Numerical solution of ODEs

Prof. Dr. Michael Möckel

TH Aschaffenburg
university of applied sciences

Recap:
ODEs as vector fields

- Example:

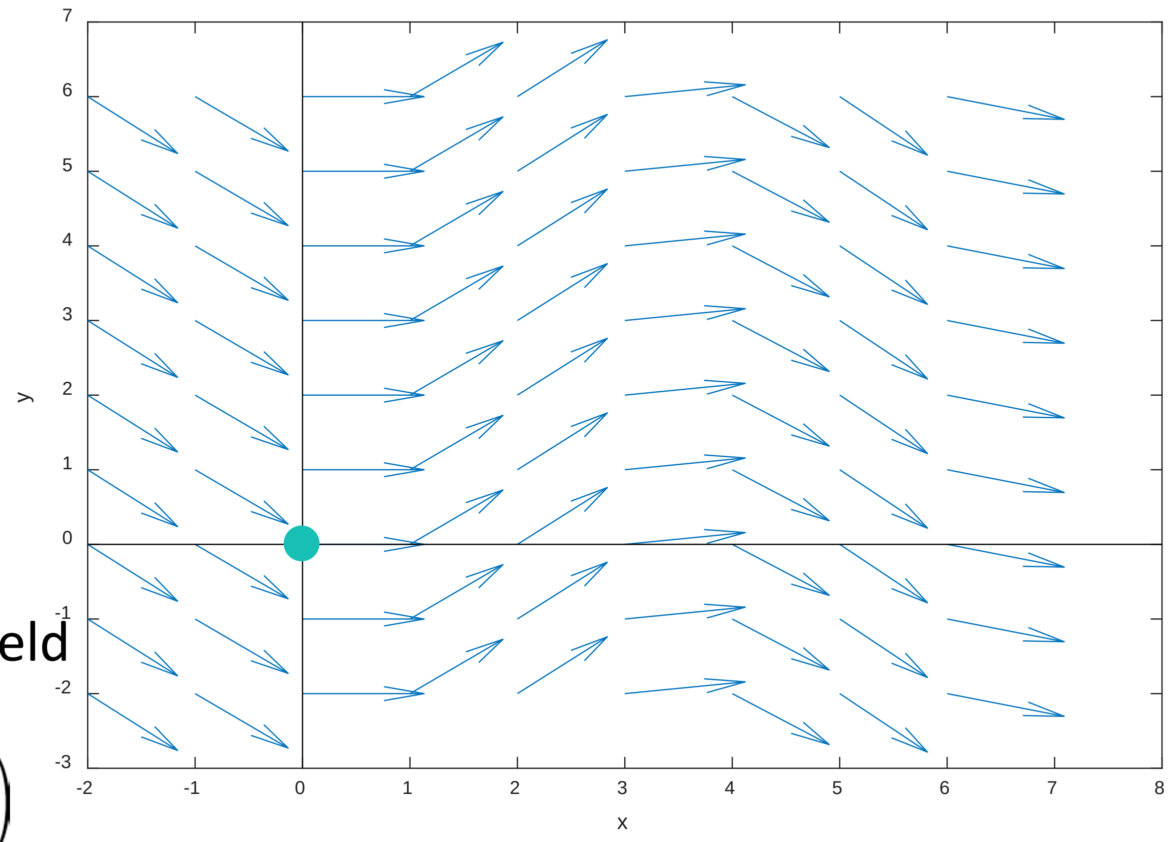
$$\frac{df(x)}{dx} = \sin(x)$$
$$y(0) = 0$$

- Analytical Solution:

$$f(x) = 1 - \cos(x)$$

- Representation as vector field

$$\vec{v} = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \Delta x \begin{pmatrix} 1 \\ \frac{\Delta y}{\Delta x} \end{pmatrix}$$



- Vectors represent tagents to the unknown function f(x)

Recap:
ODEs as vector fields

- Example:

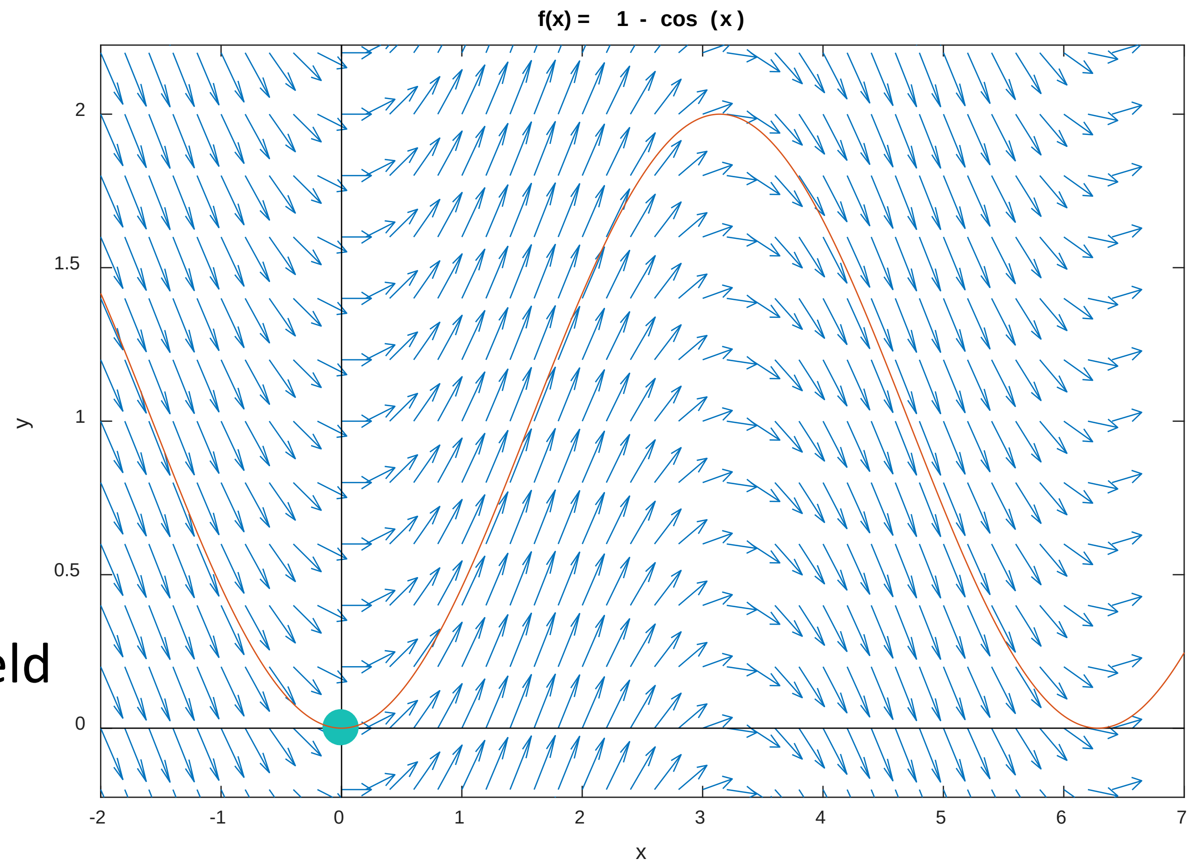
$$\frac{df(x)}{dx} = \sin(x)$$
$$y(0) = 0$$

- Analytical Solution:

$$f(x) = 1 - \cos(x)$$

- Representation as vector field

$$\vec{v} = \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \Delta x \begin{pmatrix} 1 \\ \frac{\Delta y}{\Delta x} \end{pmatrix}$$



- Vectors represent tagents to the unknown function f(x)

MATLAB – Code

```
% Vector field (normalized)
DGL = @(x,y) sin(x)
delta = 0.2;
[X Y] = meshgrid(-2:delta:6.5, -2:delta:6.5);
dY = DGL(X,Y);
dX = ones(size(dY));
L=sqrt(1+dY.^2);
quiver(X, Y, dX./L, dY./L);

% Exact Solution
sol = dsolve('Dy = sin(x), y(0) = 0','x');
pretty(sol);  fplot( sol, [-2, 7.5] )
```

```
% Graphics
hold on
xL = xlim;
yL = ylim;
line([0 0], yL, 'Color', 'black');
line(xL, [0 0], 'Color', 'black');

% Plot initial value
scatter(0,0, 200, 1, 'filled');
```

Euler – Cauchy - Method

- Construct the unknown function as a polygone

- Initial point

  (x(1), y(1)) = (0, y0)

- Next point:

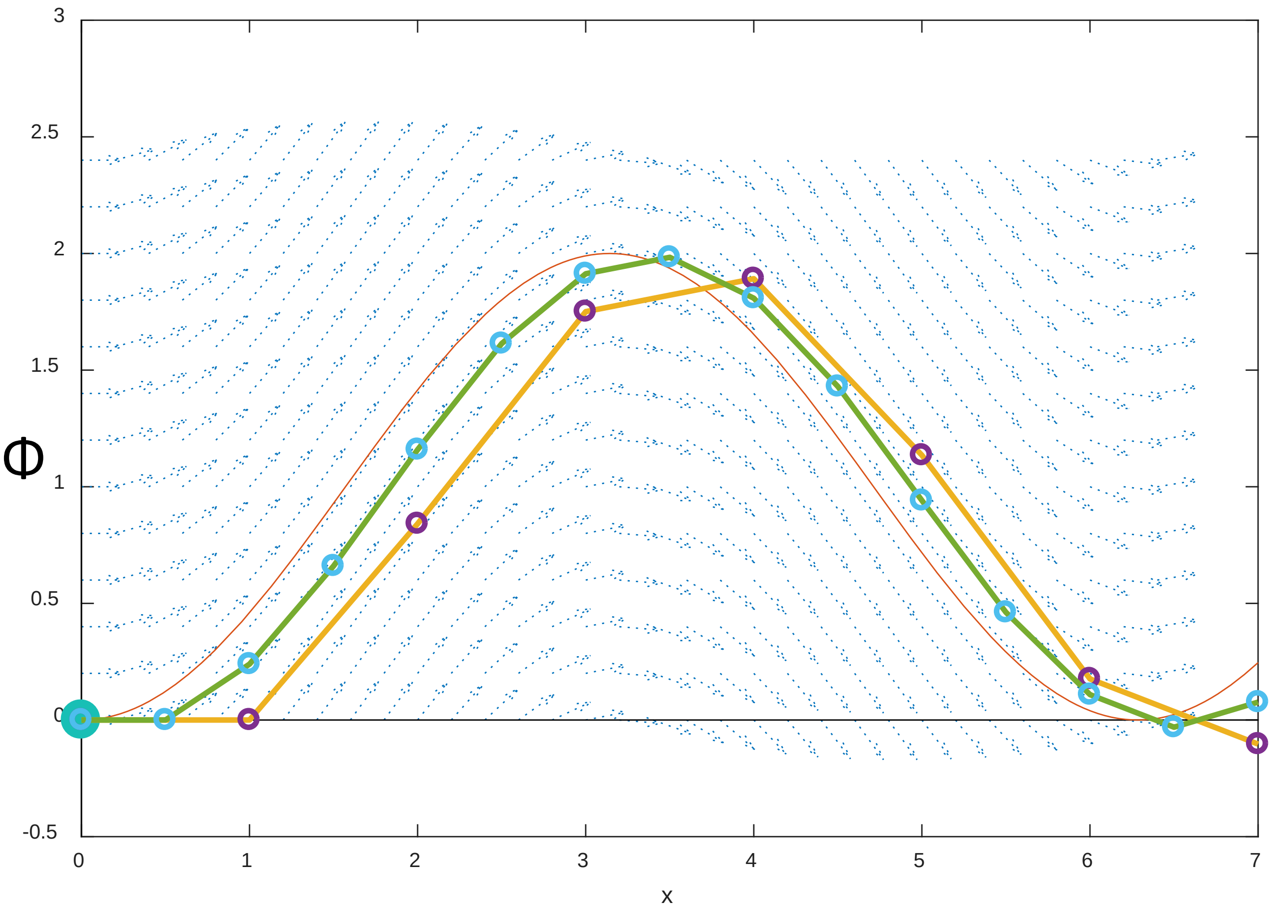
  x(i+1) = x(i) + h

- Incremental function Φ

$$y(i+1) = y(i) + h\Phi(x(i), y(i), h)$$

- Evaluation of ODEL

  Φ = DGL(x(1), y(1))



MATLAB - Code for Euler-Cauchy

```
function [ x, y ] = odeEULER_C(DGL,a,b,h,y0)
% x, y: Vectors with N+1 components,
% h: stepsize, [a,b] intervall, y0: Initial val

N = floor((b-a)/h);
x(1) = a;
y(1) = y0;
for i = 1:N
    x(i+1) = x(i) + h;
    y(i+1) = y(i) + DGL(x(i), y(i))*h;
end
end
```

Euler – Cauchy - Method

- Discussion:
- Error analysis:

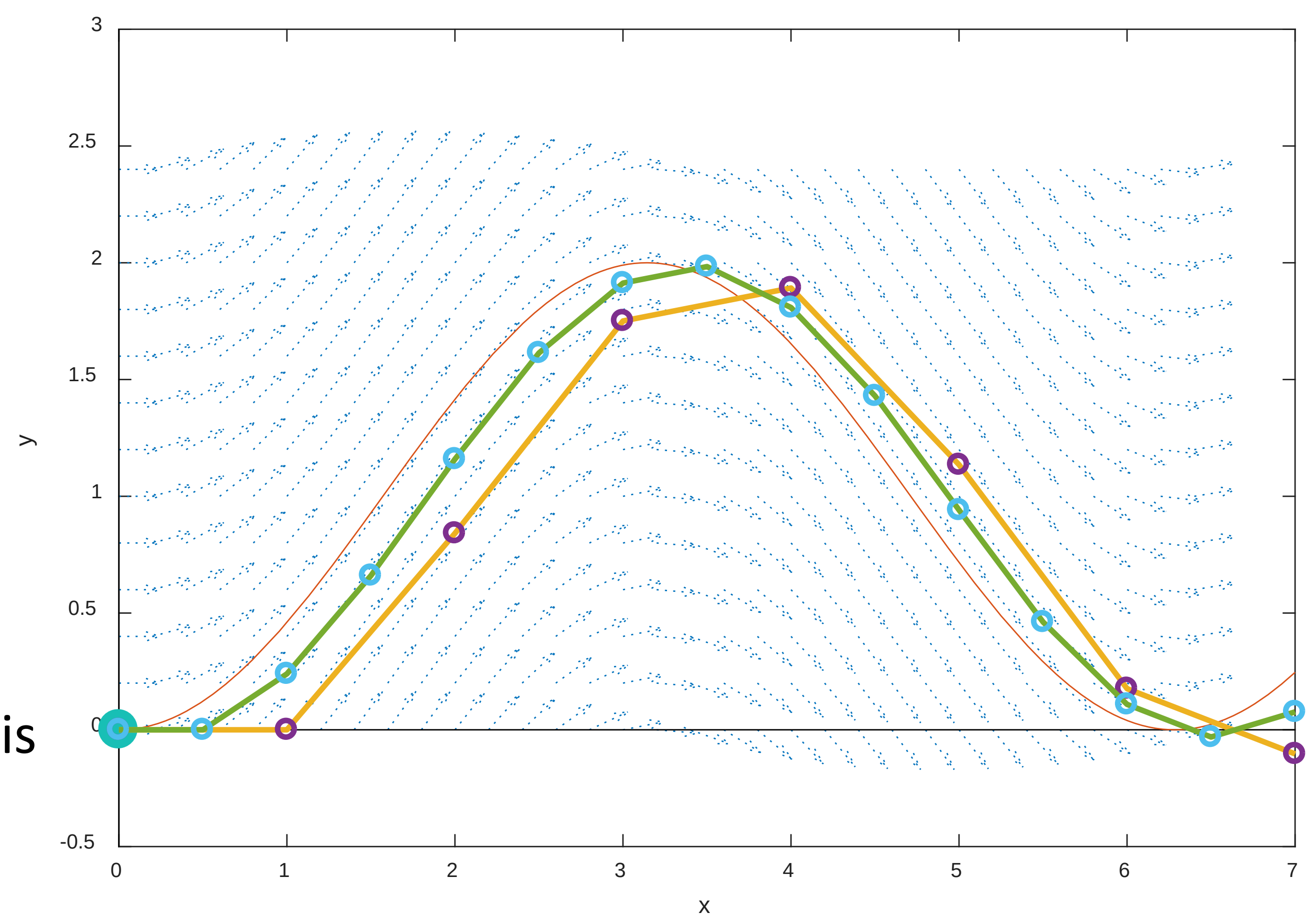
  - inhomogeneous
  - estimated by 2nd order Taylor-expansion

$$\epsilon = \frac{d^2 f(x)}{dx^2}\bigg|_{x=\xi} \frac{h^2}{2}$$

- Problem statement:

Slope of each section is determined by single point (x(i), y(i)) only

Euler-Cauchy-method is “1-step-method”



Classification of Methods for Numerical Solutions of ODEs

Today discussed:	Other methods:
1-step method (SSM) Only one previously calcula-ted point used in every step	Multistep methods (MSM) Several previously calculated points used in every step
Explicit Forward computation y(i+1) = F(x(i), y(i))	Implicit Iterative computation y(i+1) = F(x(i), y(i+1))
ODEs of 1st order	ODEs of higher orders
Example: (impr.) Euler, Runge-Kutta	Example: Adams-Bashford (Multi-Step)

Improved Euler Method

Improvement:

Take mean slope at 2 points:

[x(i), y(i)]   (initial) ,   [x(i+1), ?]

- Temporary determination

  y(i+1) = y(i) + DGL(x(i), y(i)) * h

  (Incremental function Euler-Cauchy)

- Improved Incremental function:

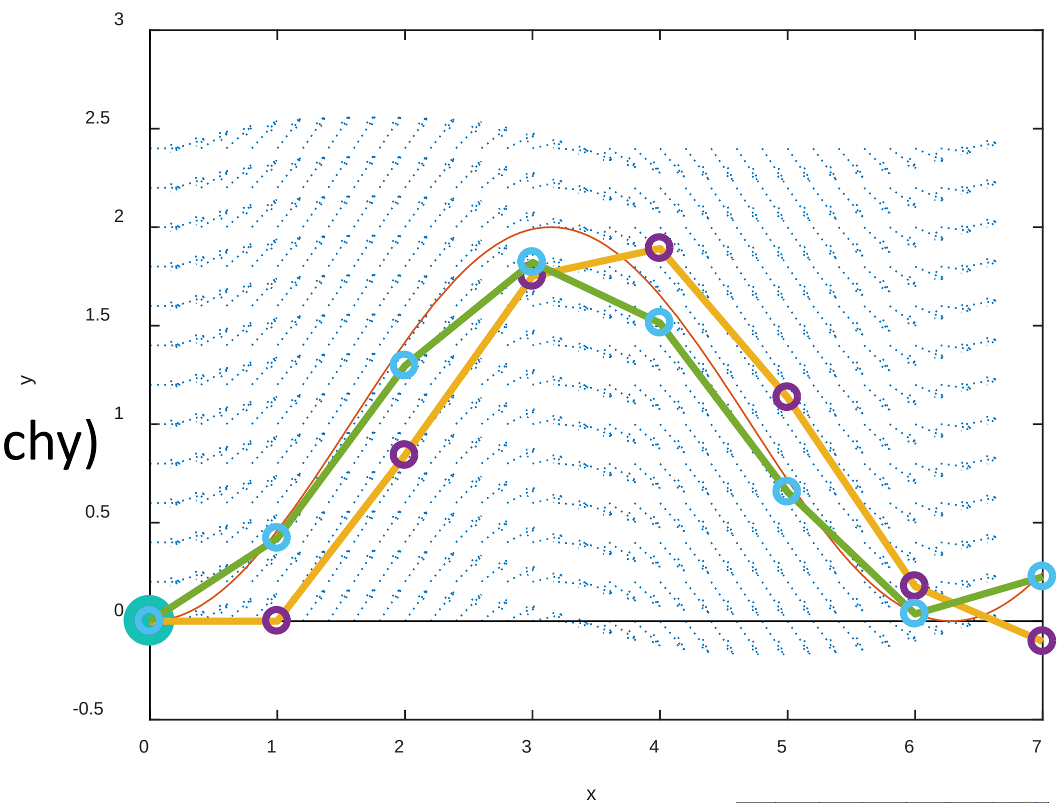
  Mean value of slopes

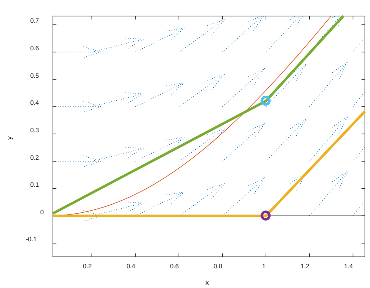
  Φ = [DGL(x(i), y(i)) +

  DGL(x(i+1), y(i+1))] / 2

- Final determination of new point:

  y(i+1) = y(i) + Φ * h





MATLAB Code
Improved Euler method

```
function [ x, y ] = odeEULER_verb(DGL,a,b,h,y0)

N = floor((b-a)/h); x(1) = a; y(1) = y0;

for i = 1:N
    x(i+1) = x(i) + h;
    m1 = DGL(x(i), y(i));
    y(i+1) = y(i) + m1*h;

    m2 = DGL(x(i+1), y(i+1));
    y(i+1) = y(i) + (m1+m2) / 2 *h;
end
end
```

Numerical solution of ODEs and Quadrature Rules for Numerical Integration

Exact determination of a point which is propagated by stepsize h

$$\boldsymbol{y}(t+h) = \boldsymbol{y}(t) + [\boldsymbol{y}(t+h) - \boldsymbol{y}(t)] = \boldsymbol{y}(t) + \int_t^{t+h} \boldsymbol{y}'(s)\, ds$$

$$= \boldsymbol{y}(t) + h \int_0^1 \boldsymbol{y}'(t + \tau h)\, d\tau.$$

$$s = t + \tau h$$
$$0 \le \tau \le 1$$

Quadrature Rules	Num. Solution of ODEs
Riemann integration with (left-point) rectangles	Euler - Cauchy - method
Trapezoidal rule	improved Euler - methode
General quadrature rule	Runge-Kutta-methods (various steps and orders)

Runge – Kutta – Methods (1-step Methods!)

- Goal: Systematic improvement by multiple calling of ODE
- Single step method: Only one previously known point is used (x(i), y(i)))
- Construction of temporary intermediate points within a step h
- Set of interm. points S = Number of temporary interm. points (x(i,j), y(i,j))

$$x(i,j) = x(i) + a(j)h$$

j: Index of intermediate points

$$y(i,j) = y(i) + \sum_{l=1}^{j-1} b(j,l)K(i,l)$$

y-values of intermediate points

$$K(i,j) = DEG(x(i,j), y(i,j))$$

K: evaluation of ODE at intermediate points

- Improves incremental function for new function value at x(i+1) = x(i) +h

$$y(i+1) = y(i) + \Phi(x(i), y(i), h) = y(i) + h\sum_{j=1}^{S} c(j)K(j)$$

Runge-Kutta-Method Parameters

- Runge-Kutta Equations

$$
\begin{aligned}
x(i,j) &= x(i) + a(j)h \\
y(i,j) &= y(i) + \sum_{l=1}^{j-1} b(j,l)K(i,l) \\
K(i,j) &= DEG(x(i,j), y(i,j)) \\
y(i+1) &= y(i) + \sum_{l=1}^{S} c(j,l)K(l)
\end{aligned}
$$

- Parameter ranges:

$$0 \leq a(j) \leq 1$$ Intermediate points within step-size

$$b_{j,l} = 0 \ \forall l \geq j$$ Explicit method

- Constraint for convergence:

$$\sum_{j=1}^{s} c_j = 1$$

- Arrangement of Parameters as Butcher-Tableau

$$
\begin{array}{c|ccccc}
a_1 & b_{11} & b_{12} & \cdots & b_{1s} \\
a_2 & b_{21} & b_{22} & \cdots & b_{2s} \\
a_3 & b_{31} & b_{32} & \cdots & b_{3s} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
a_s & b_{s1} & b_{s2} & \cdots & b_{ss} \\
\hline
 & c_1 & c_2 & \cdots & c_s
\end{array}
$$

Runge-Kutter Parameter for Improved Euler – Method

- Recap: Mean value of slope at initial point and final point

  (1) Intermediate and final point have same x-coordinate x = x(i) + h

  (2) Y-coordinate of intermediate point:

  y(i+1) = y(i) + DGL(x(i), y(i)) * h (Interm. point)

  (3) Φ = [DGL(x(i), y(i)) + DGL(x(i+1), y(i+1))] / 2 (Mean value)

  (4) y(i+1) = y(i) + Φ * h (improved final pt)

- Read off Runge-Kutta parameters

  (1) x-coordinates: a(1) = 0 (initial), a(2) = 1 (interm. point)

  (2) Interm. point: y(i,2) = y(i) + b(2,1) * K(i,1) * h

  K(i,1) = DEG(x(i), y(i)), d.h. b(2,1) = 1

  (3) Incremental function:

  y(i+1) = y(i) + ½ (K(I,1) + K(I,2)) h , d.h. c(1) = c(2) = ½

a	$B^{(2,2)}$	
0	0	0
1	1	0
	1/2	1/2

3rd Order Method by Heun

Interm. points a(j) $B^{(3,3)}$

$$
\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 0 \\
2/3 & 0 & 2/3 & 0 \\
\hline
 & 1/4 & 0 & 3/4
\end{array}
$$

Teste:

$$\sum_{j=1}^{s} c_j = 1$$

4th Order Method
Classical Runge-Kutter

a(j) B$^{(3,3)}$

0	0	0	0	0
$1/2$	$1/2$	0	0	0
$1/2$	0	$1/2$	0	0
1	0	0	1	0
	$1/6$	$2/6$	$2/6$	$1/6$

Numerically most Important Method
Fehlberg 4(5) - Method

0	0	0	0	0	0	0	
$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	0	Jointly used intermediate results make method efficient
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$	0	0	0	0	
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$	0	1	0	
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$	0	0	
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	0	
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0	4th Order
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$	5th Order

Runge – Kutta – Methods Classification

- Set of interm. points **S** of a Runge-Kutta method

  S = number of intermediate points = number of ODE evaluations

  S determines numerical effort of method

- Order **O** of a Runge-Kutta method:
  - Represent the incremental function by Taylor series expansion around (x(i), y(i)) in powers of step-size h
  - Determines precision of method (error)

- Set S and Order O are rather independent!

Set	Order	Kind	Example	Remarks
1	1	Explicit	Euler	Only method for these parameters
2	2	Explicit	Improved Euler	Several methods with S = O = 2 exist
4	4	Explicit	Runge-Kutter 4th Order	