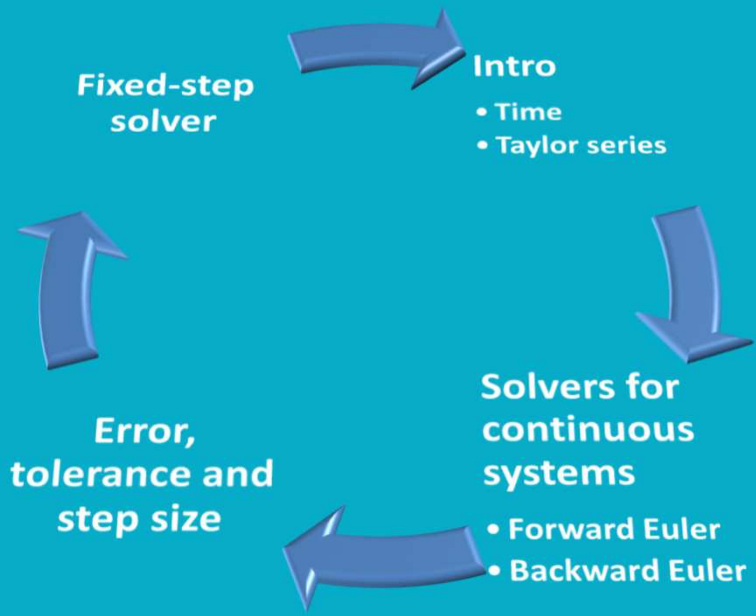
ENGINEERING

FH | JOANNEUM
University of Applied Sciences

ECE Model-Based Design
Numerical Methods for the solution of ODEs



Dipl.-Ing. (FH) Alfred Steinhuber MSc

WS2020/21

Are there any questions???

Numerical Methods

Question

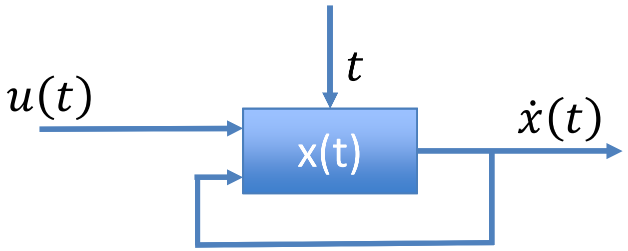
- What is the basis for model development?

Ordinary Differential Equation (ODE)

$$\boxed{\frac{dx}{dt} = x \cdot t + u}$$

$$\boxed{\frac{dx(t)}{dt} = f(x, t)}$$

$$x(t = t_0) = x_0$$



Y = f(x,p,c)

y - dependent variable (observed, calculated data)

x - independent variable (often time)

p - parameters [adjustable] (e.g. kel, V)

c - constants [fixed] (e.g. dose, duration)

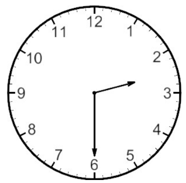
✓ Objective is to find the trajectories of the state variables x over time

✓ The state variables assume infinitely many values in any finite time interval

Numerical Methods

Time

In the real world, time simply happens -we can measure it, but not influence it!



In simulation, time does not simply happen -we need to make it happen!

> *When we simulate a system, it is our duty to manage the simulation clock*
>
> ***Consider:*** *how effectively we are able to manage the simulation clock will ultimately decide upon the efficiency of our simulation run*

Source: [1]

Numerical Methods

Simulation time?

Is not the same as clock time

- E.g.: simulation run of 10 seconds usually does not take 10 seconds

Total simulation time depends on:

- Model complexity
- Solver step sizes and
- Computer speed

How can the trajectories of our previous ODE be gathered?

A solver computes a dynamic system's states at successive time steps over a specified time span, using information provided by the model

Numerical Methods

Possibilities for finding the solution for ODE‘s

a) Find an explicit or implicit solution by calculus
   - ✓ Solving it finding exact solution
   - Difficult to solve, or even impossible to solve!

b) Approximate the solution using slope fields
   - ✓ Graphical methods offering different possibilities

c) Approximate the solution using Eulers Method

Numerical Methods

Numerical methods are classified into two classes

- Single or one step methods
- Multistep methods

Numerical methods yield solution either

a) As power series in t (Taylor's method)
b) As a set of values of t and y (Euler, Runge-Kutta…)
   - ✓ Information at a single point xi is required
   - ✓ Values of y are calculated in short steps for equal intervals of t
   - ✓ Methods are used for a limited range of t values

Further Methods for finding y over a wide range of t values

Milne

Adams-Basforth

Source: [2]

Numerical Methods

Graphical methods

Objective is to calculate

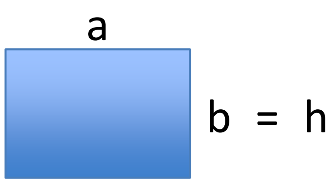
$$\boxed{y = \int_a^b f(x)dx}$$

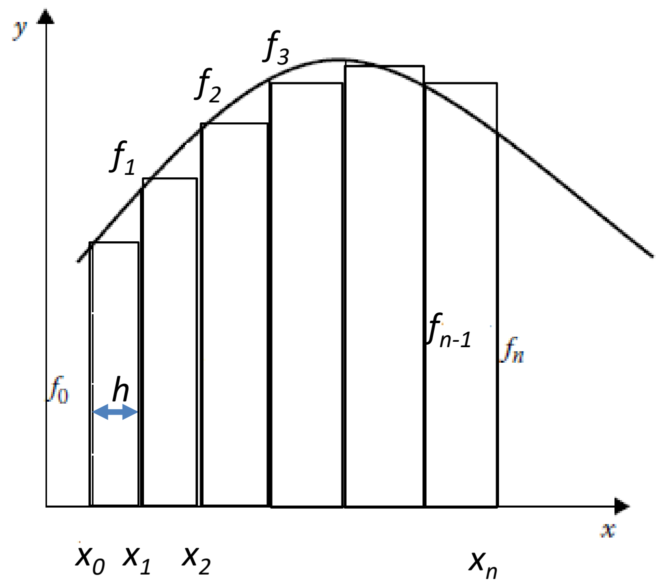
$$a < b$$

Subdivision of integrals $h = \frac{b-a}{n}$ $x_1 = a$ $x_2 = a + h$ $x_n = b$

Rectangle Rule

$A_{rect} = a * b$





Estimation

$$y_{rect} \approx h(f_0 + f_1 + f_2 + \cdots + f_{n-1})$$

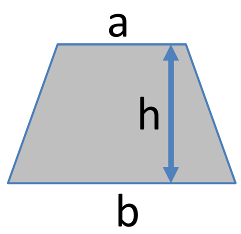
Source: [2]

Numerical Methods

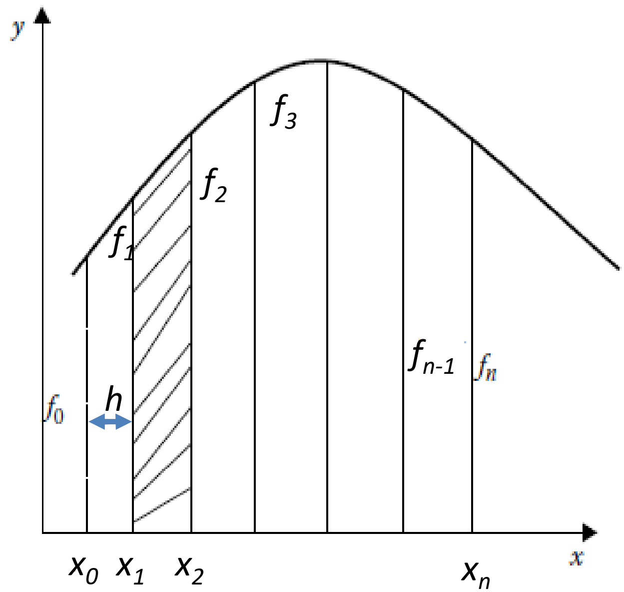
Graphical methods

Trapezoidal Formular

$$A_{trap} = \frac{(a+b)}{2} h$$



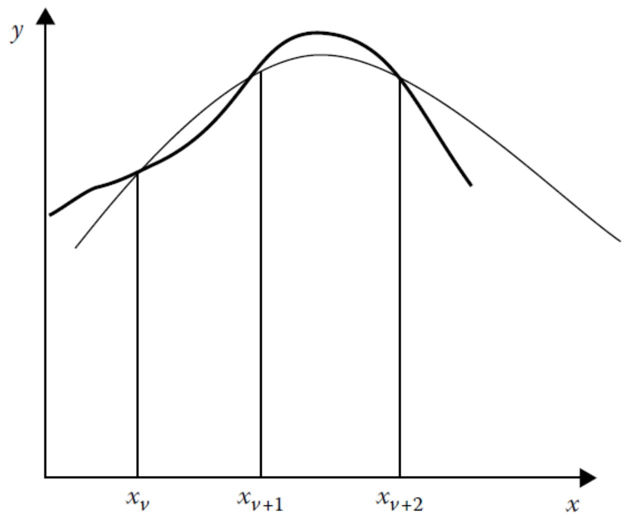
$$y_{trap} \approx \frac{h}{2}(f_0 + 2f_1 + 2f_2 + \cdots + 2f_{n-1} + f_n)$$



Simpson‘s Formular

Finds a parabola through points

$$y_{simp} \approx \frac{h}{3}(f_0 + 4f_1 + f_2)$$



Source: [2]

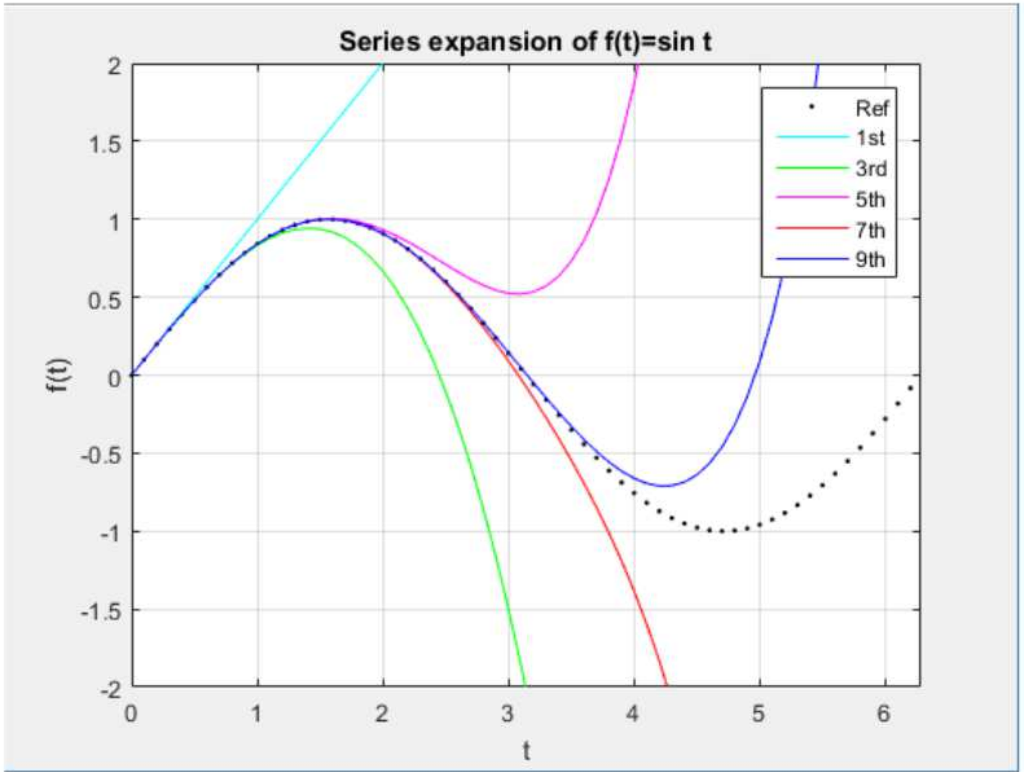
Numerical Methods

Lets look at Taylor Series

„Study of taylor series is about to take none polynomial functions and finding polynomials approximating them near some input"

Advantages:

- ✓ Compute
- ✓ Derivate
- ✓ Integrate



Numerical Methods

Taylor series expansion

Definition:

$$f(t) \approx a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + \cdots + a_n(t - t_0)^n$$

where coefficients a_k represent the polynomial order and are calculated as

$$\boxed{a_k = \frac{f^{(k)}(t_0)}{k!} \; \forall \, 0 \leq k \leq n}$$

Numerical Methods

$E.g.: f(t) = \sin t;$

Let's consider $t_0 = 0$ and calculate a_k for different orders:

$$\boxed{a_k = \frac{f^{(k)}(t_0)}{k!} \; \forall \, 0 \leq k \leq n}$$

$$a_o = \frac{f(0)}{0!} = \frac{\sin 0}{0!} = 0$$

$$a_1 = \frac{f\prime(0)}{0!} = \frac{cos\ 0}{0!} = 1$$

$$a_2 = \frac{f\prime\prime(0)}{2!} = \frac{-\sin 0}{2!} = 0$$

$$a_3 = \frac{f\prime\prime\prime(0)}{3!} = \frac{-cos\ 0}{3!} = \frac{-1}{3!}$$

$$a_4 = \frac{f\prime\prime\prime\prime(0)}{4!} = \frac{sin\ 0}{4!} = \frac{0}{4!}$$

$$a_5 = \frac{f\prime\prime\prime\prime\prime(0)}{5!} = \frac{cos\ 0}{5!} = \frac{0}{5!}$$

By deduction:

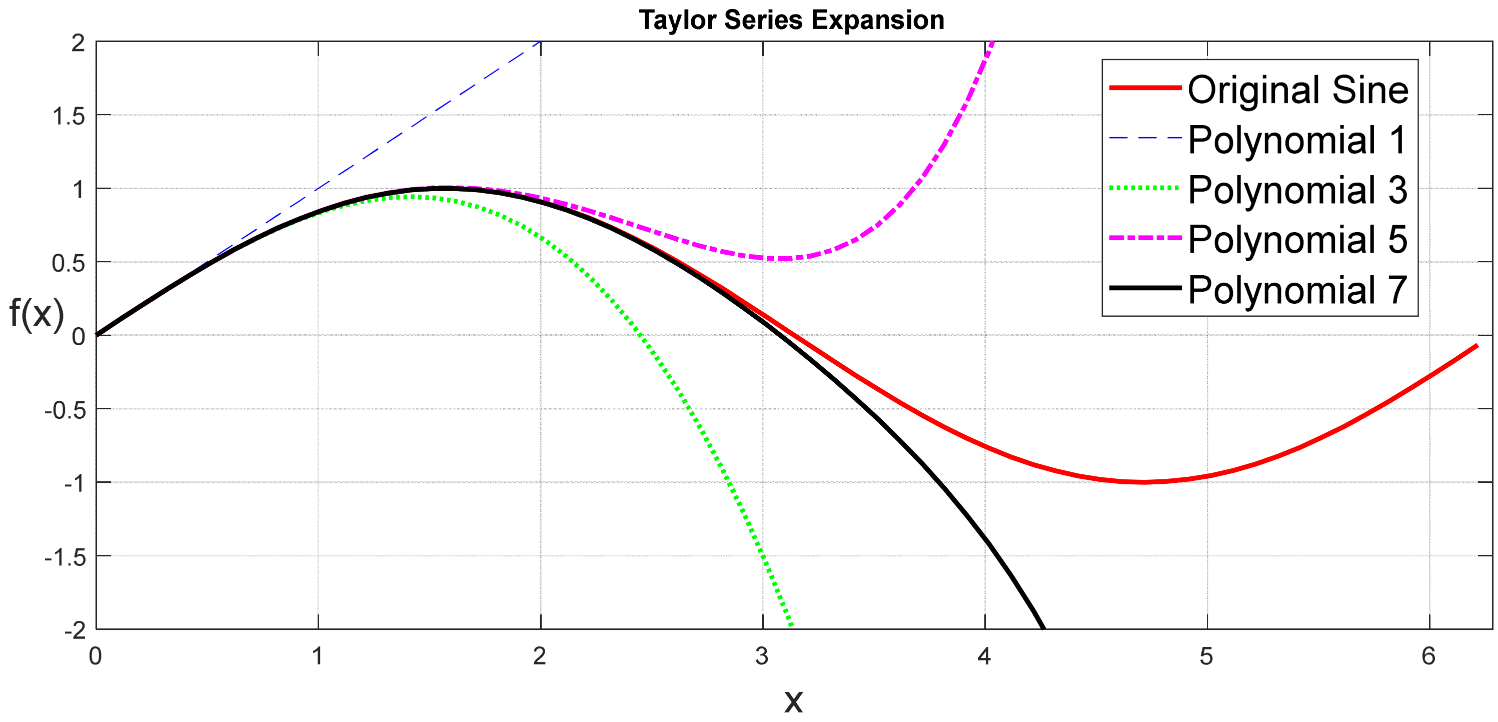
$$\boxed{f(t) \approx a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + \cdots + a_n(t - t_0)^n}$$

$$f(t) \approx 0 + t + 0 - \frac{1}{3!}t^3 + \frac{1}{5!}t^5 - \frac{1}{7!}t^7 + \frac{1}{9!}t^9 + \cdots$$

Numerical Methods

Plotting of the developed taylor series for f(x) = sin(x) for different orders

$$f(t) \approx 0 + t + 0 - \frac{1}{3!}t^3 + \frac{1}{5!}t^5 - \frac{1}{7!}t^7 + \frac{1}{9!}t^9 + \cdots$$



Numerical Methods

Assuming that the state variables are continuous and

$$\overset{future}{h = t} - \overset{now}{t^*} \Longrightarrow t = t^* + h \qquad f(t) \approx a_0 + a_1(t - t_0) + \cdots$$

$$x(t) \approx x_i(t^* + h)$$

$$x(t) \approx x_i(t^*) + \underbrace{\frac{dx_i(t^*)}{dt}}_{\text{remember } \boxed{f_i(t^*)}} \cdot h + \frac{d^2x_i(t^*)}{dt^2} \cdot \frac{h^2}{2!} + \frac{d^3x_i(t^*)}{dt^3} \cdot \frac{h^3}{3!} + \cdots$$

Plugging in the model equation:

$$x_i(t^* + h) \approx x_i(t^*) + f_i(t^*) \cdot h + \frac{df_i(t^*)}{dt} \cdot \frac{h^2}{2!} + \frac{d^2f_i(t^*)}{dt^2} \cdot \frac{h^3}{3!} + \cdots$$

✓ The order of the discretization is given by the highest power of h

✓ As shown before, the higher the order, the more accurate the solution

Numerical Methods

Continuous solver operation: Forward Euler

Truncate the Taylor Series after the first term:

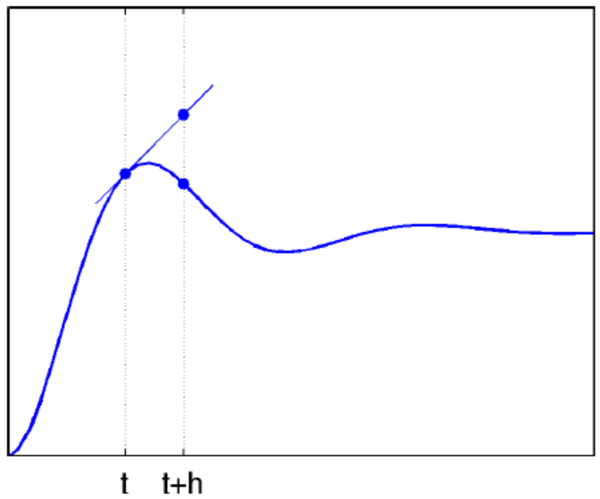
$$x(t^* + h) \approx x(t^*) + h \cdot f(x(t^*), t^*)$$

$$\boxed{x_{k+1} = x_k + h \cdot f(x_k, y_k)}$$

- ✓ 1st order accurate

Explicit integration algorithm

- ✓ Calculate the next states values in a single step using the system differential equations and the previous state values



FE Numerical experiment

- Scalar system

  $\dot{x} = a \cdot x$

  $f(x,t) = a \cdot x$

- Analytical solution

  $x(t) = x_0 \cdot e^{a \cdot t}$

- Use 'our' Forward Euler $\boxed{x_{k+1} = x_k + h \cdot a \cdot x_k}$

k	x_{k+1}
0	$x_1 = x_0 + h \cdot a \cdot x_0$
1	$x_2 = x_1 + h \cdot a \cdot x_1$
2	$x_3 = x_2 + h \cdot a \cdot x_2$
3	$x_4 = x_3 + h \cdot a \cdot x_3$

$x_1 = x_0(1 + ah)$

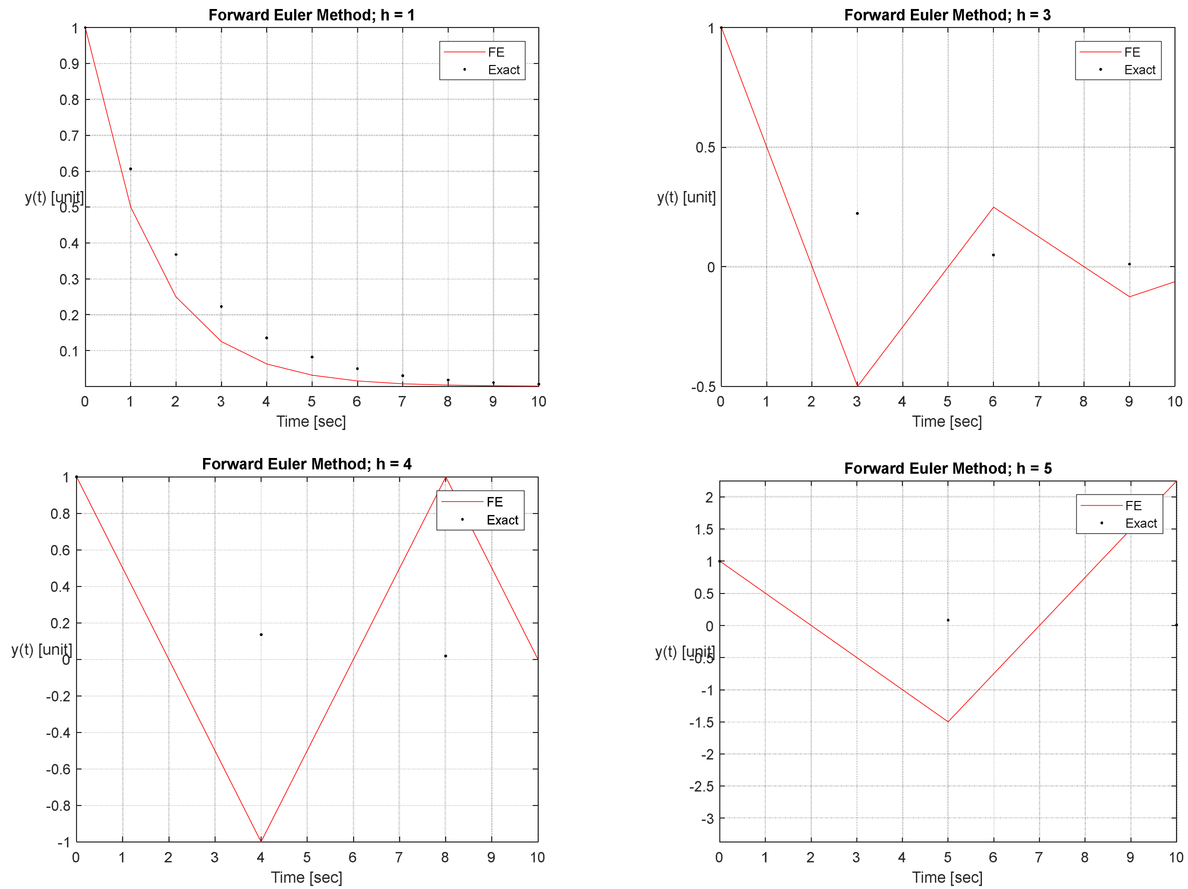
$x_2\ (1 + ah) = x_0(1 + ah)(1 + ah) = x_0(1 + ah)^2$

$\boxed{x_{k+1} = x_0(1 + ah)^k}$

FE Numerical experiment

$$f(x,t) = a \cdot x$$

- ✓ Variation of time step h



If h > 2, then the solver solution is unstable for a = -0.5

- ✓ **Explicit Solvers** are unstable when used to solve a stiff system unless its time step is set to a prohibitively small value

Backward Euler

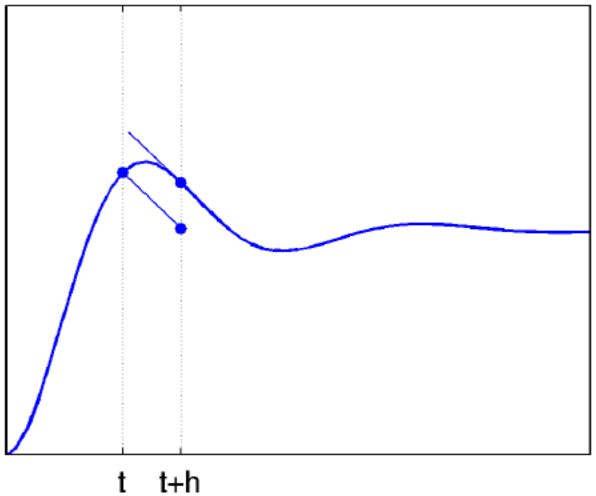
- Develop Taylor Series around a time in the future:

$$\boxed{x(t^* + h)} \approx x(t^*) + f\boxed{(x(t^* + h),}\, t^* + h) \cdot h$$

$$x_{k+1} = x_k + h \cdot f(x_{k+1}, y_{k+1})$$

- Must be solved iteratively for the unknown x(t*+h)

- Implicit integration algorithm
  - ✓ Iterative Estimation of the next state to calculate the next state until the estimate is equal to the calculated result



Numerical experiment (revisited)

- Scalar system

$$\dot{x} = a \cdot x$$

$$f(x,t) = a \cdot x$$

- Analytical solution

$$x(t) = x_0 \cdot e^{a \cdot t}$$

- Backward Euler

$$\boxed{x_{k+1} = x_k + h \cdot f(x_{k+1}, t_{k+1})}$$

k	x_{k+1}
0	$x_1 = x_0 + h \cdot a \cdot x_1$
1	$x_2 = x_1 + h \cdot a \cdot x_2$
2	$x_3 = x_2 + h \cdot a \cdot x_3$
3	$x_4 = x_3 + h \cdot a \cdot x_4$

$$x_1 = \frac{1}{(1+ah)} \cdot x_0$$

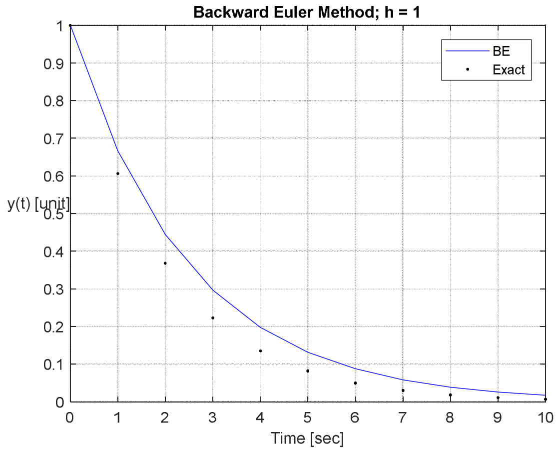
$$x_2 = \frac{1}{(1+ah)} \cdot x_0 + ah * x_2$$

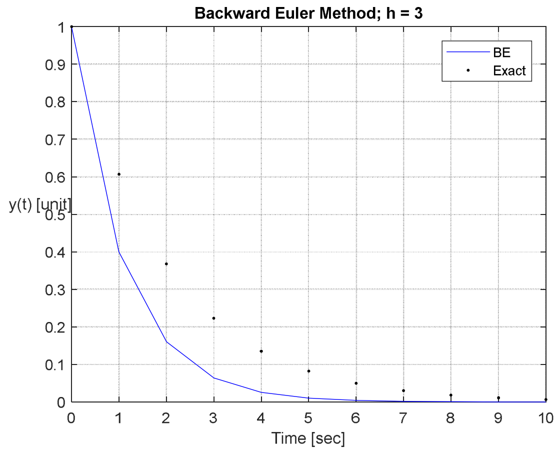
$$\boxed{x_k = \frac{1}{(1+ah)^k} \cdot x_k}$$

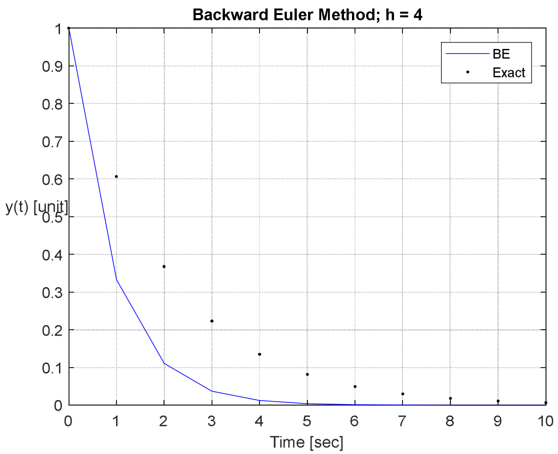
BE Numerical experiment

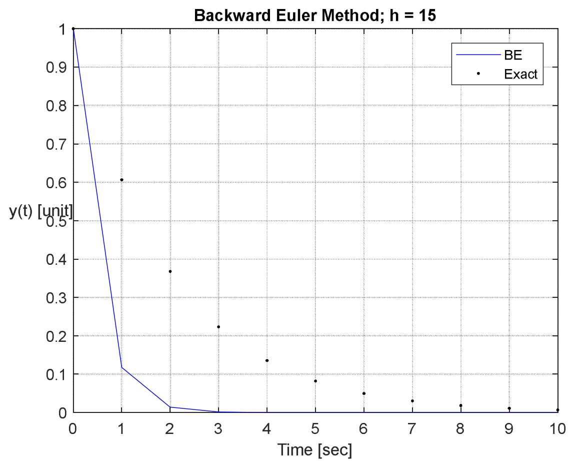
$$f(x,t) = a \cdot x$$

- ✓ Variation of time step h



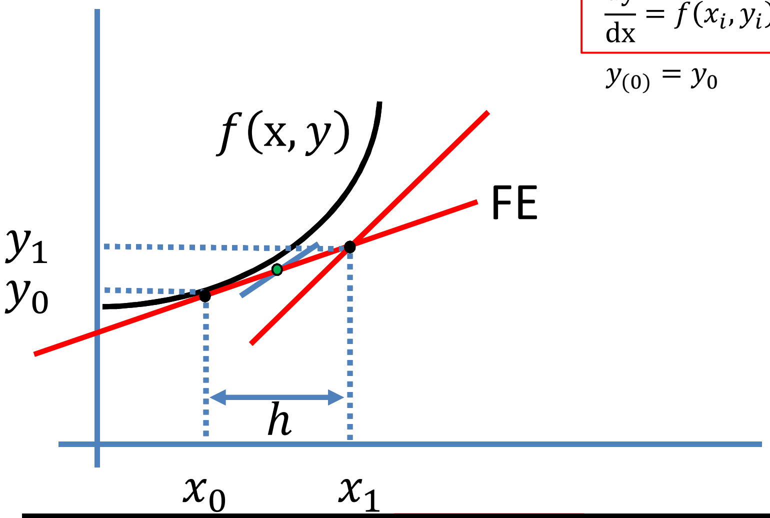






- ✓ Stability region is larger than that of a explicit solver
- ✓ Larger time steps can be taken without encountering instability problems

Runge-Kutta



$$\boxed{\frac{dy}{dx} = f(x_i, y_i)}$$

$$y_{(0)} = y_0$$

FE: $\boxed{y_1 = y_0 + h \cdot f(x_0, y_0)}$

Runge-Kutta 2nd Order

$$\boxed{y_{i+1} = y_i + (a_1 \cdot k_1 + a_2 \cdot k_2)h}$$

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1 \cdot h, y_i + q_{11} \cdot k_1 \cdot h)$$

$$a_1 + a_2 = 1; \quad a_2 \cdot p_1 = \frac{1}{2}; \quad a_2 \cdot q_{11} = \frac{1}{2}$$

Heuns-Method $\boxed{a_2 = \frac{1}{2}}$

$$a_1 + a_2 = 1 \Rightarrow a_1 = \frac{1}{2} \qquad a_2 \cdot p_1 = \frac{1}{2} \Rightarrow p_1 = 1$$

$$a_2 \cdot q_{11} = \frac{1}{2} \Rightarrow q_{11} = 1$$

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + h, y_i + k_1 \cdot h)$$

$$y_{i+1} = y_i + \left(\frac{1}{2} k_1 + \frac{1}{2} \cdot k_2\right) h$$

Midpoint-Method $\boxed{a_2 = 1}$

$$a_1 + a_2 = 1 \Rightarrow a_1 = 0 \qquad a_2 \cdot p_1 = \frac{1}{2} \Rightarrow p_1 = \frac{1}{2}$$

$$a_2 \cdot q_{11} = \frac{1}{2} \Rightarrow q_{11} = \frac{1}{2}$$

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + k_1 \cdot \frac{h}{2}\right)$$

$$y_{i+1} = y_i + k_2 \cdot h$$

Numerical Methods

Stiffness

It depends on the differential equation, the initial conditions, and the numerical method. Dictionary definitions of the word stiffness involve terms like

- not easily bent
- rigid
- stubborn

A problem is stiff if the solution being sought varies slowly, but there are nearby solutions that vary rapidly, so the numerical method must take small steps to obtain satisfactory results

Source: [4]

Numerical Methods Stiffness

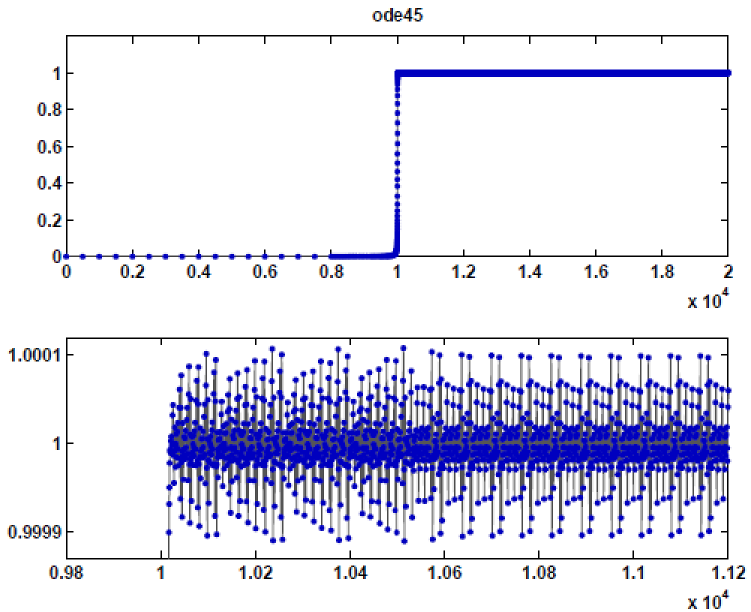
https://de.mathworks.com/company/newsletters/articles/stiff-differential-equations.html

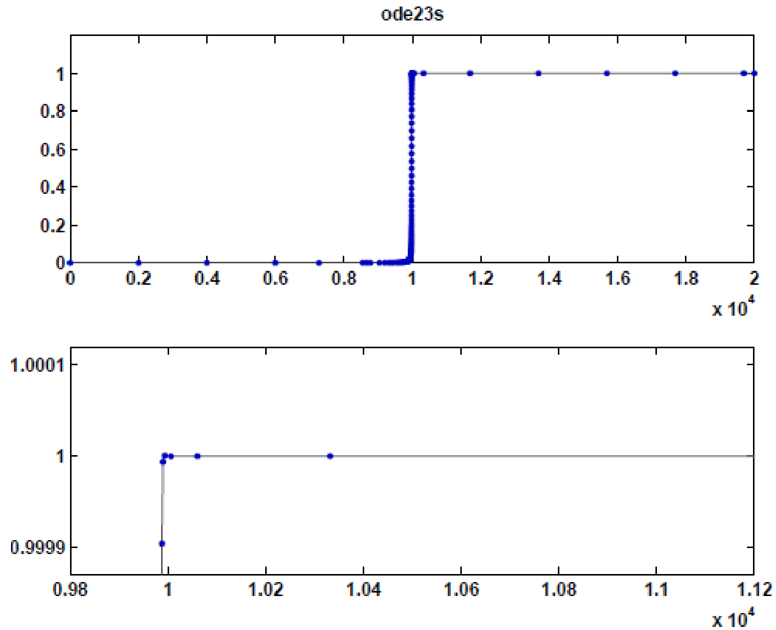
```
delta = 0.01;
F = @(t,y) y^2 - y^3';
opts = odeset('RelTol',1.e-4);
ode45(F,[0 2/delta],delta,opts);
```

Based on a model of flame propagation

```
delta = 0.00001;
ode45(F,[0 2/delta],delta,opts);
```

```
delta = 0.00001;
ode23s(F,[0 2/delta],delta,opts);
```





Source: [4] For more detail consider to take a look at the reference

Numerical Methods

Explicit (Non-stiff) Solver

- Calculate the next states values in a single step using the system differential equations and the previous state values
- Unstable when used to solve a stiff system unless its time step is set to a prohibitively small value

Implicit (Stiff) Solver

- Require multiple iteration steps to calculate the next state values
- Iterative Estimation of the next state to calculate the next state until the estimate is equal to the calculated result
- More computations less efficient than a non-stiff solver
- Stability region is larger than that of a non-stiff solver
- Larger time steps can be taken without encountering instability problems

Source: http://www.mathworks.de

Numerical Methods

Errors during Simulation with Numerical Methods

- True value = Approximation + Error
- Error = True value – Approximation

There are mainly two types of errors in numerical methods:

1. Truncation error
2. Round off error

Source: [2]

Numerical Methods

1. Truncation error

Is the error created by truncating (approximating) a mathematical procedure

1. Maclaurin Series

$$e^x = 1 + \mathrm{x} + \frac{\mathrm{x}^2}{2!} + \frac{\mathrm{x}^3}{3!} + \cdots$$

$$e^{0.5} = \boxed{1 + 0.5 + \frac{0.5^2}{2!}} + \boxed{\frac{\mathrm{x}^3}{3!}} + \cdots$$

Only first two terms are used for approximation

Truncation error! Are the remaining

2. Derivative function

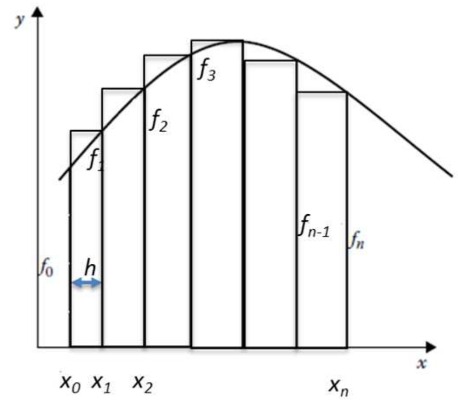
$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

$$f'(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

Original method requires to use Δx approaching zero!

But you use a finite number for Δx!

3. Integrate a function



$$y_{rect} = \int_a^b f(x)dx$$

Original method requires to use infinite number rectangles

But you use finite number of rectangels!

Numerical Methods

2. Round off error

For example, we know that the value of π is

$\pi = 3.141592653897285\ldots$

if we are using a computer that can retain only seven significant bits so this computer might store or use π as

$\pi = 3.141592$

resulting in an round off error

0.00000065

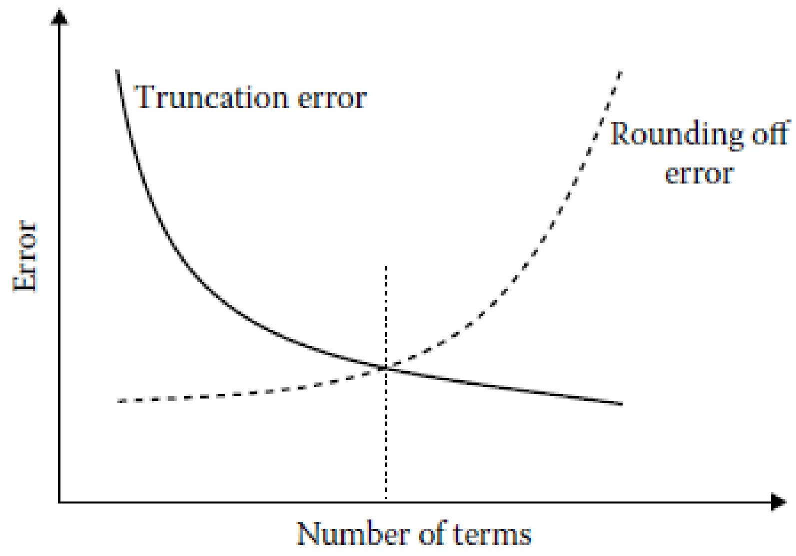
Source: [2]

FH | JOANNEUM
University of Applied Sciences

Numerical Methods

Truncation vs. Round off error

- At large step size (h), the error is dominated by the truncation error, whereas
- the round-off error dominates at small step size
- ultimate accuracy of Euler's method (or any other integration method) is determined by the accuracy, p, to which floating-point numbers are stored on the computer performing the calculation



Source: [2]

Error estimation and step size control

Problem:

- Without prior knowledge of the *true* solution, how can we estimate the error of the *numerical* solution?

Solution:

- ✓ Perform the same integration step with two integration methods of different order (e.g. 4 and 5)

$$\boxed{\varepsilon = |x_{nth} - x_{mth}| \leq tol_{rel}}$$

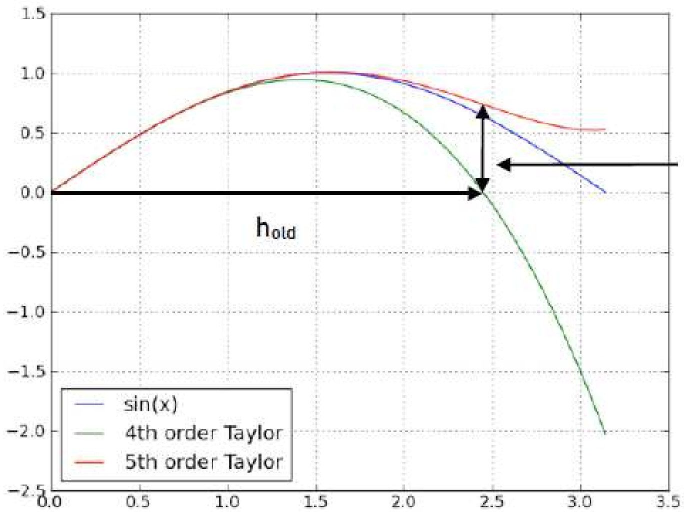
ε...	*R*elative error
x_{nth} and x_{mth} ...	Solution of method 1 and 2
tol_rel...	Relative error tolerance (defined by the user)

Numerical Methods

<u>Adaptive step size capabilities:</u>

Calculation of the next step size “h” is based on the relative tolerance and the relative error (ε) as follows:

$$\boxed{h_{new} = \left(\frac{tol_{rel}}{\varepsilon}\right)^{1/n_{max}} \cdot h_{old}}$$



$$\varepsilon = \frac{|x_{5th} - x_{4th}|}{\max(|x_{5th}|, |x_{4th}|)}$$

ε …	Relative error
tol_rel…	Relative tolerance
h_old…	Previous time step

Principle of Variable Step Solver

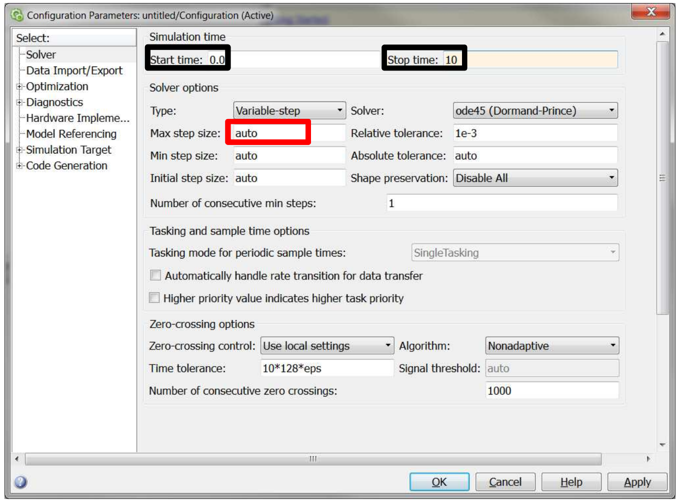
- ✓ Goal: Keep error within acceptable error limits
- ✓ Key advantage: Accuracy directly specified by the user

Absolute error tolerance determines the accuracy when the solution approaches zero

FH | JOANNEUM
University of Applied Sciences

Numerical Methods

Variable-step Solver Configuration



- Max step size
  In case of „auto“ the timesteps are set to 50 steps!

$$\underset{stepsize}{\max} = \frac{t_{stop} - t_{start}}{time_{steps}}$$

- Min step size
  smallest time step
- Initial step size
  size of the first time step
- Relative tolerance (Start with 10^(-3) (0.1%),Numerical limit is 10^(-16)
  => largest acceptable solver error
  => solver reduces the time step if tolerance is exceeded
- Absolute tolerance
  Best to set to auto $tol_{abs} = \max(|x|) \cdot tol_{rel}$

Source: http://www.mathworks.de

Numerical Methods

Variable-step Solver Tipps

- *In case you are concerned the solver missing significant behavior, change the parameter preventing the solver taking too large steps*

- *At long time spans default step size might be too large for the solver to find solution*

- *If model contains periodic behavior, set the maximum step size to some fraction (such as 1/4) of that period.*

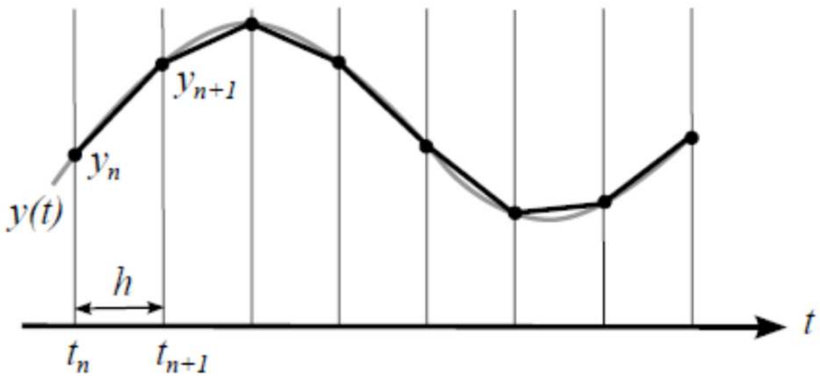
Source: http://www.mathworks.de

FH | JOANNEUM
University of Applied Sciences

Numerical Methods

Discrete Solver

Are based on trapezoidal integration method, approximating the transient response of a continuous system between two adjacent points with a line segment



System dynamics:

$$\frac{d_y}{dt} = f\ (t)$$

Trapezoidal rule:

$$y_{n+1} = y_n + \frac{h}{2}(f(t_n) + f(t_{n+1}))$$

Accuracy depends on integration sample time, h, and transient frequency of system

- ✓ Golden rule: set the integrator sample time 10 times higher then the period of the highest frequency in the system

Source: http://www.mathworks.de

Numerical Methods

Time step selection using fixed-step solver

- Accuracy is indirectly determined by the time step
  - To ensure accuracy, reduce the time step and observe any changes in the output => increases calculation time!
  - Or, compare with a continuous simulation
  - Limit for minimum step size => restricted resources: CPU time, memory size…
- Continuous waveform
  - Highest transient frequency constrains sample time
  - Set t_sample smaller t_transient/10
- Switched system
  - Switches turned on at sample instants
  - Set t_sample smaller than t_sw/100

Numerical Methods

Variable-Step Solver

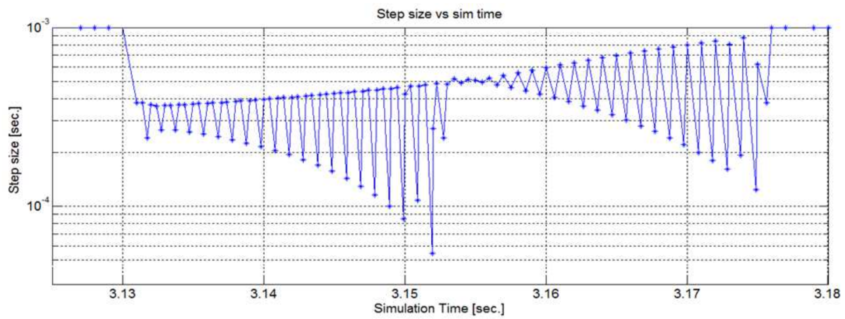
Vary step size during simulation depending on model dynamics.

- If states changing rapidly => reduce step size => increasing accuracy
- If states changing slowly => increase step size => avoid taking unnecessary steps

– Computing step size adds computational overhead

✓ But reduce total number of steps and simulation time

Variable-Step size example



Fixed Step Solver

- solve the model at regular time intervals

✓ Decreasing step size increases accuracy => increases calculation time!

– Limit for minimum step size => restricted resources: CPU time, memory size…

✓ Requirement for Real-time systems and code generation, unless you use an S-function or RSim target

Source: http://www.mathworks.de

Integral form of ODE

Modeling/Simulation software like MATLAB/Simulink solves the ODE by numerical methods that approximate iteractively the solution

- These methods are the so-called „solvers“, e.g. ODE23, ODE45,...

- ❑ Important
  - ✓ Solvers work on the integrators of the model
  - ✓ For the implementation in Simulink and other tools:
    - ✓ **ODE must be represented in the integral form**
    - ✓ **How to:** The highest derivate must be placed alone in one side of the equation and integrate at both sides as many times as the highest term‘s order

Numerical Methods

(Matlab) solvers are for solving problems of the following form

- ***O***rdinary ***D***ifferential ***E***quation => ODE

$$\frac{dx_1}{dt} = f_1(x_1, x_2, x_3 \dots, t)$$

$$\frac{dx_2}{dt} = f_2(x_1, x_2, x_3 \dots, t) \qquad \text{i.e.} \qquad \frac{dx}{dt} = f\,(x, u, t)$$

$$\frac{dx_3}{dt} = f_3(x_1, x_2, x_3 \dots, t)$$

Fixed-Step	Variable-Step
Ode1	Ode45
Ode2	Ode23
Ode3	Ode113
Ode4	Ode15s
Ode5	Ode23s
Ode8	Ode23t
	Ode23tb

Numerical experiment

Task:

- Based on the FE method try to solve the following equation in Excel and analyse the result in contrast to the exact solution

- Based on the previous slide try to integrate the Runge-Kutta Midpoint method in Excel and analyse the result in contrast to the exact solution

    - What observations have you made?

$$f(x,t) = a \cdot x \qquad\qquad y(0) = 1 \qquad a = -0.5$$

$$x(t) = x_0 \cdot e^{a \cdot t}$$

Reference sources

- [1] Schäuffele, J; Zurawka, T. "Automotive Software Engineering", SAE International
- [2] Chaturvedi, D.K. "Modeling and Simulation of System Using Matlab and Simulink", CRC Press
- [3] Aarenstrup, R. “Managing Model-Based Design”. Mathworks
- [4] Cleve Moler, “Numerical Computing with MATLAB”; https://de.mathworks.com/moler/chapters.html

FH JOANNEUM
University of Applied Sciences

ECE – MBD [WS2020/21]

Institute of Electronic Engineering
FH JOANNEUM - University of Applied Sciences
Kapfenberg and Graz, AUSTRIA

Dipl.-Ing. (FH) Alfred Steinhuber, MSc
Tel.: +43 (0) 316 5453 6345
alfred.steinhuber@fh-joanneum.at