MAT 460 Numerical Differential Equations

Spring 2016

Michael Karow, Andrea Dziubek

Lecture 11

Topics: Solutions to ODE's (ordinary differential equations)

Numerical solutions to ordinary differential equations (short introduction)

An explicit first order ordinary differential equation (ODE) is an equation of the form $y'(t) = f(t, y(t))$. If in addition an initial value $y_0 \in \mathbb{R}^n$ is specified it is called **initial value problem (IVP)**:

$$y'(t) = f(t, y(t)), \qquad y(t_0) = y_0, \qquad f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n. \qquad (*)$$

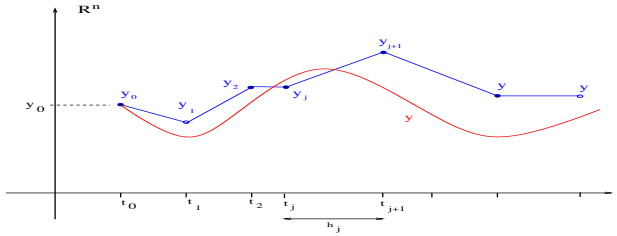
The solution $y : [t_0, t_e] \to \mathbb{R}^n$ to (*) can often only be obtained numerically.

Let y be the solution to $(*)$ and let $t_0 < t_1 < t_2 < \ldots < t_m = t_e$ a subdivision of the interval $[t_0, t_e]$. The goal is to approximate the sequence

$$y(t_0),\ y(t_1),\ y(t_2),\ y(t_3)\ldots,\ y(t_m)$$

as accurate as possible by a sequence of numerical values y_j:

$$y(t_0) = y_0,\quad y_1,\quad y_2,\quad y_3,\quad \ldots,\quad y_m$$

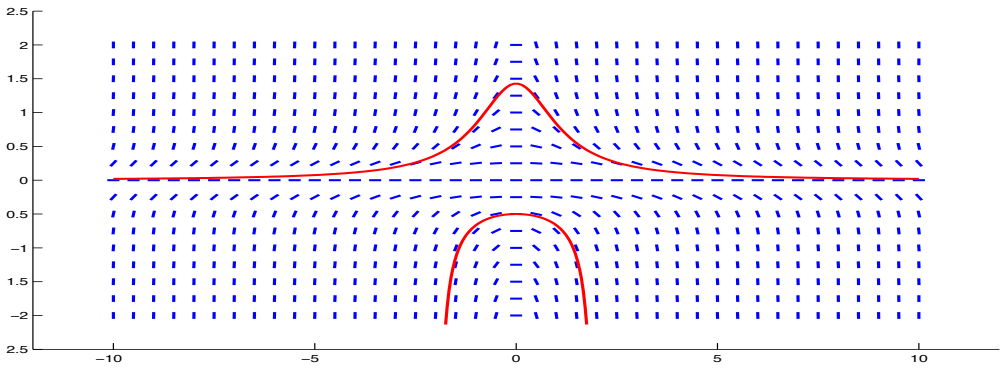


Remarks on ordinary differential equations I (scalar case):

$$\underset{\text{Slope of the solution}}{y'(t)} = \underset{\text{Prescribed slope } f \text{ at the point } (t, y(t))}{f(t, y(t))}$$

The function f describes a slope field (direction field).

Example: Slope field $f(t, y) = t\, y^2$ and two solutions to the ODE.



Remarks on ordinary differential equations II

Existence and uniqueness theorem of Picard-Lindelöf (1890):
Let $G \subseteq \mathbb{R} \times \mathbb{R}^n$ be an open domain and let $f : G \to \mathbb{R}^n$ continuous and locally **Lipshitz continuous** in the second argument of $^{(*)}$.
Then to every initial value $(t_0, y_0) \in G$ the ODE $y'(t) = f(t, y(t))$ has exactly one solution $y(t_0) = y_0$, where the graph does not end in G.

$(*)$ Local **Lipshitz continuous** means:
Every point $(\tilde{t}, \tilde{y}) \in G$ has a neighborhood $U \subseteq G$, and a constant $L \geq 0$ exist, such that

$$\|f(t, y_1) - f(t, y_2)\| \leq L \|y_1 - y_2\|$$

for all $(t, y_1), (t, y_2) \in U$. Here is $\|\cdot\|$ any norm in $\mathbb{R}^n$ and $L \geq 0$ depends on U.

Remarks on ordinary differential equations III

Let f be continuous. Then the ODE

$$y'(t) = f(t, y(t)) \qquad (*)$$

is equivalent to the integral equation

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

The proof of the existence and uniqueness theorem of solutions to initial value problems and derivation of solution methods rely on integral calculus.

Proof of the equivalence:
$(**) \Rightarrow (*)$ follows from the fundamental theorem of calculus.
$(*) \Rightarrow (**)$. A change of variables $t \to \tau$ gives $y'(\tau) = f(\tau, y(\tau))$.
Integrating gives

$$y(t) - y(t_0) = \int_{t_0}^{t} y'(\tau)\, d\tau = \int_{t_0}^{t} f(\tau, y(\tau))\, d\tau$$

By adding $y(t_0)$ to both sides we get $(**)$.

Remarks on ordinary differential equations IV

Higher order ODE's can be written as first order ODE's,
by introducing the lower order variables as new variables.

Example: Consider the ODE $x''(t) = g(t, x(t), x'(t))$. Set $v(t) = x'(t)$. Then

$$\frac{d}{dt} \underbrace{\begin{bmatrix} x(t) \\ v(t) \end{bmatrix}}_{y(t)} = \underbrace{\begin{bmatrix} v(t) \\ g(t, x(t), v(t)) \end{bmatrix}}_{f(t, y(t))}.$$

Method Types

Initial value problem:

$$y'(t) = f(t, y(t)), \qquad y(t_0) = y_0.$$

Goal: Approximate the sequence

$$y(t_0),\ y(t_1),\ y(t_2),\ y(t_3) \ldots,\ y(t_m)$$

by a sequence which is easy to compute

$$y(t_0) = y_0,\ y_1,\ y_2,\ y_3,\ \ldots,\ y_m$$

Notation: $h_j = t_{j+1} - t_j$ (step size)

The terms of the sequence y_j are computed recursively.

Fundamental Types:

y_{j+1}	$=$	$\psi(f, t_j, y_j,\ h_j)$	explicit one-step-methods
y_{j+1}	$=$	$\psi(f, t_j, y_{j+1}, y_j,\ h_j)$	implicit one-step-methods
y_{j+1}	$=$	$\psi(f, t_j, y_j, \ldots, y_{j-p},\ h_j)$	explicit multi-step-methods
y_{j+1}	$=$	$\psi(f, t_j, y_{j+1},\ y_j, \ldots, y_{j-p},\ h_j)$	implicit multi-step-methods

Implicit methods are usually numerically more stable than explicit methods.
Some ODE's are difficult to solve with explicit methods.
Problem with implicit methods: we need to solve (non) linear equations.

Numerical error order

The goal of a numerical solution to an ODE is of course that the error $\|y_j - y(t_j)\|$ is as small as possible. It depends (among other factors) on the step size $h_j = t_{j+1} - t_j$.

A method is **consistent of order** p, when

$$\|y_{j+1} - y(t_{j+1})\| = \mathcal{O}(h^{p+1}) \qquad \text{if } \ y_j = y(t_j).$$

Here $h = t_{j+1} - t_j$ is the step size.
(Error after one step, when the values y_{j-1}, $y(t_{j-1})$ are equal.
Local truncation error)

A method is **convergent of order** p, when

$$\max_{t_j \in [t_0, t_e]} \|y_j - y(t_j)\| = \mathcal{O}(h^p), \qquad \text{where } h = \max_j (t_{j+1} - t_j).$$

h is the maximal step size.

With only weak assumptions on the ODE and the method we have

order of convergence = order of consistency

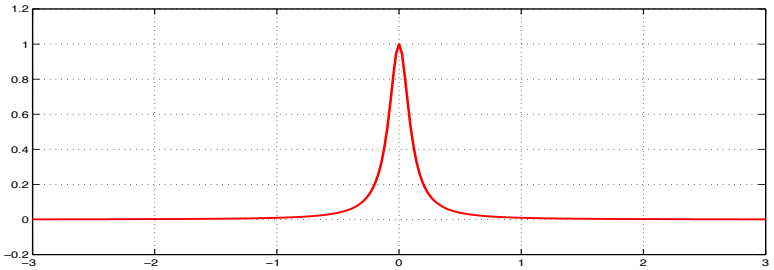
Some one-step-methods

Name	Method	Order
Euler explicit	$y_{j+1} = y_j + h_j\, f(t_j, y_j)$	1
Euler implicit	$y_{j+1} = y_j + h_j\, f(t_{j+1}, y_{j+1})$	1
Trapezoidal-rule	$y_{j+1} = y_j + h_j \dfrac{f(t_j, y_j) + f(t_{j+1}, y_{j+1})}{2}$	2
Heun	$y_{j+1} = y_j + h_j \dfrac{f(t_j, y_j) + f(t_j + h_j,\, y_j + h_j f(t_j, y_j))}{2}$	2
impl. Midpoint-rule	$y_{j+1} = y_j + h_j\, f\left(t_j + \dfrac{h_j}{2}, \dfrac{y_j + y_{j+1}}{2}\right)$	2
Collatz (mod. Euler)	$y_{j+1} = y_j + h_j\, f\left(t + \frac{h_j}{2},\, y_j + \frac{h_j}{2} f(t_j, y_j)\right)$	2

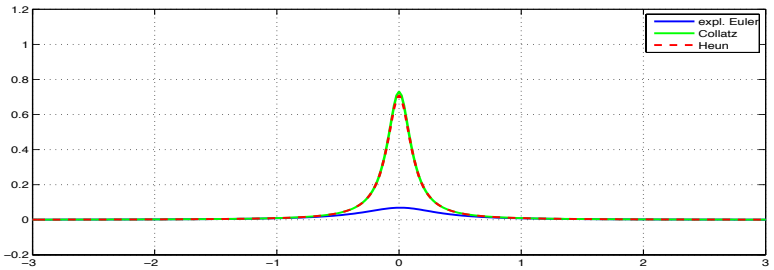
Method of Heun in cleaner notation:

$$k_1 := f(t_j, y_j), \qquad k_2 := f(t_j + h_j, y_j + h_j\, k_1), \qquad y_{j+1} = y_j + h_j \left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right).$$

Example: Numerical solutions to the ODE $y'(t) = -2\,a\,t\,y(t)^2$.
Exact solutions have the form $y(t) = 1/(c + a\,t^2),\ \ c \in \mathbb{R}$.
Here is the solution for $a = 100$, $c = 1$:

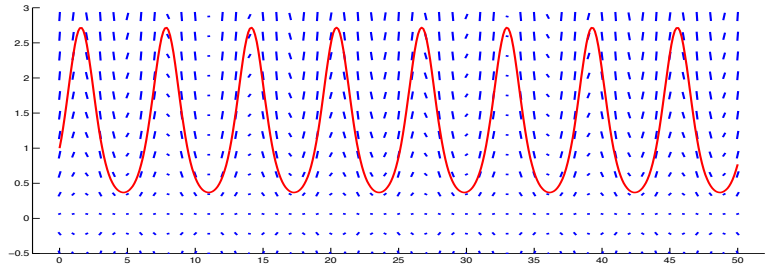


Here are the results with different solvers and with step size $h = 0.02$:

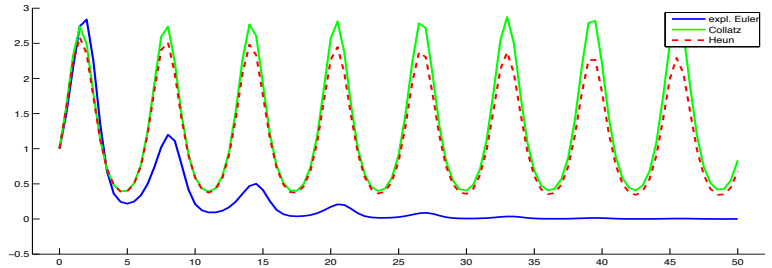


Example: Numerical solutions to the initial value problem $y'(t) = \cos(t)\, y(t),\ y(0) = 1$.

The exact solution is the periodic function $y(t) = e^{\sin(t)}$:



Here are results obtained with different solvers, where step size is large $h = 0.5$:



General s-stage Runge-Kutta-method:

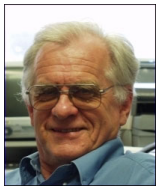
$$\begin{aligned}
k_1 &= f(t_j + c_1\, h_j,\ y_j + h_j(a_{11}\, k_1 + a_{12}\, k_2 + \ldots + a_{1s} k_s)\ ) \\
k_2 &= f(t_j + c_2\, h_j,\ y_j + h_j(a_{21}\, k_1 + a_{22}\, k_2 + \ldots + a_{2s}\, k_s)\ ) \\
&\vdots \qquad\quad \vdots \\
k_s &= f(t_j + c_s\, h_j,\ y_j + h_j(a_{s1}\, k_1 + a_{s2}\, k_2 + \ldots + a_{ss}\, k_s)\ )
\end{aligned}$$

$$y_{j+1} = y_j + h_j\,(b_1\, k_1 + b_2\, k_2 + \ldots + b_s\, k_s)$$

Butcher-table:

c_1	a_{11}	a_{12}	$\ldots$	a_{1s}
c_2	$\vdots$			$\vdots$
$\vdots$	$\vdots$			$\vdots$
c_s	a_{s1}	a_{s2}	$\ldots$	a_{ss}
	b_1	b_2	$\ldots$	b_s

Methods are explicit, when $[a_{ij}]$ is a strict lower triangular matrix.



John C. Butcher (1933-),
Auckland, Neuseeland.
Table in article of 1963

Classical Runge-Kutta-methods (order 4):

$$
\begin{aligned}
k_1 &= f(t_j,\ y_j) \\
k_2 &= f(t_j + (1/2)h_j,\ y_j + h_j\,(1/2)\,k_1) \\
k_3 &= f(t_j + (1/2)h_j,\ y_j + h_j\,(1/2)\,k_2) \\
k_4 &= f(t_j + h_j,\ y_j + h_j\,k_3)
\end{aligned}
$$

$$
y_{j+1} = y_j + h_j\left(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}\right).
$$

Butcher-Tabelle:

0				
1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	1/3	1/3	1/6

C.D.T.Runge
(1856 – 1927)

W.M.Kutta
(1867 – 1944)

Maximal order of explicit Runge-Kutta-methods

The following table shows the maximal order $p(s)$

which can be realized with an explicit s-stage method.

Stage s	1	2	3	4	5	6	7	8	9	≥ 9
Order $p(s)$	1	2	3	4	4	5	6	6	7	$< s-2$

A 2-stage implicit Runge-Kutta-method of order 4 (Gauss-method)

$$
\begin{aligned}
k_1 &= f(t_j + c_1\,h_j,\ y_j + h_j(a_{11}\,k_1 + a_{12}\,k_2)) \\
k_2 &= f(t_j + c_2\,h_j,\ y_j + h_j(a_{21}\,k_1 + a_{22}\,k_2))
\end{aligned}
$$

$$
y_{j+1} = y_j + h_j(b_1\,k_1 + b_2\,k_2)
$$

with Butcher-table:

c_1	a_{11}	a_{12}	$1/2 - 1/(2\sqrt{3})$	$1/4$	$(1/4) - 1/(2\sqrt{3})$
c_2	a_{21}	a_{22}	$1/2 + 1/(2\sqrt{3})$	$(1/4) + 1/(2\sqrt{3})$	$1/4$
	b_1	b_2		$1/2$	$1/2$

Main idea of Runge-Kutta-methods

The solution to the ODE $y'(t) = f(t, y(t))$ is

$$
y(t_{j+1}) = y(t_j) + \int_{t_j}^{t_{j+1}} f(\tau, y(\tau))\,d\tau.
$$

Let $h_j = t_{j+1} - t_j$. To approximate the integral choose a quadrature formula with interpolation points τ_i and weights b_i:

$$
y(t_{j+1}) \approx y(t_j) + h_j(\underbrace{b_1\,f(\tau_1, y(\tau_1)) + b_2\,f(\tau_2, y(\tau_2)) + \ldots + b_s\,f(\tau_s, y(\tau_s))}_{\text{weighted sum of slopes}})
$$

Unfortunately the interpolation points $f(\tau_i, y(\tau_i))$ are unknown.
Find good approximations to the slopes

$$
k_i \approx f(\tau_i, y(\tau_i))
$$

and substitute:

$$
y(t_{j+1}) \approx y(t_j) + h_j(\underbrace{b_1\,k_1 + b_2\,k_2 + \ldots + b_s\,k_s}_{\text{weighted sum of slopes}})
$$

Following this approach we get:

Midpoint-rule $\Rightarrow$ Collatz
Trapezoidal-rule $\Rightarrow$ Heun
Simpson-rule $\Rightarrow$ Classical Runge-Kutta-method

Find the order of a method with Taylor-series

Initial value problem: $y'(t) = f(t, y(t)) \qquad y(t_0) = y_0$
Result of a numerical one-step method after the first step:

$$y_1 = y_1(h) = \psi(f, t_0, y_0, h)$$

Local truncation error

$$\|y(t_0 + h) - y_1(h)\|$$

Example: Collatz-methods.
Collatz-methods are of the form

$$y_1(h) = y_0 + h\, f\left(t_0 + \frac{h}{2},\ y_0 + \frac{h}{2} f(t_0, y_0)\right).$$

With the derivatives at the point $h = 0$,

$$y_1'(0) = f(t_0, y_0), \qquad y_1''(0) = \frac{\partial f}{\partial t}(t_0, y_0) + \frac{\partial f}{\partial y}(t_0, y_0)\ f(t_0, y_0),$$

a Taylor-series gives

$$y_1(h) \ = \ y_0 + f(t_0, y_0)\, h + \left[\frac{\partial f}{\partial t}(t_0, y_0) + \frac{\partial f}{\partial y}(t_0, y_0)\ f(t_0, y_0)\right](h^2/2) + \mathcal{O}(h^3)$$

With $y'(t) = f(t, y(t))$, $y''(t) = \frac{d}{dt} f(t, y(t)) = \frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t))\, y'(t)$
this is the same as the Taylor-series for the exact solution:

$$y(t_0 + h) = y_0 + f(t_0, y_0)\, h + \left[\frac{\partial f}{\partial t}(t_0, y_0) + \frac{\partial f}{\partial y}(t_0, y_0)\ f(t_0, y_0)\right](h^2/2) + \mathcal{O}(h^3)$$

Then $\|y_1(h) - y(t_0 + h)\| = \mathcal{O}(h^3)$. $\Rightarrow$ So the order of consistency is at least 2.

Embedded Runge-Kutta-methods. Examples

1) Method of Bogacki/Shampine (Matlab-Solver `ode23`)

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{3}{4}$	0	$\frac{3}{4}$		
1	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	
	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{4}{9}$	0
	$\frac{7}{24}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{8}$

order 2
order 3

2) Method of Dormand/Prince (Matlab-Solver `ode45`, Python-solver `dopri5`)

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

order 4
order 5

Embedded Runge-Kutta-methods

Goal: Reduce numerical cost with step size control.
Increase/decrease the step size h_j in every step,
such that a given accuracy is maintained.

Given: Two Runge-Kutta-methods with the same coefficients a_{ij}, c_i,
but with different weights b_i and different orders.

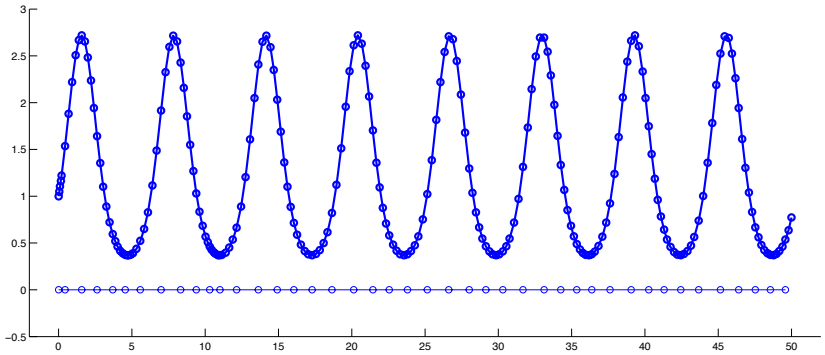
methods of order p: $\qquad y_{j+1} = y_j + h_j(b_1 k_1 + b_2 k_2 + \ldots + b_s k_s)$
methods of order $p+1$: $\qquad \widehat{y}_{j+1} = y_j + h_j(\widehat{b}_1 k_1 + \widehat{b}_2 k_2 + \ldots + \widehat{b}_s k_s)$

$\widehat{y}_{j+1}$ approximates the exact solution more accurately than y_{j+1}.
The number $\|y_{j+1} - \widehat{y}_{j+1}\|$ is an estimate for the difference of y_{j+1} to the exact solution.
Choose a tolerance tol and determine in every step the step size h_j such that

$$err := \frac{\|y_{j+1} - \widehat{y}_{j+1}\|}{h_j(1 + \|y_j\|)\, tol} \approx 1$$

Other (more complicated) error functions err can be found in the literature.

Example: Solution to the initial value problem $y'(t) = \cos(t)\, y(t)$, $y(0) = 1$ with `ode45`



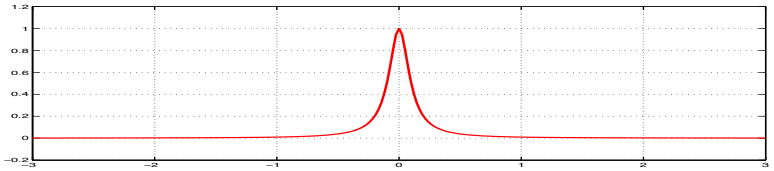
Every fifth interpolation point t_j is plotted on the zero line.

Matlab-Code:

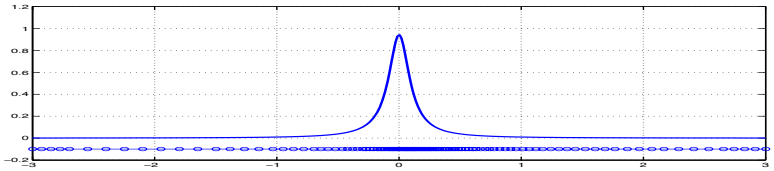
```
[T,Y] = ode45(@(t,y)(cos(t).*y),[0,50],1);
clf
hold on
axis([-2,52, -.5,3])
plot(T,Y,'-o','linewidth',2)
plot(T(1:5:length(T)),0*(1:5:length(T)),'-o')
```

Example: Solutions to the ODE $y'(t) = -2\,a\,t\,y(t)^2$ with Matlab.

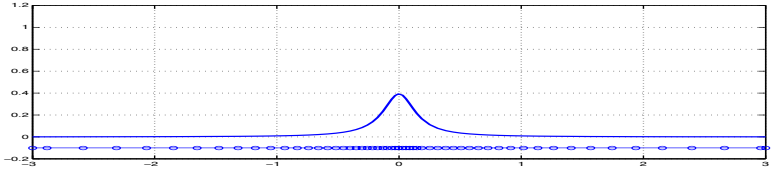
Exact solutions have the form $y(t) = 1/(c + a\,t^2),\ \ c \in \mathbb{R}$.
Plotted is the solution for $a = 100$, $c = 1$:



Numerical solution with `ode45`:



Numerical solution with `ode23`:



Easiest Adams-Bashforth-method

The interpolation polynomial through the points

$$(t_j, f(t_j, y_j)), \quad (t_{j-1}, f(t_{j-1}, y_{j-1})), \qquad t_j - t_{j-1} = h$$

is a line

$$P(t) = f(t_{j-1}, y_{j-1}) + \frac{f(t_j, y_j) - f(t_{j-1}, y_{j-1})}{h}(t - t_{j-1}).$$

Integration gives

$$y_{j+1} = y_j + \int_{t_j}^{t_{j+1}} P(\tau)\,d\tau = y_j + h\left(\frac{3}{2}f(t_j, y_j) - \frac{1}{2}f(t_{j-1}, y_{j-1})\right).$$

Multi-step-methods of Adams-Bashforth

Let $y_j, y_{j-1}, y_{j-2}, \ldots, y_{j-p+1}$ be already computed approximations of the solution to the initial value problem, with constant step size h=$const$.

Let P be the interpolation polynomial through the points

$$(t_i, f(t_i, y_i)), \qquad i = j, j-1, j-2, \ldots j-p+1.$$

Approximate

$$y(t_{j+1}) \;=\; y_j + \int_{t_j}^{t_{j+1}} f(\tau, y(\tau))\,d\tau$$

by

$$y_{j+1} \;=\; y_j + \int_{t_j}^{t_{j+1}} P(\tau)\,d\tau.$$

The so constructed methods have order p.

Note: Multi-step-methods needs to be started with one-step-methods, to get the values for the first time points.

Programming assignment 2:

Consider two masses m_1, m_2 which are at time t in the positions $\vec{x}_1(t), \vec{x}_2(t)$. According to Newton's law of gravitation the mass m_2 attracts the mass m_1 with the force

$$\vec{F}_1(\vec{x}_1(t), \vec{x}_2(t)) = \frac{\gamma\, m_1\, m_2}{\|\vec{x}_2(t) - \vec{x}_1(t)\|_2^3}(\vec{x}_2(t) - \vec{x}_1(t)),$$

where γ is the gravitational constant. The attraction force of m_1 on m_2 is $\vec{F}_2 = -\vec{F}_1$. Let $\vec{v}_k(t) = \vec{x}_k'(t)$ the velocity of the mass k at time t. Then

$$m_k\,\vec{v}_k'(t) = \vec{F}_k(t), \quad k = 1, 2 \qquad \text{(force = mass times acceleration)}.$$

The motion of the masses is described by the system of ordinary differential equations

$$\underbrace{\frac{d}{dt}\begin{bmatrix}\vec{x}_1(t)\\ \vec{x}_2(t)\\ \vec{v}_1(t)\\ \vec{v}_2(t)\end{bmatrix}}_{=:\vec{y}(t)} = \underbrace{\begin{bmatrix}\vec{v}_1(t)\\ \vec{v}_2(t)\\ \vec{F}_1(\vec{x}_1(t), \vec{x}_2(t))/m_1\\ \vec{F}_2(\vec{x}_1(t), \vec{x}_2(t))/m_2\end{bmatrix}}_{\vec{f}(\vec{y}(t))} \qquad (*)$$

Write a Matlab or Python function `twomasses(m1,m2,x1,x2,p,h)` which solves the system $(*)$ using a classical Runge-Kutta method and generate an animation showing the motion of the masses. Assume all vectors are 2-dimensional (plane motion). The parameter `h` is the time step size, and the parameter `p` determines the initial velocities:

$$\vec{v}_1(t_0) = \begin{bmatrix}0\\ p/m_1\end{bmatrix}, \quad \vec{v}_2(t_0) = \begin{bmatrix}0\\ -p/m_2\end{bmatrix}$$

Assume the gravitational constant is $\gamma = 1$.

Hint: First write a function `y_new=ruku_step(f,y,h)` to compute one Runge-Kutta step with step size h for a differential equation with arbitrary right side f (not explicitly depending on time).

The function f in our example does not explicitly depend on the time (i. e. we have $f(y)$ and not $f(t, y)$).

Then the classical Runge-Kutta-method has the form:

$$\begin{aligned} k_1 &= f(y_j) \\ k_2 &= f(y_j + h_j\,(1/2)\,k_1) \\ k_3 &= f(y_j + h_j\,(1/2\,)k_2) \\ k_4 &= f(y_j + h_j\,k_3) \\ y_{j+1} &= y_j + h_j\left(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}\right). \end{aligned}$$

Hints and suggestions for the programming assignment (Matlab).

Write three functions and save them in one file:

main function: `twomasses(M1,M2,x1,x2,p,h)`

sub-functions: `y_new=ruku_step(f,y,h)`, `f=gravi(y)`

`gravi(y)` is the right side of the ODE. Call the function `ruku_step` in `twomasses` with `gravi(y)` as parameter: `y=ruku_step(@gravi,y,h)`

Then plot the positions of y.

Note: In every plot of the animation the entire plot needs to be rebuild (with clf, hold on, axis ...).

End the plot of each figure with the command `drawnow`.

Declare masses and gravitational constant as global variables.

Possible implementation of the function `twomasses`:

```
function twomasses(M1,M2,x1,x2,p,h)

global gamma
global m1
global m2

m2 = M2;
m1 = M1;
gamma = 1;

v1 = [0; p/m1];
v2 = [0; -p/m2];
y = [x1; x2; v1; v2];

   Implement for loop to plot animation

end
```

The command lines

```
global gamma
global m1
global m2
```

also need to be in the function gravi.

Stability and Stiffness

Test problem

Let $\lambda = \alpha + i\omega \in \mathbb{C}$ be a given number.
Consider the linear ODE (test problem)

$$y'(t) = \lambda\, y(t)$$

with the exact solution

$$y(t) = y(0)\, e^{\lambda t}, \qquad y(0) \in \mathbb{C} \text{ arbitrary.}$$

The absolute value of the solution is

$$|y(t)| = |y(0)|\, e^{\mathrm{Re}(\lambda)\, t} = |y(0)|\, e^{\alpha t}$$

From which we get

1. When $\alpha = \mathrm{Re}(\lambda) < 0$, then $\lim_{t\to\infty} y(t) = 0$.
2. When $\alpha = \mathrm{Re}(\lambda) > 0$, then $\lim_{t\to\infty} |y(t)| = \infty$.
3. When $\mathrm{Re}(\lambda) = 0$, then $\lambda = i\omega$, and $|y(t)| = const = |y(0)|$.

A numerical solution to the test problem should show the same qualitative behavior, to be useful.
We will see that this is true in general only if the step size satisfies conditions.

Note: In case 1 we say 0 is a stable equilibrium point of the ODE.

Solution of the test problem $y'(t) = \lambda\, y(t)$ with Euler implicit

Applying the implicit Euler-method to the test problem gives the sequence

$$y_{j+1} = y_j + h\,\lambda\, y_{j+1} \qquad \Rightarrow \qquad y_{j+1} = \frac{1}{1 - h\,\lambda}\, y_j.$$

I. e.:

$$y_j = \left(\frac{1}{1 - h\,\lambda}\right)^j y(0)$$

From this it follows that:

1. When $|1 - h\,\lambda|^{-1} < 1$, then $\lim_{j\to\infty} y_j = 0$
2. When $|1 - h\,\lambda|^{-1} > 1$, then $\lim_{j\to\infty} |y_j| = \infty$
3. When $|1 - h\,\lambda|^{-1} = 1$, then $|y_j| = const = |y(0)|$.

The qualitative behavior of the numerical solution does not depend on $\mathrm{Re}(\lambda)$ but it depends on $|1 - h\,\lambda|^{-1}$.

We consider again the easiest case $\lambda = \alpha < 0$. Then $\lim_{t\to\infty} y(t) = \lim_{t\to\infty} |y(0)|\, e^{\lambda t} = 0$.

Now the numerical solution also converges to 0, for **any** step size, because

$$\lambda < 0 \quad \Rightarrow \quad 1 - h\lambda > 1 \quad \Rightarrow \quad |1 - h\lambda|^{-1} < 1.$$

Solution to the test problem $y'(t) = \lambda\, y(t)$ with Euler explicit

Applying the explicit Euler-method to the test problem gives the sequence

$$y_{j+1} = y_j + h\,\lambda y_j = (1 + h\,\lambda)\, y_j, \qquad y_0 = y(0).$$

I. e.:

$$y_j = (1 + h\,\lambda)^j\, y(0)$$

From this it follows that:

1. When $|1 + h\,\lambda| < 1$, then $\lim_{j\to\infty} y_j = 0$
2. When $|1 + h\,\lambda| > 1$, then $\lim_{j\to\infty} |y_j| = \infty$
3. When $|1 + h\,\lambda| = 1$, then $|y_j| = const = |y(0)|$.

I. e., the qualitative behavior of the numerical solution does not depend on $\mathrm{Re}(\lambda)$ but it depends on $|1 + h\,\lambda|$.

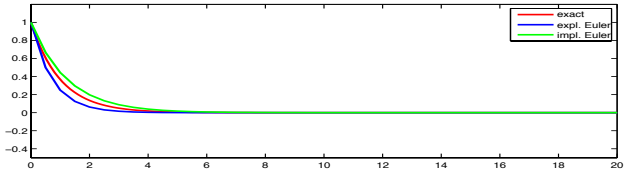
Consider the easiest case $\lambda = \alpha < 0$. Then $\lim_{t\to\infty} y(t) = \lim_{t\to\infty} |y(0)|\, e^{\lambda t} = 0$.
But for the numerical solution we have

$$\lim_{j\to\infty} y_j = 0 \quad \Leftrightarrow \quad |1 + h\,\lambda| < 1 \quad \Leftrightarrow \quad -1 < 1 + h\,\lambda = 1 - h\,|\lambda| \quad \Leftrightarrow \quad h < 2/|\lambda|$$

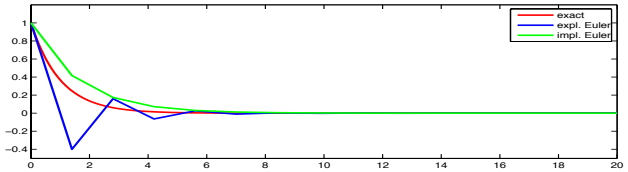
When $h \geq 2/|\lambda|$ then the numerical solution does not reflect the long term behavior of the exact solution. In addition: the numerical solution oscillates when $h > 1/|\lambda|$ (because then $1 + h\,\lambda < 0$), but the exact solution does not oscillate.

Comparison of explicit/implicit Euler for the test ODE $y'(t) = -y(t)$

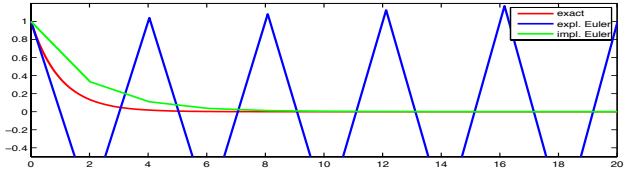
step size $h = 0.5$:



step size $h = 1.4$:



step size $h = 2.02$:



Stability regions of explicit/implicit Euler-methods

We saw that

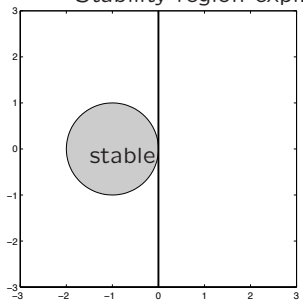
for explicit Euler: $\lim_{j\to\infty} y_j = 0 \quad \Leftrightarrow \quad |1 + h\,\lambda| < 1, \qquad (*)$

for implicite Euler: $\lim_{j\to\infty} y_j = 0 \quad \Leftrightarrow \quad |1 - h\,\lambda|^{-1} < 1. \qquad (**)$

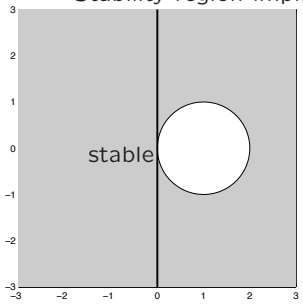
These conditions van be represented graphically:

Let $z = h\,\lambda \in \mathbb{C}$. Then the inequality $(*)$ states that z is inside the circle with radius 1. The inequality $(**)$ states that z is outside the circle with radius 1. The regions which satisfies the inequalities $(*)$ and $(**)$ resp., are called stability regions of these methods.

Stability region expl. Euler



Stability region impl. Euler



To do: Add

1. Stability function and stability regions of Runge-Kutta-methods
2. A-stability. Implicit midpoint-rule and trapezoidal-rule are A-stabil.
3. Generalization to vector valued linear ODEs. Relation to eigenvalues.