Numerical Methods for the Solution of Differential Equations

Markus Grasmair

Vienna, winter term 2011–2012

Analytical Solutions of Ordinary Differential Equations

1. Find the general solution of the differential equation
$$\dot{y} = y^2 \,.$$
In addition, find particular solutions for the initial conditions $y(1) = 1$, $y(1) = -1$, and $y(1) = 0$.

2. Find the general solution of the differential equation
$$(t^2 + 1)\dot{y} + ty = \frac{1}{2} \,.$$

3. Find the general solution of the differential equation
$$(3t - y)\dot{y} + t = 3y \,.$$
NB: This equation is of homogeneous type.

Numerical Solution of ODEs

4. Implement in Matlab or Octave the explicit Euler method, the midpoint method, and Heun's method for the solution of an ODE (or a system of ODEs) of the form
$$\dot{y}(t) = f\big(t, y(t)\big) \,, \qquad y(t_0) = y_0 \,, \tag{1}$$
up to some given time $T > t_0$.

   Test your code on the system of ODEs
$$\dot{y}_1 = -y_2 \,, \qquad \dot{y}_2 = y_1 \,, \qquad y_1(0) = 1 \,, \qquad y_2(0) = 0 \,, \tag{2}$$

with final time $T = 2\pi$, and compute the approximation error err_j (the norm of the difference between the solution of the ODE and its approximation) for step sizes $2\pi/2^j$, $j = 1, \dots, 10$ (the solution is: $y_1(t) = \cos(t)$, $y_2(t) = \sin(t)$).

A numerical approximation of the order of the methods (the *numerical order*) can be found by observing the ratios

$$-\frac{\ln(\mathrm{err}_{j+1} \,/\, \mathrm{err}_j)}{\ln(2)}$$

for large j. Explain why, and determine the numerical order of the methods.

5. Use your code to solve the ODE

$$\dot{y} = 3y^{2/3}\,, \qquad\qquad y(-1) = 1\,,$$

and determine the numerical order of the methods at the final times $T = -0.5$, $T = 0$, and $T = 1$. Explain the result!

6. Implement in MATLAB or OCTAVE Taylor's method of second order for the solution of an ODE of the form (1). Test your implementation on the ODE

$$\dot{y}(t) = y(t)\cos(t)\,, \qquad\qquad y(0) = 1\,.$$

Multi-step Methods

7. Write a MATLAB or OCTAVE implementation of the Adams–Bashforth method of third order and test your code on the system of ODEs in (2).

8. Write a MATLAB or OCTAVE implementation of a predictor–corrector method based on the third order explicit and implicit Adams methods and test your code on the system of ODEs in (2).

Stiff ODEs

In the next exercises, we will mainly consider the mass–spring system discussed in the lecture, which can lead to a stiff ODE depending on the relative size of the input parameters. The second order system of ODEs is given by the equations

$$\begin{aligned}\ddot{y}_1 &= -\frac{k_1}{m_1}y_1 + \frac{k_2}{m_1}(y_2 - y_1)\,,\\ \ddot{y}_2 &= -\frac{k_2}{m_2}(y_2 - y_1)\,,\end{aligned}$$

with initial conditions

$$\begin{aligned}y_1(0) &= a\,, \quad \dot{y}_1(0) = b\,,\\ y_2(0) &= c\,, \quad \dot{y}_2(0) = d\,.\end{aligned}$$

Here, y_1 and y_2 describe the displacement of the masses m_1 and m_2 from the equilibrium position, and k_1 and k_2 the spring constants.

9. Rewrite the second order mass–spring system as a first order system by introducing auxiliary variables for $\dot{y}_1$ and $\dot{y}_2$.

10. Write a MATLAB or OCTAVE implementation of the classical Runge–Kutta method and use it for computing numerical solutions of the mass–spring system with the following parameters:

    a) *Non-stiff setting:*

    $$\begin{aligned} k_1 &= 100, & k_2 &= 200, & m_1 &= 10, & m_2 &= 5, \\ a &= 0, & b &= 1, & c &= 0, & d &= 1\,. \end{aligned}$$

    b) *Stiff setting:*

    $$\begin{aligned} k_1 &= 100, & k_2 &= 2000, & m_1 &= 10, & m_2 &= 0.1, \\ a &= 0, & b &= 1, & c &= 0, & d &= 1\,. \end{aligned}$$

    Solve the systems for different numbers of steps and compare the results (for the stiff setting, the comparison between 1000 and 1005 steps can be interesting).

    In addition, test your implementation on the linear ODE

    $$\dot{y} = -200\big(y - \sin(t)\big) + \cos(t)\,. \tag{3}$$

11. The *implicit Euler method* for the solution of a linear ODE of the form

    $$\dot{y} = Ay\,, \qquad\qquad y(0) = y_0\,,$$

    is defined by the iteration

    $$y_{k+1} = y_k + hAy_{k+1}\,.$$

    Write a MATLAB or OCTAVE implementation and use it for the numerical solution of the mass–spring system in the stiff setting.

12. Write a MATLAB or OCTAVE implementation of the trapezoidal rule (second order Adams–Moulton method) for *linear* ODEs and use it for the numerical solution of the mass–spring system in the stiff setting.

13. Modify your implementations of the implicit Euler method and the trapezoidal rule in such a way that they can also be used for linear (inhomogeneous and non-autonomous) ODEs of the form

    $$\dot{y} = g(t) + A(t)y\,,$$

    where $g\colon \mathbb{R} \to \mathbb{R}^d$ and $A\colon \mathbb{R} \to \mathbb{R}^{d\times d}$ are given functions. Test your implementations on the ODE (3) and compare the results for different initial conditions $y(0) \in \mathbb{R}$ and different step sizes.

A Non-linear Stiff ODE

Next we consider the *non-linear* system of ODEs

$$\begin{aligned} m\ddot{y}_1 &= -k\left(\|y\| - l\right)\frac{y_1}{\|y\|} - d\langle y, \dot{y}\rangle\frac{y_1}{\|y\|^2}\,, \\ m\ddot{y}_2 &= -g - k\left(\|y\| - l\right)\frac{y_2}{\|y\|} - d\langle y, \dot{y}\rangle\frac{y_2}{\|y\|^2} \end{aligned} \tag{4}$$

with initial conditions

$$\begin{aligned} y_1(0) &= a\,, & \dot{y}_1(0) &= c\,, \\ y_2(0) &= b\,, & \dot{y}_2(0) &= d\,, \end{aligned}$$

which approximately describes the behaviour of the damped spring pendulum depicted in Figure 1. Here g is the gravitional acceleration at the earth's surface, m is the mass of the weight attached to the spring, k is the spring constant, l is the length of the spring at its resting position, d is the damping coefficient, and y_1, y_2 denote the coordinates of the weight (the suspension point of the spring is assumed to be the origin). Moreover $\|y\| := \sqrt{y_1^2 + y_2^2}$ is the Euclidean norm of y (that is, the length of the pendulum), and $\langle y, \dot{y}\rangle = y_1\dot{y}_1 + y_2\dot{y}_2$ is the scalar product of y and $\dot{y}$ (and thus the length of the projection of $\dot{y}$ onto y). This equation is stiff, if, for instance, the spring constant is much larger then the gravitational acceleration and also the damping is strong.

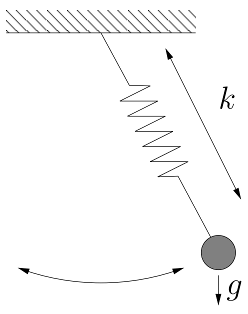

Figure 1: Spring pendulum.

14. Rewrite the system (4) as a first order system of ODEs.

15. Consider the numerical solution of the ODE (4) with the implicit Euler method. Formulate explicitly the equations that have to be solved in each time step. In addition, formulate the linear equations that appear when the non-linear equations are solved using the simplified Newton method.

16. Write MATLAB or OCTAVE implementations of the two-stage Gauß method and the two-stage Radau IIA method for the solution of *non-linear* ODEs. Use the simplified Newton method for the solution of the non-linear system that appears in each time step. Test your code on the system (4) and compare the results (and

the necessary step lengths) with the results using an explicit method of a similar order (for instance, the classical Runge–Kutta method).

Remark: The `m`-files `spring_pendulum.m` and `spring_pendulum_der.m` on the web-page of the lecture contain implementations of the functions defining the system (4) (already written as a first order system of ODEs) and its derivative. The functions take as input one scalar (the time, which is not used) and one vector with four components. The first component vector corresponds to y_1, the second to y_2, the third to $\dot{y}_1$, and the fourth to $\dot{y}_2$.

The coefficients in the implementation are $g = 1$, $m = 1$, $k = 100$, $l = 5$, $d = 1000$. Interesting initial conditions might be:

- $y_1(0) = 0$, $\dot{y}_1(0) = 0$ and any intial conditions $y_2(0) < 0$ and $\dot{y}_2(0) \in \mathbb{R}$. This setting can be good for testing purposes, as the first variable should remain zero during the whole evolution.
- $y_1(0) = 0$, $y_2(0) = -5$, $\dot{y}_1(0) = c \in \mathbb{R}$, $\dot{y}_2(0) = 0$. Then the trajectory of the solution lies almost on a circle. Depending on the size of c, this circle is closed or not.
- $y_1(0) = 6$, $y_2(0) = -6$, $\dot{y}_1(0) = \dot{y}_2(0) = 0$.