Cyber-Physical Systems – Modeling and Simulation of Continuous Systems

Matthias Althoff

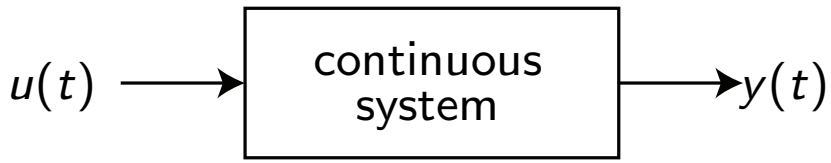
TU München

29. May 2015

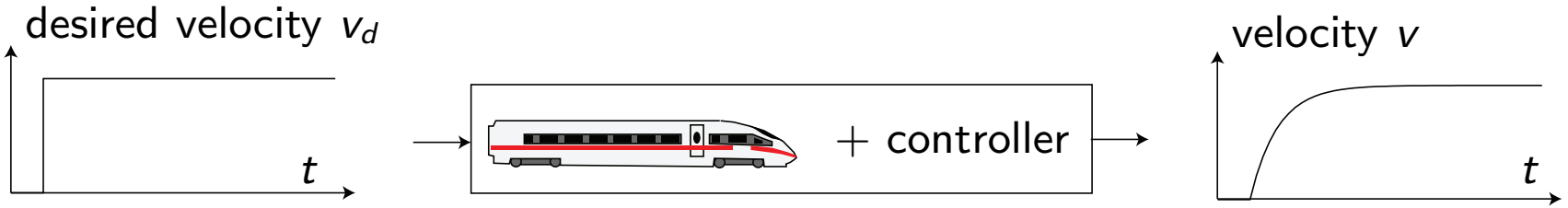
Continuous Systems

Continuous systems describe the dynamics of systems that can be described by a set of continuous states (e.g. position, temperature, etc.). Starting from an initial state vector $x(t_0)$ and for a given input signal $u(t)$, a continuous system creates a continuous output signal $y(t)$.

- We always assume that the state, the input, and the output are vectors of proper dimension.
- Static systems are special cases, where the output directly follows from a function of the input: $y(t) = g(u(t))$.



Train example from the first lecture:



Continuous System in State Space Form

Definition

A continuous system in state space form C is a tuple (ordered set):

$$\mathrm{C} = (\mathcal{X}, \mathcal{U}_c, \mathcal{Y}_c, f, g, x(0)),$$

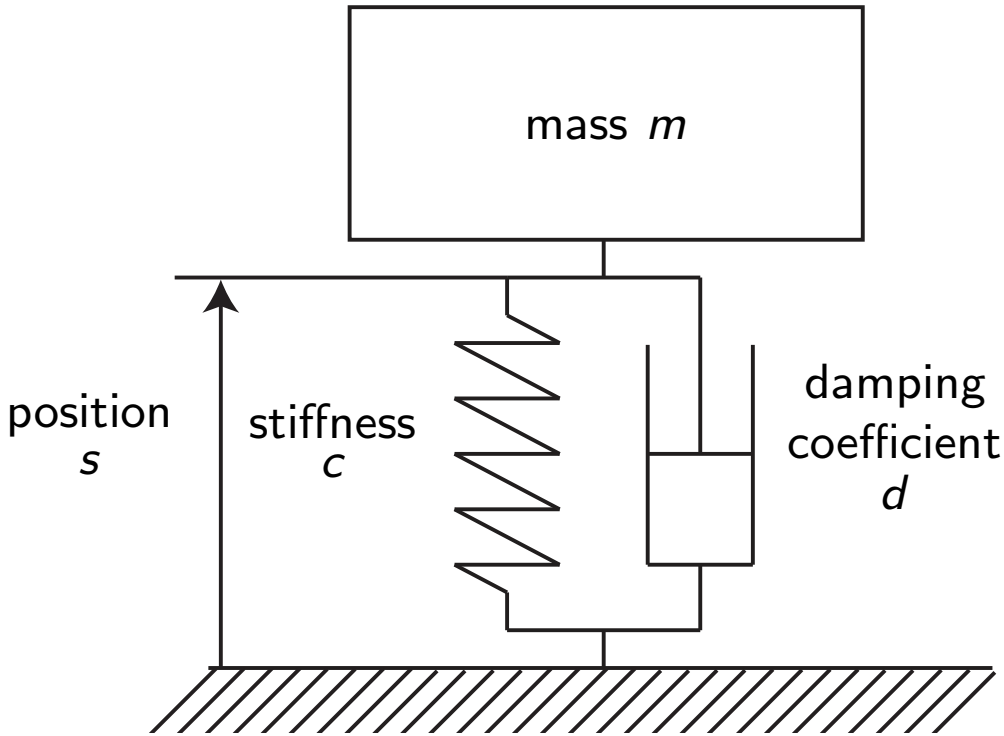
where $x(0) \in \mathbb{R}^n$ is the initial state,

$\mathcal{X} \subseteq \mathbb{R}^n$	set of states
$\mathcal{U}_c \subseteq \mathbb{R}^m$	set of continuous inputs
$\mathcal{Y}_c \subseteq \mathbb{R}^o$	set of continuous outputs

and

$f : \mathcal{X} \times \mathcal{U}_c \to \mathcal{X}$	flow function	$\dot{x}(t) = f(x(t), u(t))$
$g : \mathcal{X} \times \mathcal{U}_c \to \mathcal{Y}_c$	output function	$y(t) = g(x(t), u(t))$

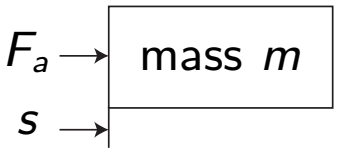
Example I: Mass Spring Damper (MSD) System



Equations of Each Component (MSD System)

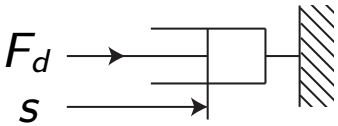
Variables: s (position), $v = \dot{s}$ (velocity), $a = \dot{v} = \ddot{s}$ (acceleration), F_e (external force);
Parameters: m (mass), d (damping coefficient), c (stiffness), s_0 (position of relaxed spring)

Mass



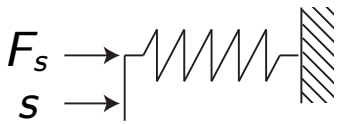
$$F_a = m\,a = m\,\ddot{s}.$$

Damper



$$F_d = d\,v = d\,\dot{s}.$$

Spring



$$F_s = c\,(s - s_0).$$

Sum of forces equals 0: $m\,\ddot{s} + d\,\dot{s} + c\,(s - s_0) + F_e = 0$

Transformation to State Space Form (MSD System)

The differential equation

$$m\,\ddot{s} + d\,\dot{s} + c\,(s - s_0) + F_e = 0$$

is of second order. To transform it in state space form we need a set of first order differential equations. This is achieved by introducing a new state variable for each higher derivative that has to be lowered:

$$x_1 := s$$

$$x_2 := v$$

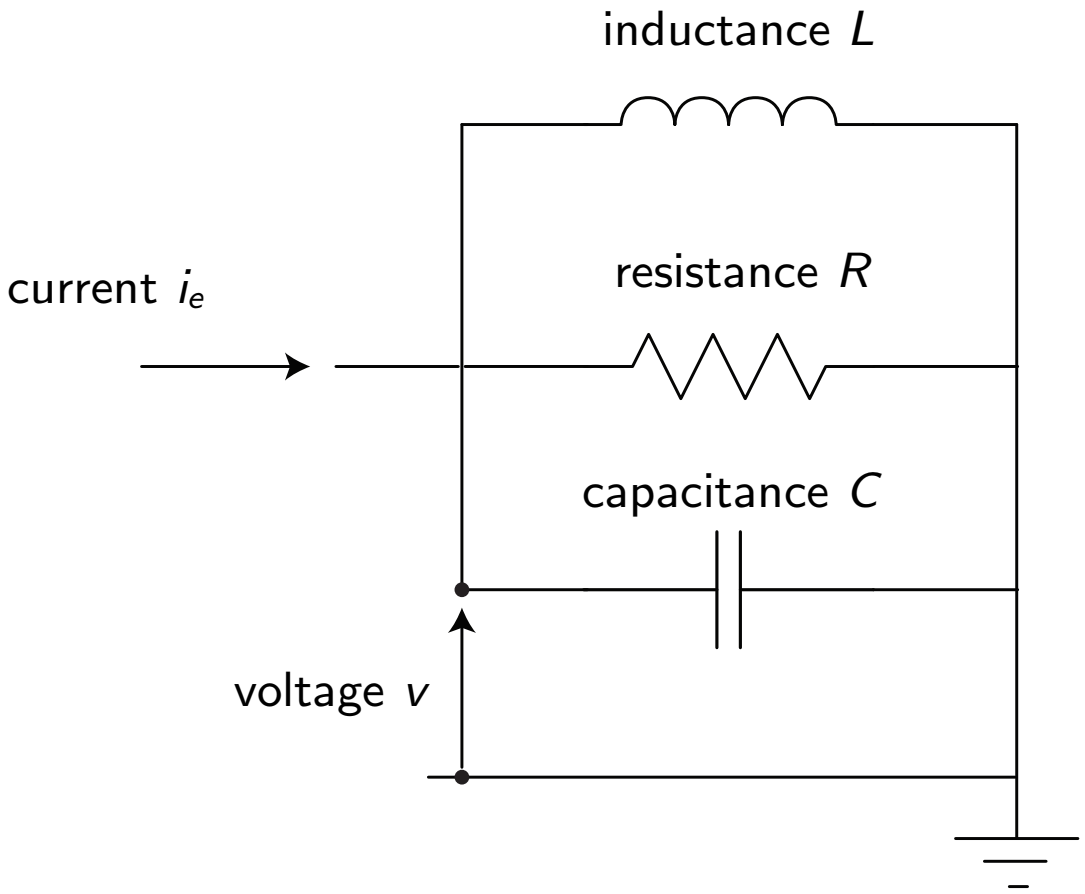
We also introduce the input $u(t) = F_e(t)$. The final set of differential equations is

$$\dot{x}_1 = x_2,$$

$$\dot{x}_2 = \frac{1}{m}(-c\,(x_1 - s_0) - d\,x_2 - u),$$

which has the form $\dot{x}(t) = f(x(t), u(t))$.
The output is the position of the mass: $y = x_1$.

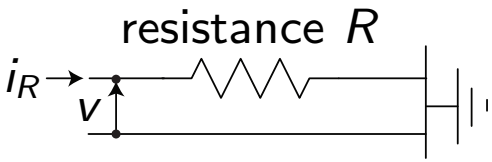
Example II: RLC Circuit



Equations of each Component (RLC Circuit)

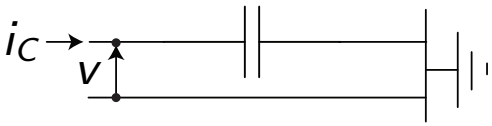
Variables: i (current), v (voltage), i_e (external current);
Parameters: R (resistance), C (capacitance), L (inductance).

Resistance



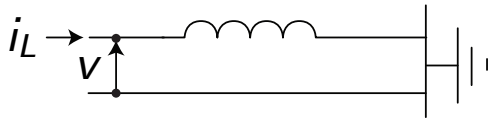
$$i_R = \frac{1}{R} v.$$

Conductance



$$i_C = C\,\dot{v}.$$

Inductance



$$i_L = \frac{1}{L}\int_{t_0}^{t} v(\tau)\,d\tau + i_L(t_0).$$

Sum of currents equals 0: $C\,\dot{v} + \frac{1}{R}v + \frac{1}{L}\int_{t_0}^{t} v(\tau)\,d\tau + i_L(t_0) + i_e = 0.$

Differentiation with respect to time yields: $C\,\ddot{v} + \frac{1}{R}\dot{v} + \frac{1}{L}v + \frac{d}{dt}i_e = 0$

Transformation to State Space Form (RLC Circuit)

The differential equation

$$C\,\ddot{v} + \frac{1}{R}\dot{v} + \frac{1}{L}v + \frac{d}{dt}i_e = 0$$

is of second order. Analogously to the MSD system, we introduce state variables:

$$x_1 := v$$
$$x_2 := \dot{v}$$

We also introduce the input $u(t) = \frac{d}{dt}i_e(t)$. The final set of differential equations is

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = \frac{1}{C}(-\frac{1}{L}\,x_1 - \frac{1}{R}\,x_2 - u),$$

which has the form $\dot{x}(t) = f(x(t), u(t))$.
The output is the voltage: $y = x_1$.

Comparison of Example I and II

The mechanical and the electrical system have the same structure.
Mechanical:

$$\begin{aligned}\dot{x}_1 &= x_2,\\ \dot{x}_2 &= \frac{1}{m}(-c\,(x_1 - s_0) - d\,x_2 - u),\\ y &= x_1,\end{aligned}$$

Electrical:

$$\begin{aligned}\dot{x}_1 &= x_2,\\ \dot{x}_2 &= \frac{1}{C}(-\frac{1}{L}\,x_1 - \frac{1}{R}\,x_2 - u),\\ y &= x_1,\end{aligned}$$

Their behavior is identical for $m = C$, $c = \frac{1}{L}$, $d = \frac{1}{R}$, and $s_0 = 0$. From now on, we will only use the abstract state space form, which represents all kinds of systems: mechanical, electrical, thermal, chemical, etc.

Analytical Solutions

For a very limited class of ODEs, analytical solutions exist (i.e. solutions that can be expressed analytically in terms of a finite number of certain "well-known" functions). There exist formularies for known solutions. We will only look at the technique *separation of variables*. Given is the ODE

$$\frac{d\upsilon}{d\xi} = \tilde{g}(\xi)\tilde{f}(\upsilon).$$

Rearranging yields

$$\frac{1}{\tilde{f}(\upsilon)}\frac{d\upsilon}{d\xi} = \tilde{g}(\xi),$$

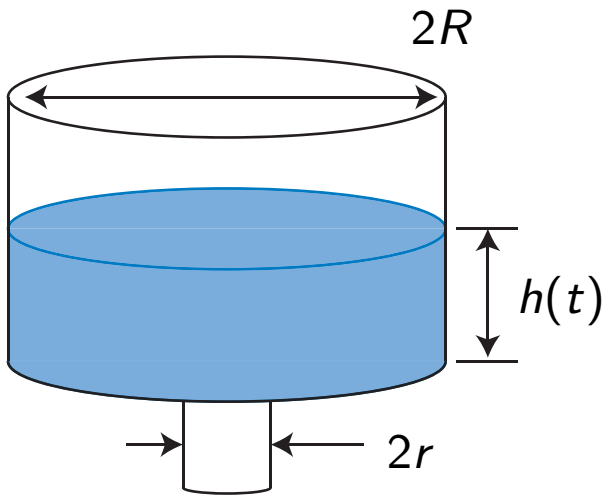
and integrating both sides with respect to ξ results in

$$\int \frac{1}{\tilde{f}(\upsilon)}\frac{d\upsilon}{d\xi}\, d\xi = \int \tilde{g}(\xi)\, d\xi,$$

or equivalently,

$$\int \frac{1}{\tilde{f}(\upsilon)}\, d\upsilon = \int \tilde{g}(\xi)\, d\xi.$$

Example: Solution for Torricelli's Law (I)



The separation of variables makes it possible to solve the differential equation for a liquid that flows out of a tank according to Toricelli's law:

$$\dot{h} = -k\sqrt{h}, \quad k = \frac{r}{R}\sqrt{2g},$$

where h is the height of the fluid in the tank, r and R are the radii of the hole and the tank, and g is the gravity constant, see figure above. With $\upsilon = h$, $\xi = t$, we obtain

$$\frac{d\upsilon}{d\xi} = \tilde{g}(\xi)\tilde{f}(\upsilon), \quad \tilde{g}(\xi) = 1, \quad \tilde{f}(\upsilon) = -k\sqrt{\upsilon} = -k\upsilon^{0.5}.$$

Example: Solution for Torricelli's Law (II)

Integration yields

$$\int \frac{1}{\tilde{f}(v)}\, dv = \int -\frac{1}{k} v^{-0.5}\, dv = -\frac{2}{k} v^{0.5} + C_1$$

$$\int \tilde{g}(\xi)\, d\xi = \int 1\, d\xi = \xi + C_2$$

Since $\int \frac{1}{\tilde{f}(v)}\, dv = \int \tilde{g}(\xi)\, d\xi$, one obtains

$$-\frac{2}{k} v^{0.5} = \xi + C$$

$$v^{0.5} = -\frac{k}{2}(\xi + C)$$

$$v = \Big(-\frac{k}{2}(\xi + C) \Big)^2.$$

The constant C is obtained from the initial state $v(0) = h_0$:

$$v(0) = (-\frac{k}{2} C)^2 = h_0 \rightarrow C = \pm \frac{2}{k} \sqrt{h_0}.$$

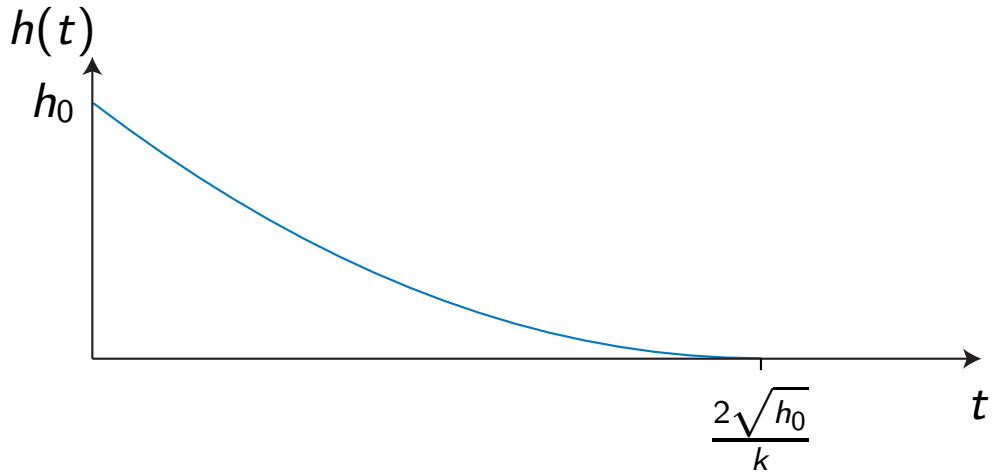
Example: Solution for Torricelli's Law (III)

Inserting $v = h$ and $\xi = t$ yields the solution

$$h(t) = \left(\sqrt{h_0} - \frac{k}{2}t\right)^2.$$

Since the differential equation is no longer defined for $h(t) = 0$ (tank is empty), we obtain the final result

$$h(t) = \begin{cases} \left(\sqrt{h_0} - \frac{k}{2}t\right)^2 & \text{for } 0 \leq t \leq \frac{2\sqrt{h_0}}{k}, \\ 0 \quad \text{for } \frac{2\sqrt{h_0}}{k} < t. \end{cases}$$



Linear Time-Invariant Ordinary Differential Equations

An important class are linear time invariant (LTI) ordinary differential equations since

- many technical systems can be described by LTI systems,
- LTI systems have an analytical solution,
- the solution for the initial state and the input can be obtained separately (superposition principle, see later).

A one-dimensional LTI system is

$$\dot{x}(t) = a\,x(t) + u(t),$$

where x is the state and u is the input. We first consider the homogeneous solution ($u(t) = 0$):

$$\dot{x}(t) = a\,x(t),$$

for which *separation of variables* can be applied using $v = x$, $\xi = t$.

Homogeneous Solution of Scalar LTI Systems

Integration yields

$$\int \frac{1}{\tilde{f}(v)}\, dv = \int \frac{1}{av}\, dv = \frac{1}{a} \ln |v| + C_1; \qquad \int \tilde{g}(\xi)\, d\xi = \int 1\, d\xi = \xi + C_2$$

Since $\int \frac{1}{\tilde{f}(v)}\, dv = \int \tilde{g}(\xi)\, d\xi$, one obtains

$$\begin{aligned} \frac{1}{a} \ln |v| &= \xi + C \\ |v| &= e^{a(\xi + C)} \\ v &= C^* e^{(a\,\xi)} \end{aligned}$$

The constant C^* is obtained from the initial state $v(t_0) = x_0$:

$$v(t_0) = C^* e^{(a\,t_0)} = x_0 \quad \rightarrow C^* = e^{(-a\,t_0)} x_0.$$

Homogeneous solution

$$x^{(h)}(t) = e^{a\,(t - t_0)}\, x_0$$

Particular Solution of Scalar LTI Systems

We assume that variation of parameters (German: 'Variation der Konstanten') yields the particular solution $x^{(p)}(t)$ for $x_0 = 0$, $u(t) \neq 0$:

$$x^{(p)}(t) = c(t)x^{(h)}(t).$$

Re-arranging the LTI system and inserting $x^{(p)}(t)$ results in

$$\begin{aligned} u(t) &= \dot{x}^{(p)}(t) - a\,x^{(p)}(t) \\ &= \dot{c}(t)x^{(h)}(t) + \underbrace{c(t)\dot{x}^{(h)}(t) - c(t)a\,x^{(h)}(t)}_{=0 \text{ since } \dot{x}^{(h)}(t)=a\,x^{(h)}(t)} \end{aligned}$$

from which follows that

$$c(t) = \int_{t_0}^{t} \frac{1}{x^{(h)}(\tau)} u(\tau)\,d\tau + C = \int_{t_0}^{t} e^{-a\,(\tau - t_0)} u(\tau)\,d\tau + C^*$$

Particular solution ($x_0 = 0 \rightarrow C^* = 0$)

$$x^{(p)}(t) = c(t)x^{(h)}(t) = \int_{t_0}^{t} e^{a\,(t-\tau)} u(\tau)\,d\tau$$

Superposition Principle of LTI Systems

Superposition principle

If $x^{(1)}(t)$ and $x^{(2)}(t)$ are solutions of

$$\dot{x}(t) = a\,x(t) + u(t)$$

for $x_0^{(1)}$, $x_0^{(2)}$ and $u^{(1)}(t)$, $u^{(2)}(t)$, respectively, then

$$x(t) = \alpha x^{(1)}(t) + \beta x^{(2)}(t)$$

is a solution for $x_0 = \alpha x_0^{(1)} + \beta x_0^{(2)}$ and $u(t) = \alpha u^{(1)}(t) + \beta u^{(2)}(t)$.

Proof:

$$\begin{aligned}
x(t) &= \alpha x^{(1)}(t) + \beta x^{(2)}(t) \\
\rightarrow \dot{x}(t) &= \alpha \dot{x}^{(1)}(t) + \beta \dot{x}^{(2)}(t) \\
&= \alpha\left(a\,x^{(1)}(t) + u^{(1)}(t)\right) + \beta\left(a\,x^{(2)}(t) + u^{(2)}(t)\right) \\
&= a\left(\alpha x^{(1)}(t) + \beta x^{(2)}(t)\right) + \alpha u^{(1)}(t) + \beta u^{(2)}(t) \\
&= a x(t) + u(t).
\end{aligned}$$

Solution of Scalar LTI Systems

Due to the superposition principle, the final solution is a combination of the homogeneous solution and the particulate solution:

Solution for one-dimensional LTI systems

$$x(t) = x^{(h)}(t) + x^{(p)}(t) = e^{a\,(t-t_0)}\,x_0 + \int_{t_0}^{t} e^{a\,(t-\tau)}\,u(\tau)\,d\tau$$

Example:

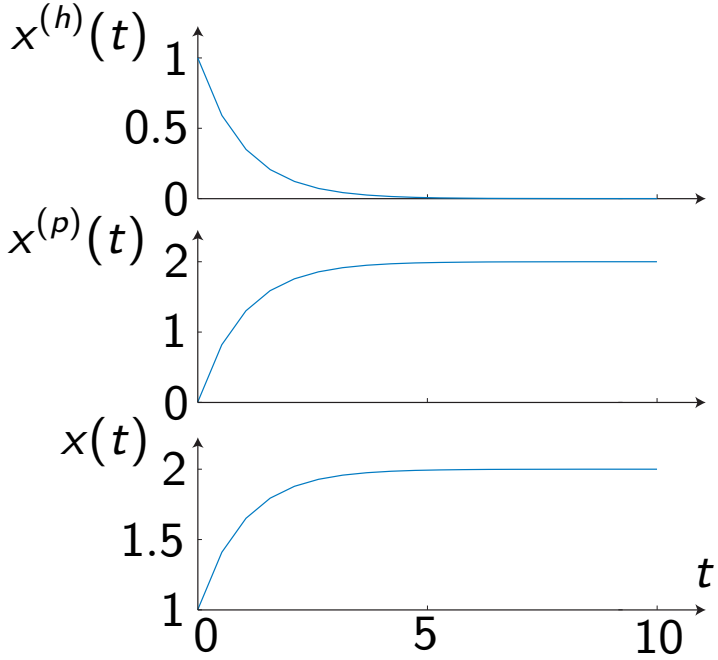
$$\dot{x}(t) = -x(t) + u(t), \quad x_0 = 1, \quad u(t) = 2 \text{ (const)}$$

The solution is

$$x^{(h)}(t) = e^{-t}$$

$$x^{(p)}(t) = 2\int_{t_0}^{t} e^{-(t-\tau)}\,d\tau = 2(1 - e^{-(t-t_0)})$$

The particular solution always becomes dominant for stable systems. Why?



Linear Time-Invariant Systems (n-dimensional)

We are now considering n-dimensional instead of one-dimensional LTI systems:

$$\dot{x}(t) = A\,x(t) + u(t),$$

where $x \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^n$ is the input, and $A \in \mathbb{R}^{n \times n}$ is the system matrix. The solution is analogous to one-dimensional LTI systems:

Solution for n-dimensional LTI systems

$$x(t) = x^{(h)}(t) + x^{(p)}(t) = e^{A\,(t-t_0)}\,x_0 + \int_{t_0}^{t} e^{A\,(t-\tau)}\,u(\tau)\,d\tau$$

The exponential matrix is defined as the power series

$$e^X = \sum_{k=0}^{\infty} \frac{1}{k!} X^k = I + X + \frac{1}{2!}X^2 + \frac{1}{3!}X^3 + \ldots$$

where I is the identity matrix. From this follows that

$$\frac{d}{dt}e^{At} = \frac{d}{dt}I + \frac{d}{dt}At + \frac{d}{dt}\frac{1}{2!}(At)^2 + \ldots = A + A\frac{1}{1!}(At) + A\frac{1}{2!}(At)^2 + \ldots = Ae^{At}$$

Proof of Solution for Linear Time-Invariant Systems

To prove correctness, we require the Leibniz integral rule:

$$\frac{d}{dx}\int_{u(x)}^{v(x)} F(x,\phi)\,d\phi = F(x,v)\frac{dv}{dx} - F(x,u)\frac{du}{dx} + \int_{u(x)}^{v(x)} \frac{\partial}{\partial x}F(x,\phi)\,d\phi$$

Differentiation of $x(t)$ yields

$$\begin{aligned}
\dot{x}(t) =& \frac{d}{dt}e^{A(t-t_0)}x_0 + \underbrace{\frac{d}{dt}\int_{t_0}^{t} e^{A(t-\tau)}u(\tau)\,d\tau}_{Leibniz} \\
=& Ae^{A(t-t_0)}x_0 + u(t) + \int_{t_0}^{t} Ae^{A(t-\tau)}u(\tau)\,d\tau \\
=& A\left(e^{A(t-t_0)}x_0 + \int_{t_0}^{t} e^{A(t-\tau)}u(\tau)\,d\tau\right) + u(t) \\
=& Ax(t) + u(t) \quad \checkmark
\end{aligned}$$

Example: Mass Spring Damper (MSD) System

Reminder:

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = \frac{1}{m}(-c\,(x_1 - s_0) - d\,x_2 - \hat{u}).$$

After introducing the state matrix A and the input vector u as

$$A = \begin{bmatrix} 0 & 1 \\ -\frac{c}{m} & -\frac{d}{m} \end{bmatrix}, \qquad u = \begin{bmatrix} 0 \\ \frac{c}{m} s_0 - \frac{1}{m}\hat{u} \end{bmatrix}$$

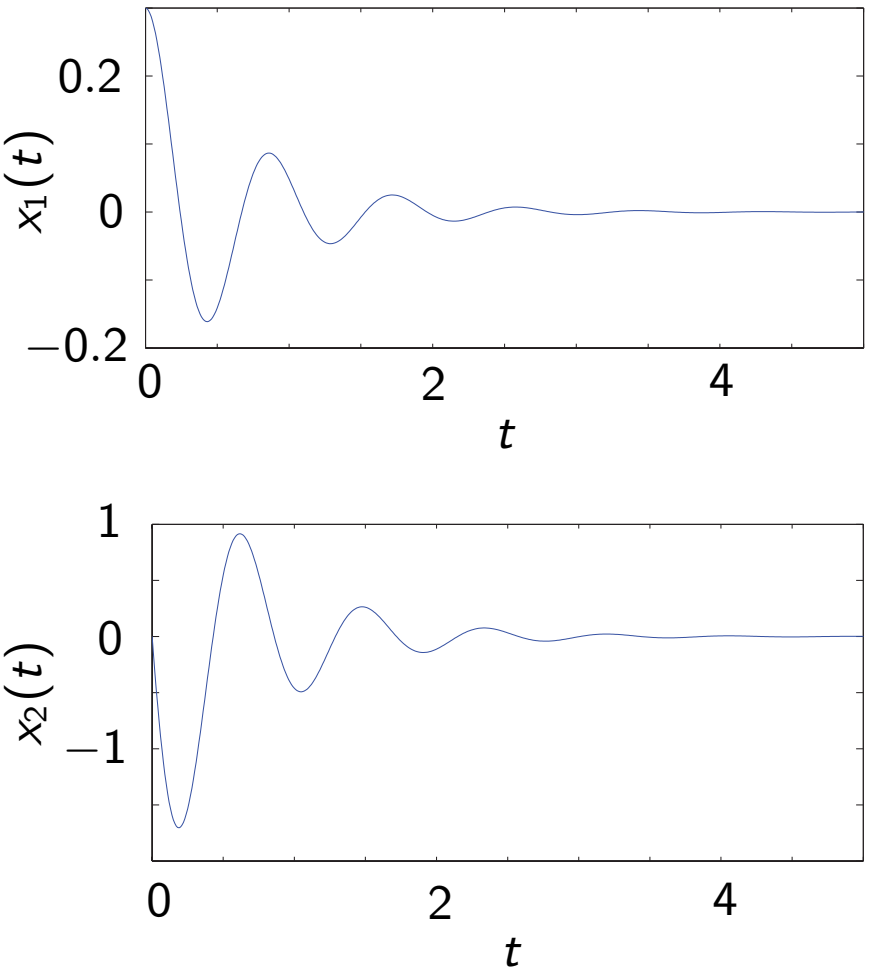
The system dynamics can be written as

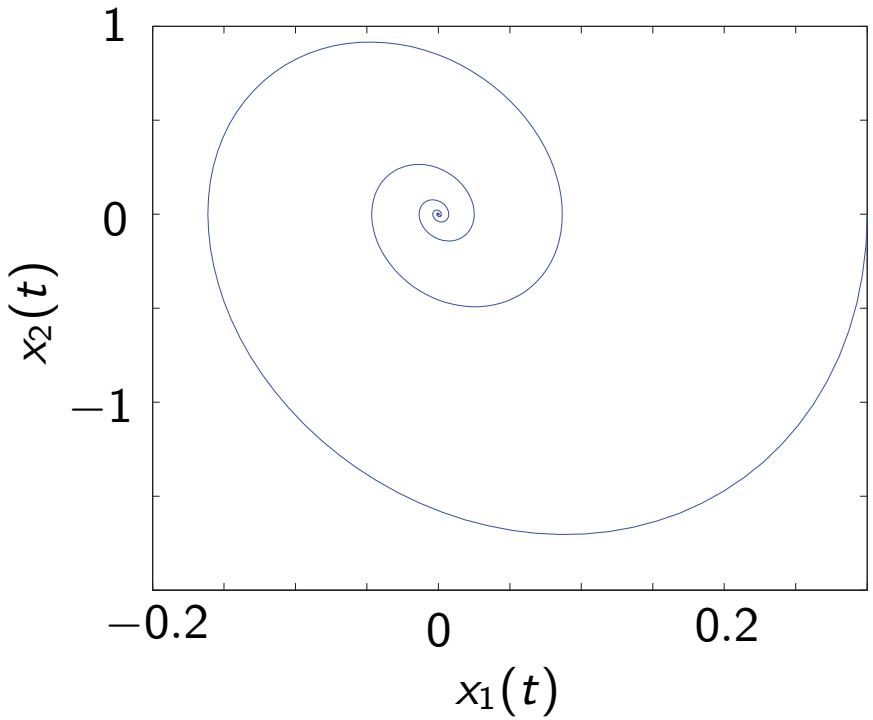
$$\dot{x}(t) = A\,x(t) + u(t).$$

We use parameters from a quarter car model:

m [kg]	c [N/m]	d [Ns/m]	s_0 [m]
450	25000	1300	0

Exact Solution of MSD System





Numerical Solutions

Analytical solutions only exist for specific nonlinear ODEs and linear ODEs. All other systems have to be solved using numerical techniques. We will start with simple techniques and move towards more sophisticated techniques used in commercial tools (such as MATLAB/Simulink):

- Euler method
- Heun method
- Runge-Kutta method
- Runge-Kutta-Fehlberg method

For simplicity, we will not consider inputs from now on. Inputs, however, can be easily integrated in each method.

Euler Method

The Euler method is the simplest algorithm for approximating solutions of ODEs. The solution of the ODE $\dot{x}(t) = f(x(t))$ for a future point in time is

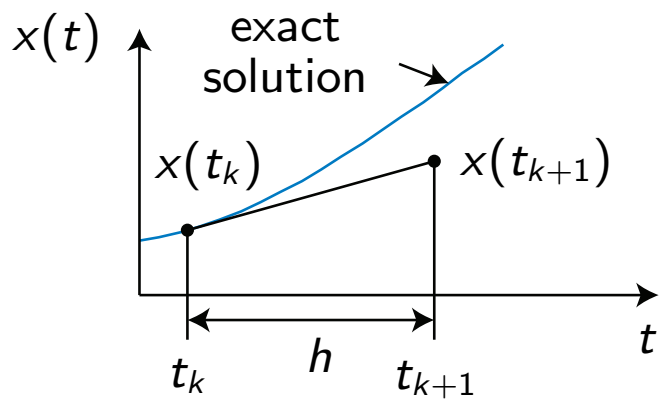
$$x(t_{k+1}) = x(t_k) + \int_{t_k}^{t_{k+1}} f(x(\tau))d\tau.$$

Since the integral cannot be exactly obtained in general, one can use:

Euler method

$$x(t_{k+1}) \approx x(t_k) + \underbrace{(t_{k+1} - t_k)}_{=:h} f(x(t_k)).$$

The Euler method is an *explicit method* since it only uses values from the past.



Heun Method

Heun's method does not only use the slope at the previous time step, but the average slope from the previous and the next time step:

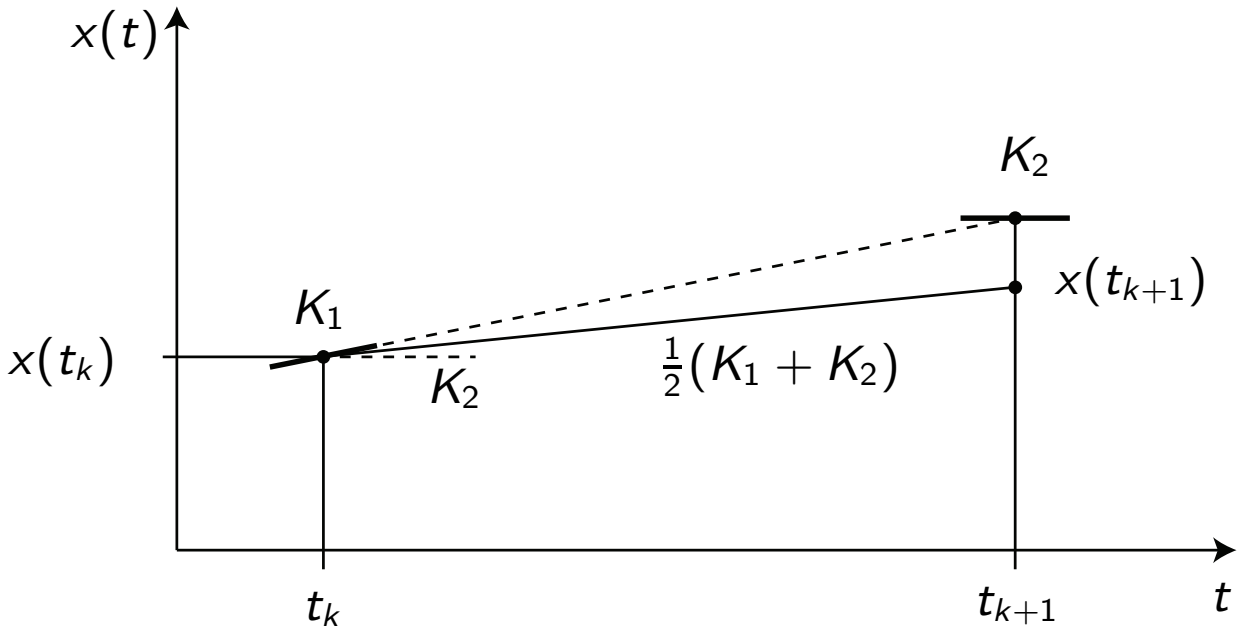
$$x(t_{k+1}) \approx x(t_k) + \underbrace{(t_{k+1} - t_k)}_{=:h} \frac{1}{2} \Big(f\big(x(t_k)\big) + f\big(x(t_{k+1})\big) \Big)$$

The above scheme is also referred to as the trapezoidal rule, which is an implicit method since it requires future values. Solving for $x(t_{k+1})$ requires solving a set of nonlinear equations, which is typically done using Newton's method. Heun's method approximates $x(t_{k+1})$ by Euler's method:

Heun's method

$$x(t_{k+1}) \approx x(t_k) + h \frac{1}{2}(K_1 + K_2)$$
$$K_1 = f\big(x(t_k)\big)$$
$$K_2 = f\big(x(t_k) + hK_1\big)$$

Illustration of Heun Method



Runge-Kutta Method

The Runge-Kutta method is a very popular method to numerically solve ODEs. To further improve the accuracy, a new supporting point and four function evaluations are used:

Runge-Kutta method

$$x(t_{k+1}) \approx x(t_k) + h\frac{1}{6}(K_1 + 2\,K_2 + 2\,K_3 + K_4)$$

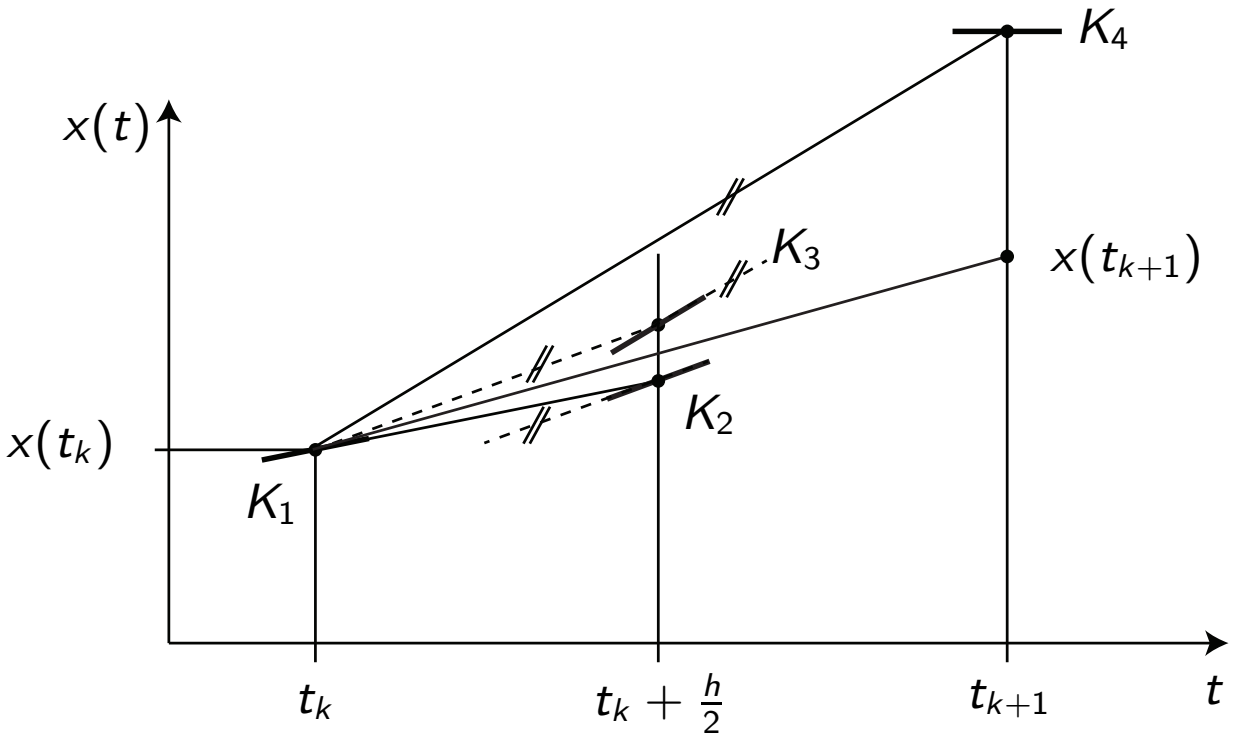
$$K_1 = f\left(x(t_k)\right)$$

$$K_2 = f\left(x(t_k) + \frac{1}{2}hK_1\right)$$

$$K_3 = f\left(x(t_k) + \frac{1}{2}hK_2\right)$$

$$K_4 = f\left(x(t_k) + hK_3\right)$$

Illustration of Runge-Kutta Method



Local Truncation Error

The Heun method requires 2 function evaluations and the Runge-Kutta method even 4 function evaluations. Is the additionally effort justified?

Yes! The accuracy is orders of magnitude better, making it possible to extend the step size (see exercise).

Order of numerical methods

The method has order p if there exist constants h and C such that

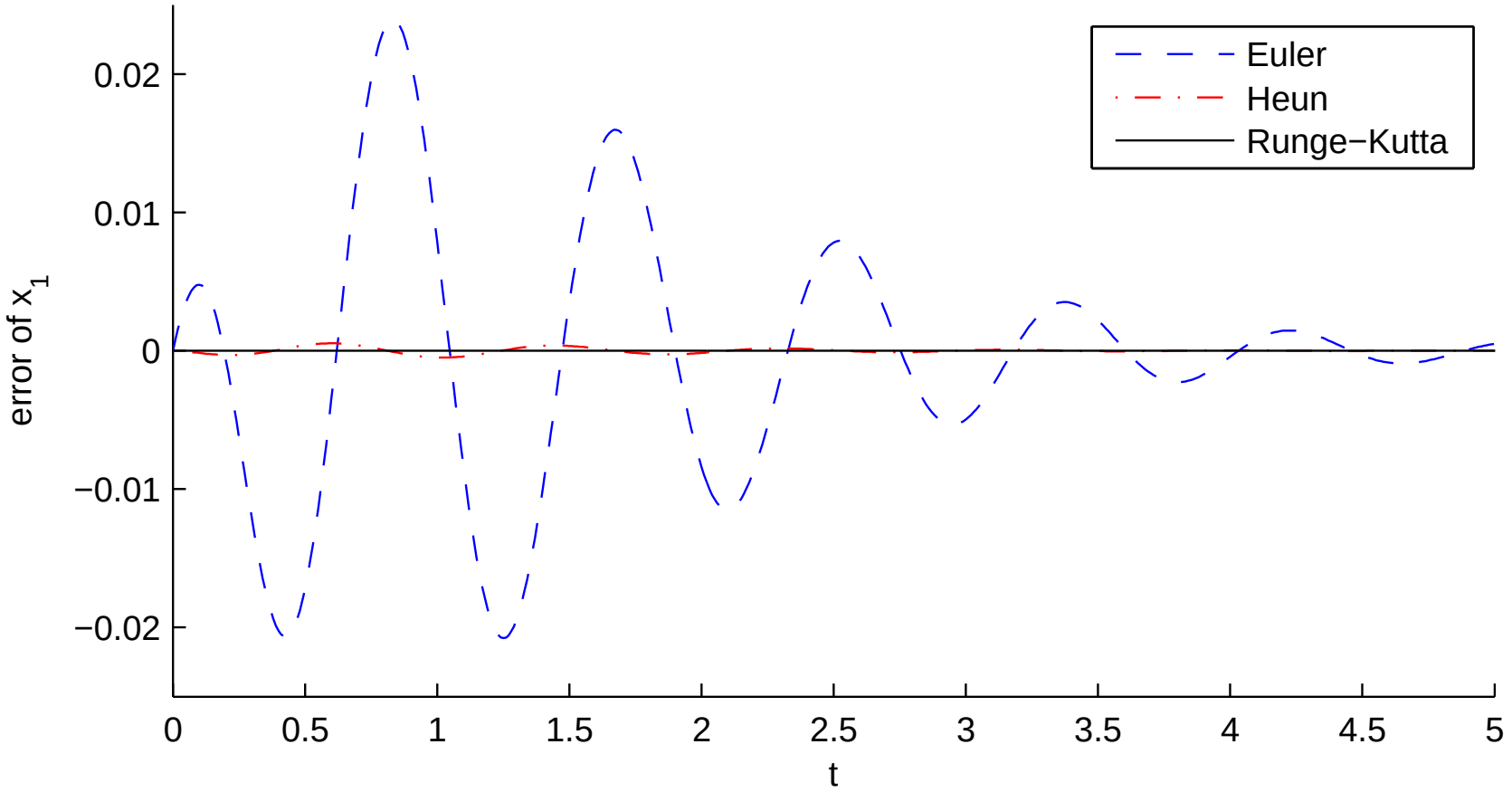
$$\left| \frac{x^e(t_k + h) - x(t_k + h)}{h} \right| \leq C\, h^p$$

One can show that

method:	Euler	Heun	Runge-Kutta
order:	1	2	4

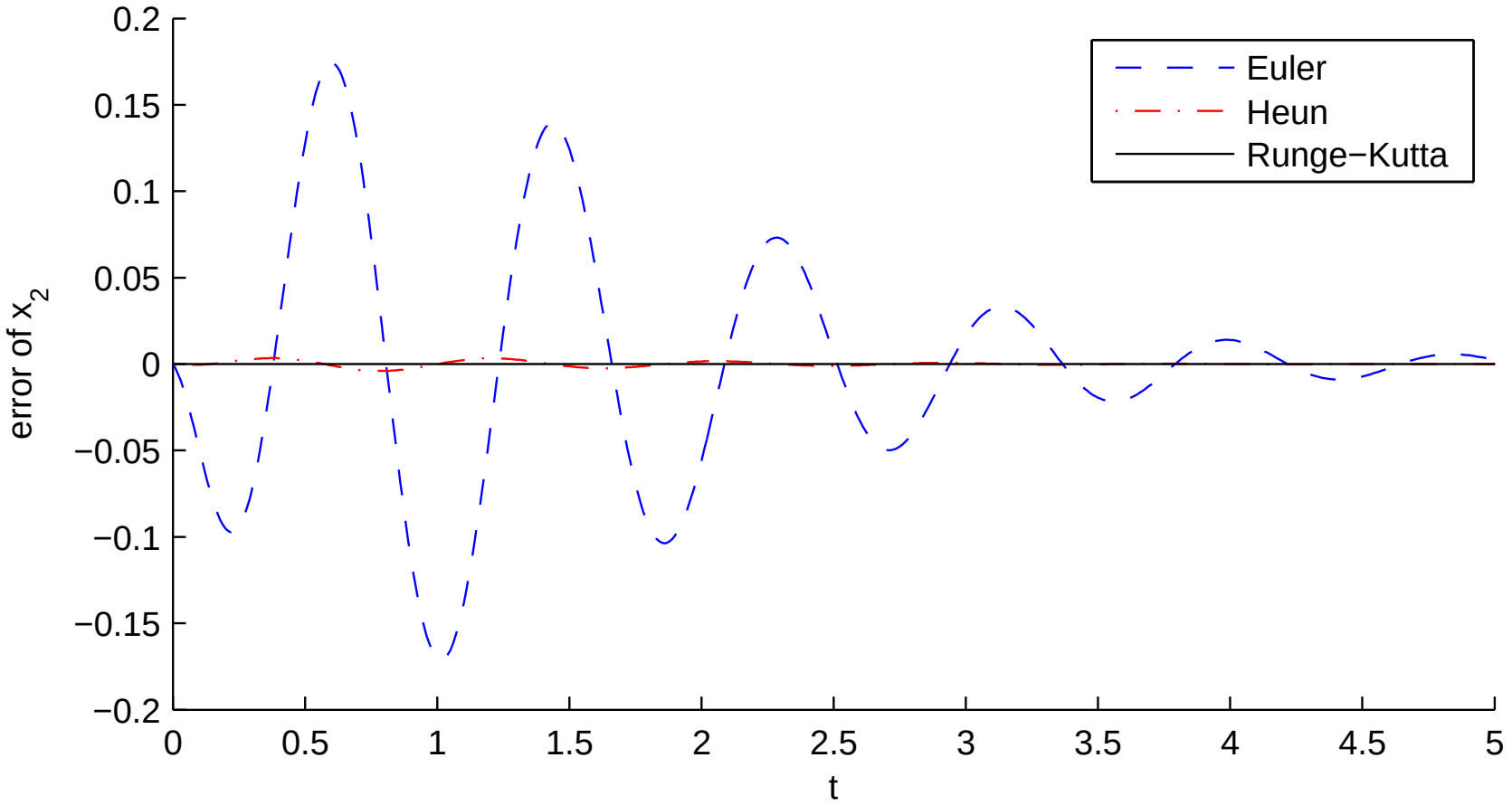
Comparison of Methods for h=0.01

We use the MSD system from previous slides to compute the error for $\mathbf{h = 0.01}$ to the exact solution (exact solution exists for linear systems).



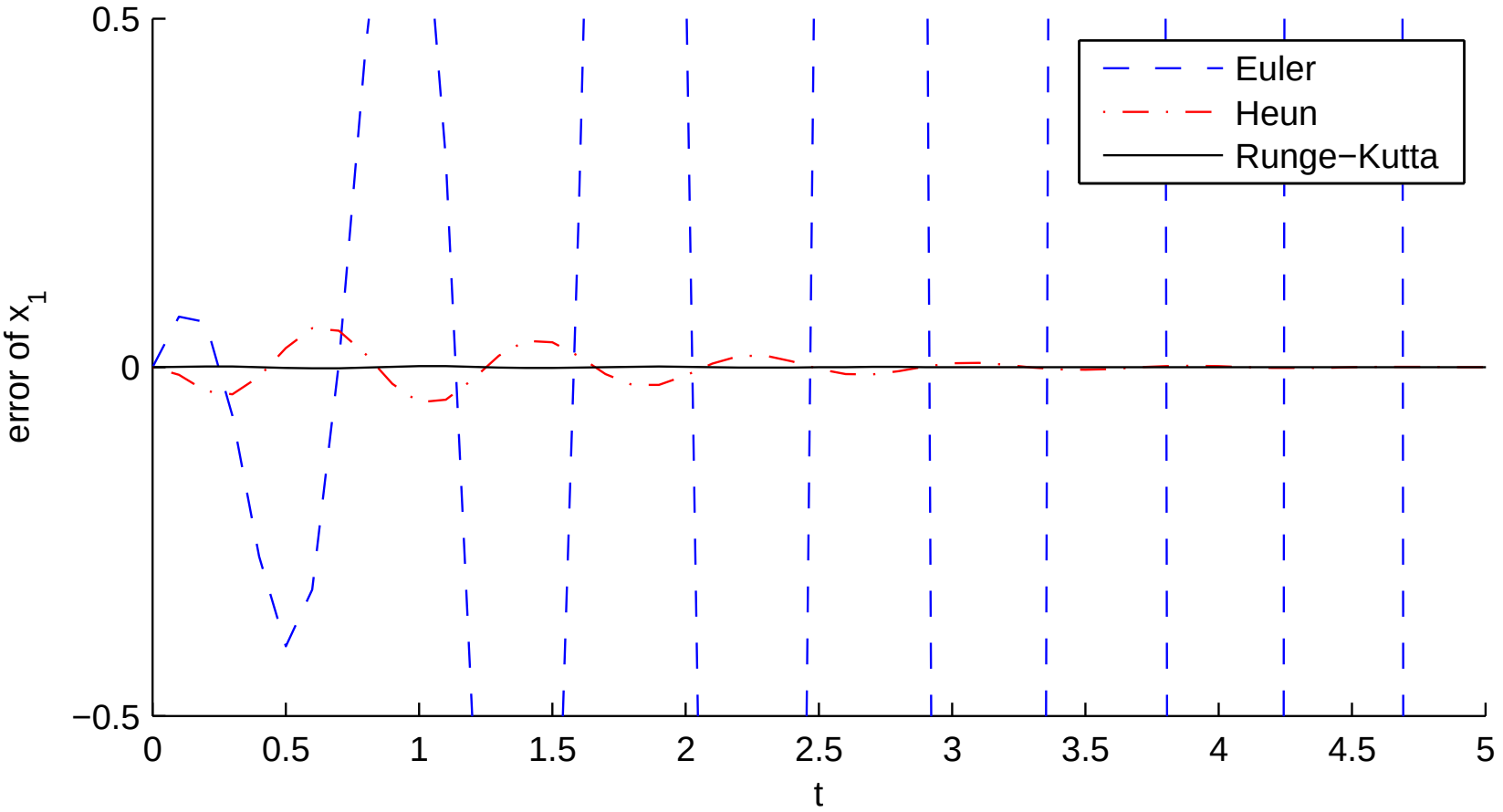
Comparison of Methods for h=0.01

We use the MSD system from previous slides to compute the error for $\mathbf{h = 0.01}$ to the exact solution (exact solution exists for linear systems).



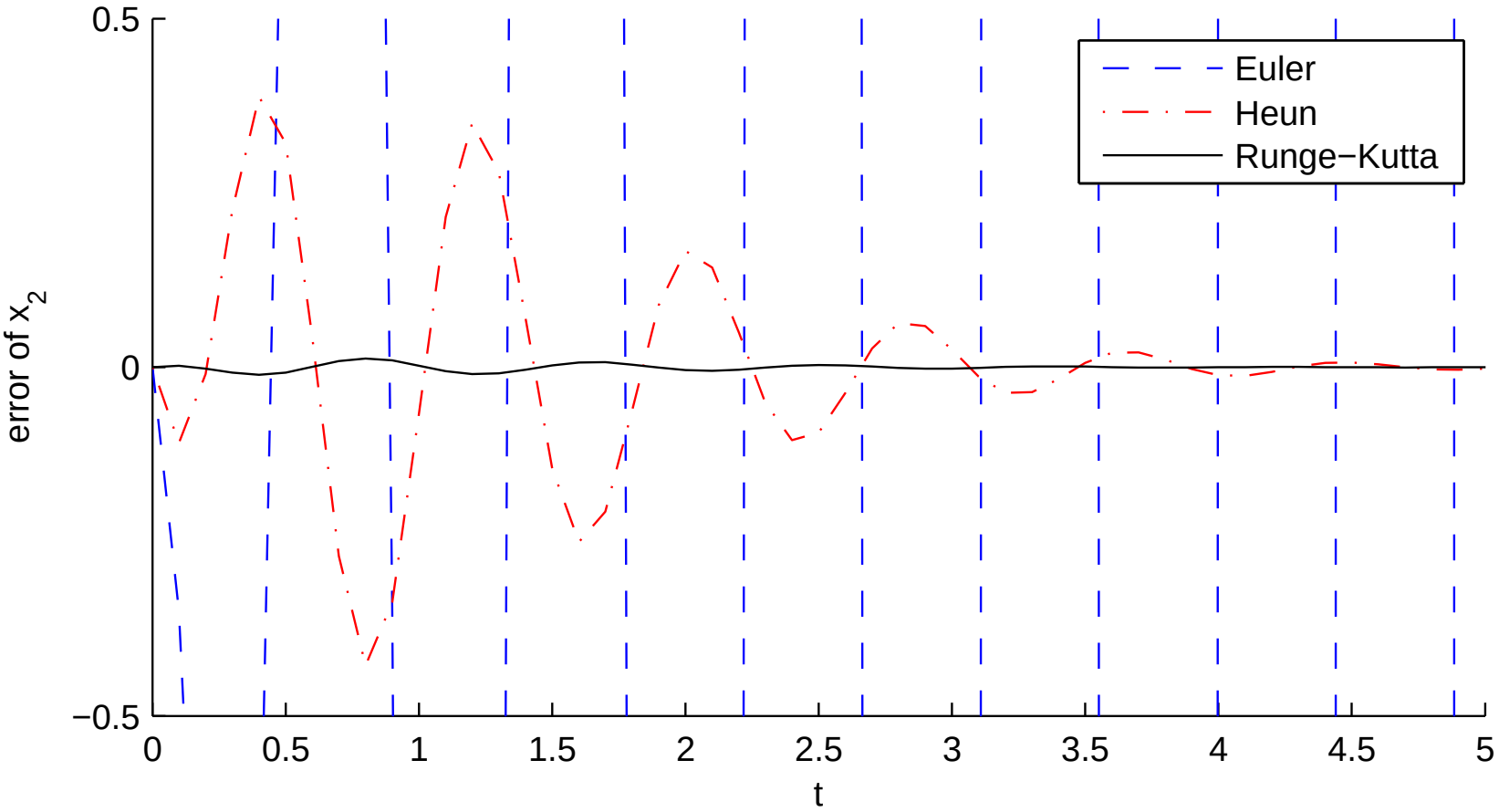
Comparison of Methods for h=0.1

We use the MSD system from previous slides to compute the error for $\mathbf{h = 0.1}$ to the exact solution (exact solution exists for linear systems).



Comparison of Methods for h=0.1

We use the MSD system from previous slides to compute the error for $\mathbf{h = 0.1}$ to the exact solution (exact solution exists for linear systems).



Discretization and Rounding Error

Numerical techniques suffer from two kinds of errors:

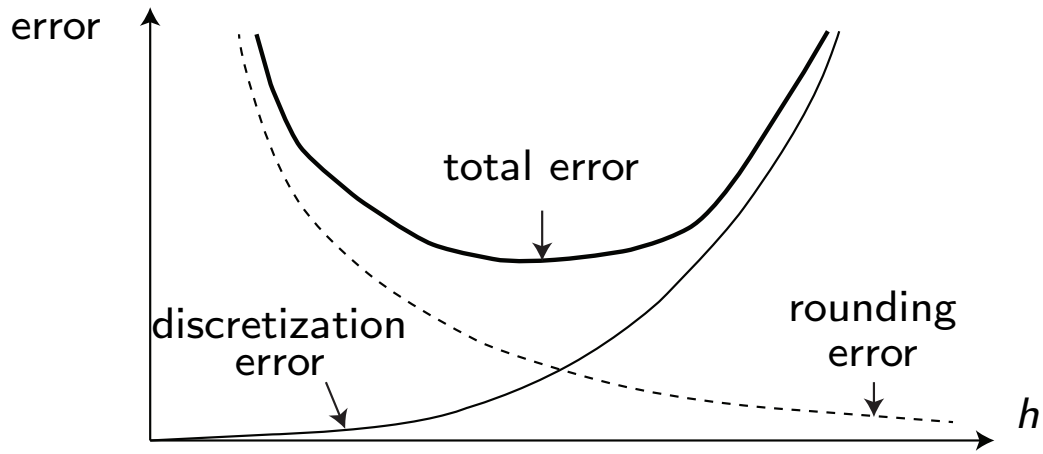
Discretization error

Refers to the error due to time discretization. Smaller step sizes h decrease the discretization error.

Rounding error

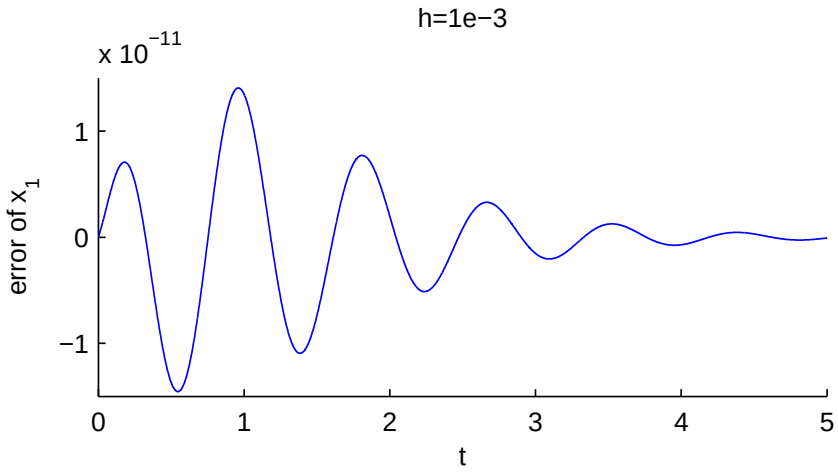
Refers to the error due to the finite number of digits to represent numbers in computers. Larger step sizes h increase the number of time steps and thus the number of computations with rounding errors.

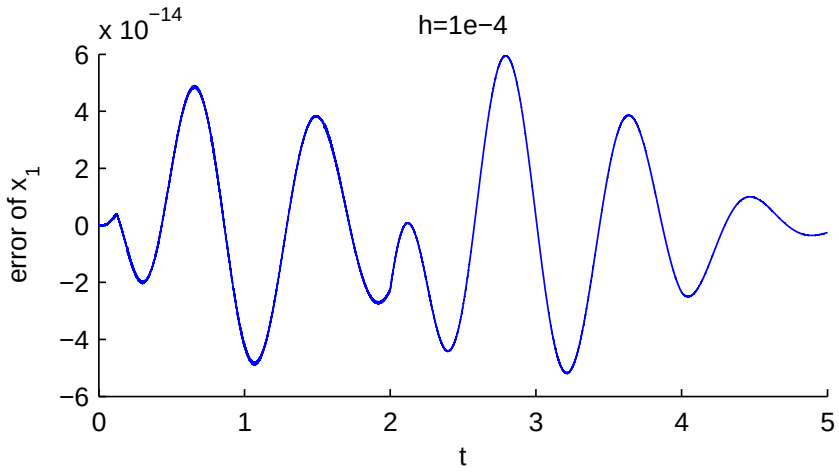
There exists an optimal step size for the total error:

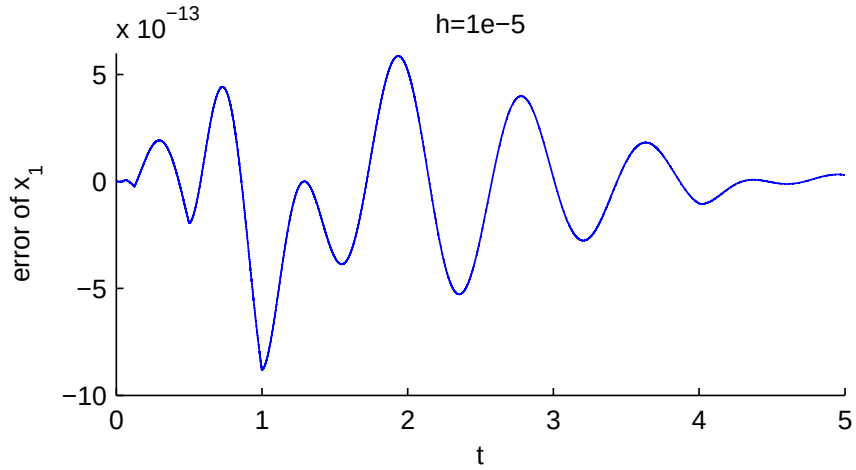


Comparison of Step Sizes

Error using Runge-Kutta for different step sizes:







The best step size is $h = 1e - 4$.

Adaptive Step Size

The discretization error varies depending on the characteristics of the ODE. In some areas, $f(x)$ is sensitive with respect to changes of x, i.e. $\frac{\partial}{\partial x} f(x)$ is large, while it is small elsewhere.

A popoular method that is based on the Runge-Kutta family of methods is the Runge-Kutta-Fehlberg method. It is implemented in MATLAB/Simulink as *ode45*.

Basic idea of the Runge-Kutta-Fehlberg method

1. Compute one-step solution with lower order Runge-Kutta method (order 4).
2. Compute one-step solution with higher order Runge-Kutta method (order 5).
3. If the error is large, the step size is shortened, otherwise the step size is enlarged.

Runge-Kutta-Fehlberg Method

$$K_1 = f\left(x(t_k)\right), \qquad K_2 = f\left(x(t_k) + \frac{1}{4}hK_1\right)$$

$$K_3 = f\left(x(t_k) + \frac{3}{32}hK_1 + \frac{9}{32}hK_2\right)$$

$$K_4 = f\left(x(t_k) + \frac{1932}{2197}hK_1 - \frac{7200}{2197}hK_2 + \frac{7296}{2197}hK_3\right)$$

$$K_5 = f\left(x(t_k) + \frac{439}{216}hK_1 - 8hK_2 + \frac{3680}{513}hK_3 - \frac{845}{4104}hK_4\right)$$

$$K_6 = f\left(x(t_k) - \frac{8}{27}hK_1 + 2hK_2 - \frac{3544}{2565}hK_3 + \frac{1859}{4104}hK_4 - \frac{11}{40}hK_5\right)$$

The solution of the 4^{th} order method is

$$x(t_{k+1}) = x(t_k) + h\left(\frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4101}K_4 - \frac{1}{5}K_5\right) \qquad (K_2 \text{ not used}).$$

The solution of the 5^{th} order method is

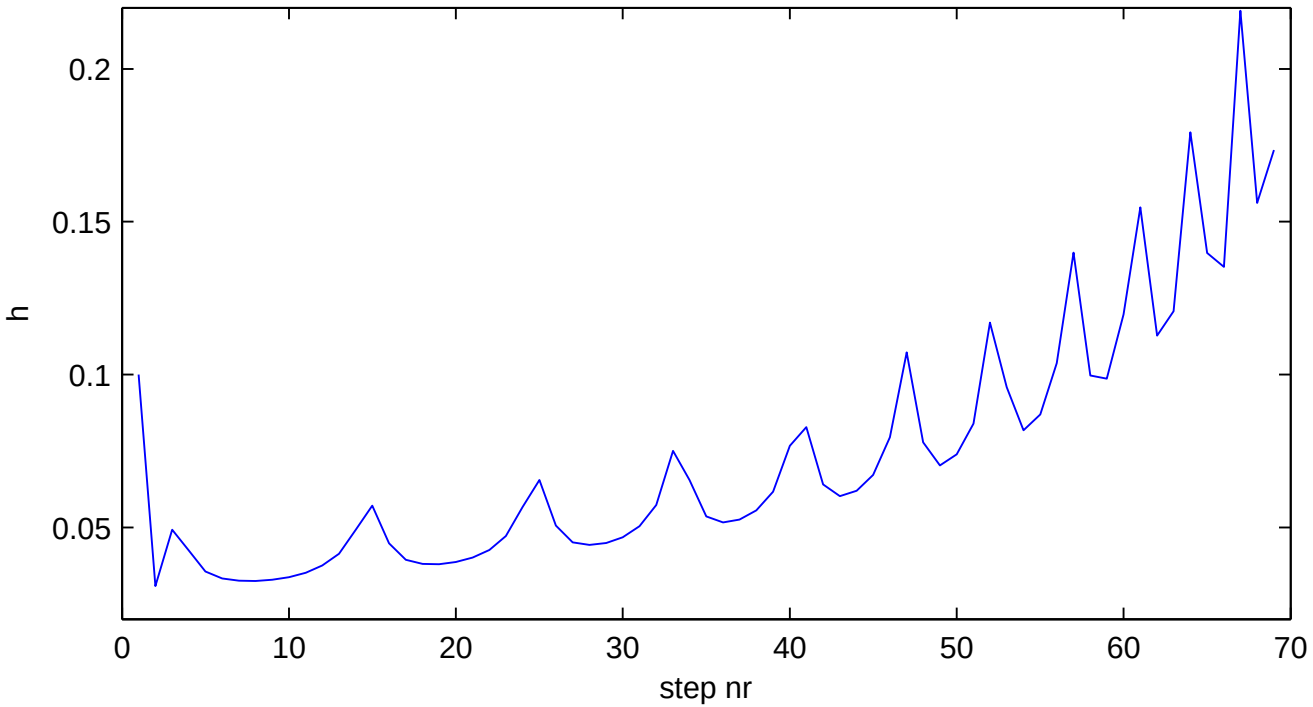
$$\tilde{x}(t_{k+1}) = x(t_k) + h\left(\frac{16}{135}K_1 + \frac{6656}{12825}K_3 + \frac{28561}{56430}K_4 - \frac{9}{50}K_5 + \frac{2}{55}K_6\right) \qquad (K_2 \text{ not used}).$$

The new step size for a specified error tolerance tol is

$$h := \left(\frac{\mathrm{tol}\, h}{2|\tilde{x}(t_{k+1}) - x(t_{k+1})|}\right)^{\frac{1}{4}} h.$$

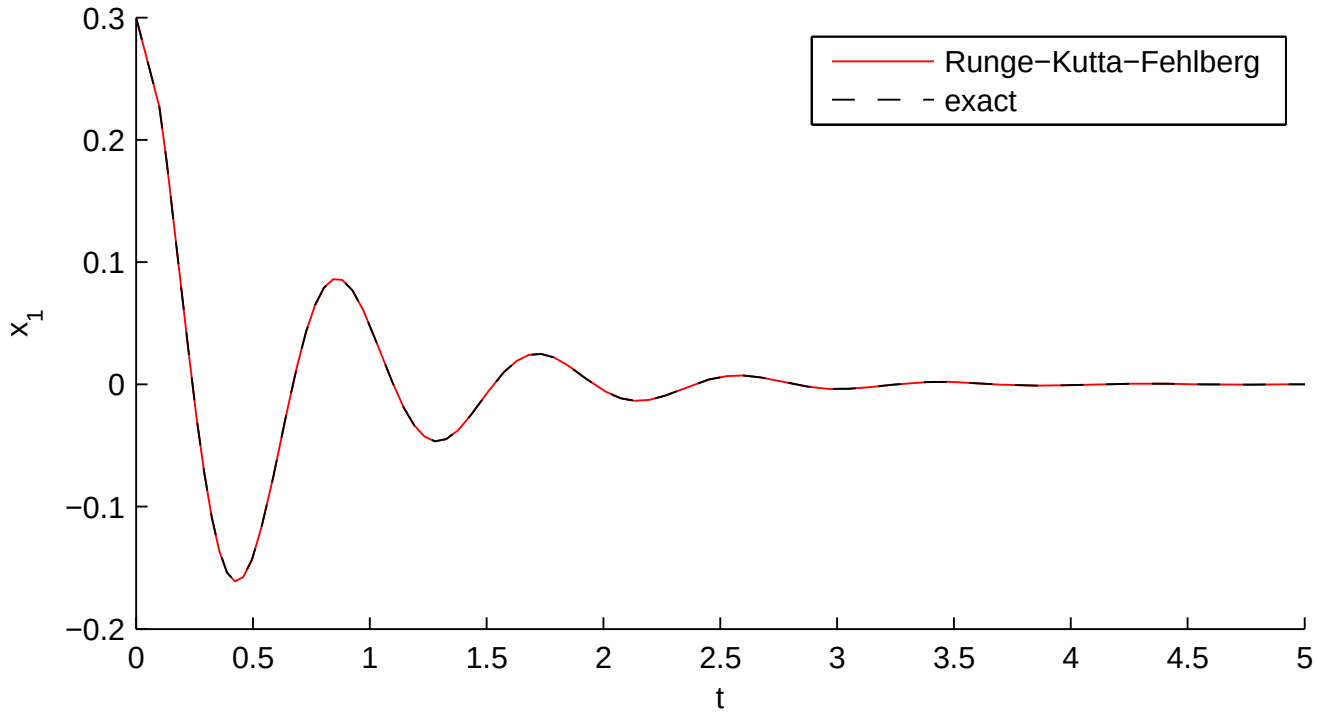
Adaptive Step size for MSD System

We use the MSD system from previous slides to compute the varying step size h for $\mathrm{tol} = 1e-4$ and the initial step size $h = 0.01$.



Adaptive Step size for MSD System

We use the MSD system from previous slides to compute the varying step size h for $\mathrm{tol} = 1e-4$ and the initial step size $h = 0.01$.



The solution is very accurate for this system. However: One should always be skeptical with numerically obtained results.

Conclusions

- The main modeling formalism for dynamical continuous systems are ordinary differential equations (ODEs).
- There exist only a very limited amount of analytical solutions for OSEs. If they exist, one should prefer them over numerical solutions, of course.
- There exist analytical solutions for linear time-invariant systems.
- Numerical solutions of ODEs are best obtained with higher-order methods due to their preferred ratio of accuracy and computation time.
- Adaptive step size typically provides better results than fixed step size.
- One should always be skeptical with numerically obtained results.