1 \qquad \mu = 1000 \qquad \mathbf{y}_0 = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \qquad t \in [0, 3000]$$