Yannis Bähni
yannis.baehni@uzh.ch

SOLUTIONS SHEET 3

Exercise 8. Disclaimer: I have used the theory given on page 125-126 in *Stoer/Bulirsch, Numerische Mathematik II, Springer Verlag, 5. Auflage.* The IVP which has to be solved, differs there with a multiplication by minus one from the one provided in the script by Dr. Sauter. I talked to him and he will have a look at it. But for now, I will use the one given in the Book. Using the IVP given in the script however, we will get $e_1(t) = \frac{t}{2}e^{-t}$.

We clearly have $f \in C^\infty([0,1]\times\mathbb{R},\mathbb{R})$, since $f(t,y) = -y$ is a polynomial in two variables. The *Euler method* is given by

$$\begin{cases} \eta(t_0;h) = y_0 \\ \eta(t+h;h) = \eta(t;h) + hf\left(t,\eta(x;h)\right) \end{cases} \tag{1}$$

Where

$$x \in \{t_0 + kh | k \in \mathbb{N}\} \qquad h \in \left\{ \frac{t-t_0}{n} \middle| n \in \mathbb{N}_{>0} \right\} \tag{2}$$

Further we know, that the Euler method is of order one. I now use the proof provided by the theorem about asymptotic expansion, since the proof gives an explicit way of finding the terms $e_k(t)$ in the expansion. We have, that

$$y(t+h) - y(t) - hf(t,y(t)) = d_2(t)h^2 + O(h^3) \tag{3}$$

As in the proof, consider the discretization method

$$\begin{cases} \hat{\eta}(t;h) := \eta(t;h) - e_1(t)h \\ \hat{\eta}(t+h;h) := \hat{\eta}(t;h) + h\hat{\Phi}\left(t,\hat{\eta}(t;h);h\right) \end{cases} \tag{4}$$

where $e_1(t)$ is a (yet) unknown function and

$$\hat{\Phi}\left(t,y;h\right) := f(t,y+e_1(t)h) - (e_1(t+h) - e_1(t)) \tag{5}$$

Taylor expansion for $h = 0$ yields

$$\begin{aligned} y(t+h) - y(t) - h\hat{\Phi}\left(t,y;h\right) &= \left(d_2(t) - \frac{\partial f}{\partial y}e_1(t) - \frac{d}{dt}e(t)\right)h^2 + O(h^3) \\ &= \left(d_2(t) + e_1(t) - \frac{d}{dt}e(t)\right)h^2 + O(h^3) \end{aligned}$$

Since we want, that the method provided by $\hat{\Phi}$ is of order two, we must solve the IVP

$$\begin{cases} e_1'(t) = d_2(t) + e_1(t) \\ e_1(0) = 0 \end{cases} \tag{6}$$

From the Taylor expansion of the exact solution we know $d_2(t) = \frac{1}{2}\left(\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial y}\right)$. Hence

Yannis Bähni
yannis.baehni@uzh.ch

(7)
$$\begin{cases} e_1'(t) = e_1(t) + \frac{e^{-t}}{2} \\ e_1(0) = 0 \end{cases}$$

This is a first order linear inhomogene differential equation. Consider the approach $e_1(t) = c(x)e^t$. Differentiation yields $e_1'(t) = c'(t)e^t + c(t)e^t = c'(t)e^t + e_1(t)$. Definite integration yields

(8)
$$e_1(t) = \left(\frac{1}{2}\int_0^t e^{-2x}dx\right)e^t = \frac{1}{4}\left(e^t - e^{-t}\right) = \frac{1}{2}\sinh(t)$$

Exercise 9. **a.** From figure 3 we can see, that for $h = 0.2$, the error first increases up to $t = 1$ and then decreases (in the plot it gets smaller but taking its modulus is characteristic). For $h = 0.4$, we have a larger error in comparison to the smaller stepsize (however this is clear). So the exact solution e^{-t} is approximated with a moderate error by the Euler method. The error first increases and then decreases (this is due to theslope of the exact solution, since the euler method is not able to hold this speed). For the second IVP it looks different: the error increases for $h = 0.2$ and $h = 0.4$ (for $h = 0.2$ clearly slower). This can also be seen on the right side of figure 4, since the numerical solutions diverge.

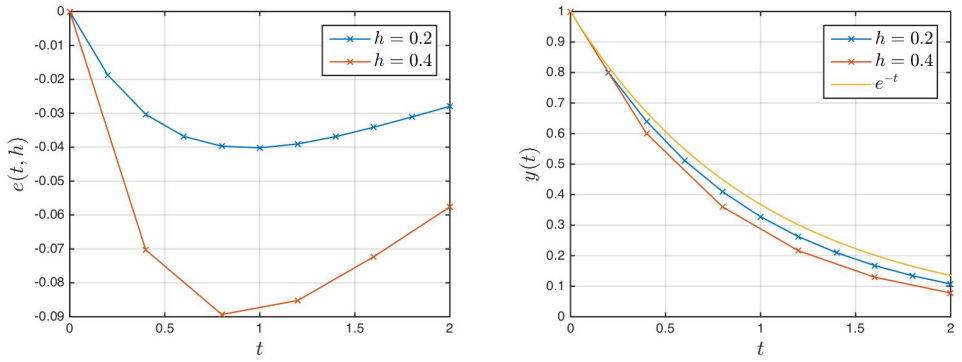

FIGURE 1. `src/ex_9_euler_1.jpg`

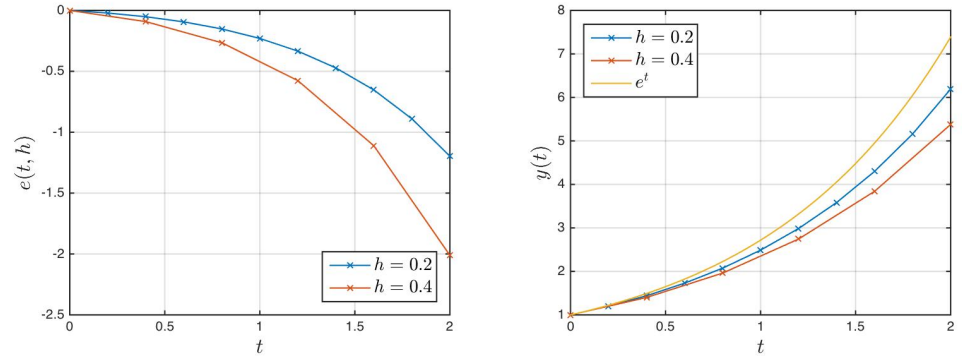

FIGURE 2. `src/ex_9_euler_2.jpg`

Yannis Bähni
yannis.baehni@uzh.ch

b. Remarkable is here, that the error behaves almost the same, as in **a.** (only the scale differs with a factor 10^{-3}). We also see, that the error for the exact solution e^t diverges.

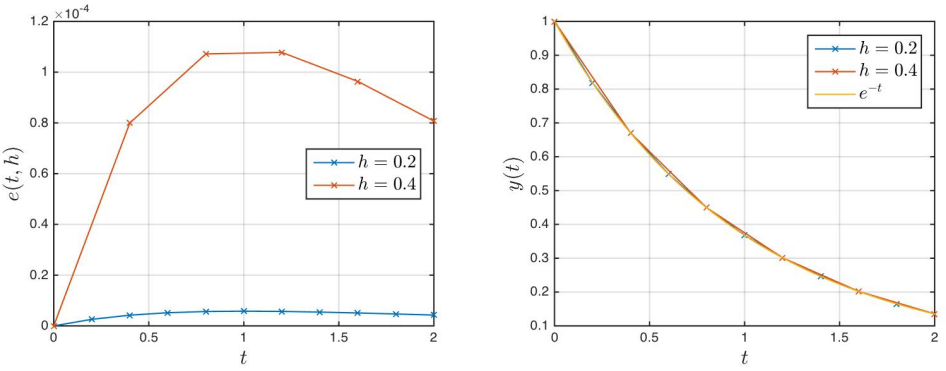

FIGURE 3. `src/ex_9_rk4_1.jpg`

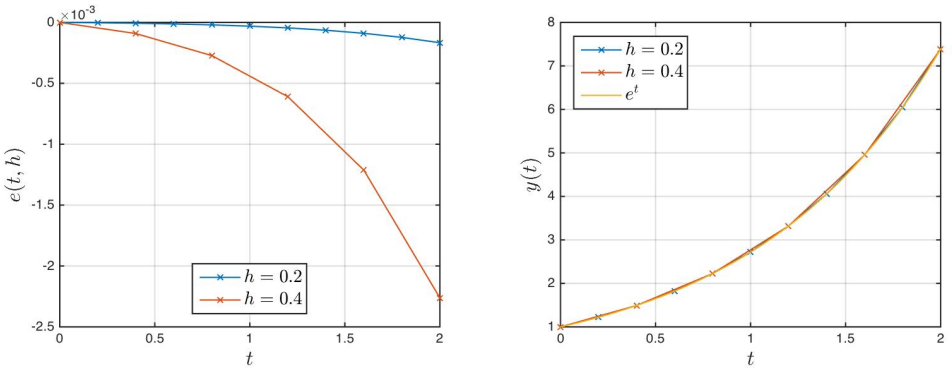

FIGURE 4. `src/ex_9_rk4_2.jpg`

Exercise 10. (a) The *global discretisation error* is given by

$$\min_j e(t_j, h) \approx -0.12 \tag{9}$$

as one can see in figure 5. From figure 6 we can tell, that for $N > 435$, the global discretisation error will be smaller than 10^{-4}.

(b) I have implemented an adaptive algorithm found in *Stoer/Bulirsch, Numerische Mathematik II* on page 133. The algorithm given during the lecture results in a monotone decreasing sequence of step-sizes, and hence results in a non-progressive solution. However, the algorithm provided in Stoer/Bulirsch does not have this issue. Given x_0, y_0 from an IVP, $m \in \mathbb{N}_{>1}$ and an initial stepsize H, do the following steps:

a. Compute discretisations $\eta(x_0 + H; H)$ and $\eta(x_0 + H; H/m)$ with a one-step method of order p, further compute h from

$$\frac{H}{h} \doteq \left(\frac{m^p}{m^p - 1} \frac{|\eta(x_0 + H; H) - \eta(x_0 + H; H/m)|}{\varepsilon} \right)^{1/(p+1)} \tag{10}$$

Yannis Bähni
yannis.baehni@uzh.ch

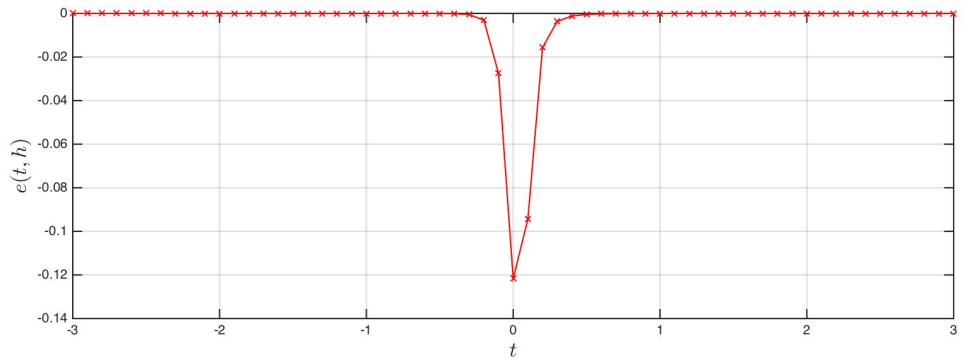

FIGURE 5. `src/ex_10_a_globerr.jpg`

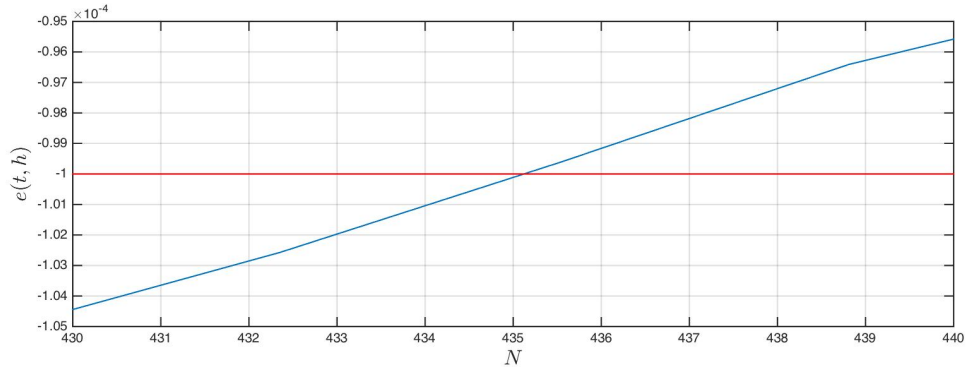

FIGURE 6. `src/ex_10_a_globerr_small.jpg`

b. Check $\frac{H}{h} \gg m$. If true, set $H = 2h$ and go to a., if false, set $H = 2h$ and go to a. with $x_0 = x_0 + H$ and $y_0 = \eta(x_0 + H; H/m)$.

c. In figure 7 we see the discretisation of the test function $y(t) = \frac{1}{1+100t^2}$ by a stepsize-controlled Runge-Kutta method of order four. In the plot we can nicely see the *adaptiveness* of the method (step-size control). We further see, that $m = 2$ performs much worser in terms of number of steps than $m = 10$, this is clear since with $m = 2$ the stepsize can grow much slower than with $m = 10$. Thus, in to answer the question, I choose $m = 10$ for the further proceeding. As one can see in figure 8, for $\varepsilon \approx 0.2 \cdot 10^{-10}$ the global discretisation error is more or less smaller than 10^{-4}. From the same figure I can tell, that the steps used are more than 110. Thus, comparing this with **a.**, this is significandly less than the number of steps attained there. Hence the step-size controlled method should be preffered in this particular example in terms of efficency. This experiment cannot be done using $m = 2$, since for the required precision ε, the method takes a long time to compute.

Yannis Bähni
yannis.baehni@uzh.ch

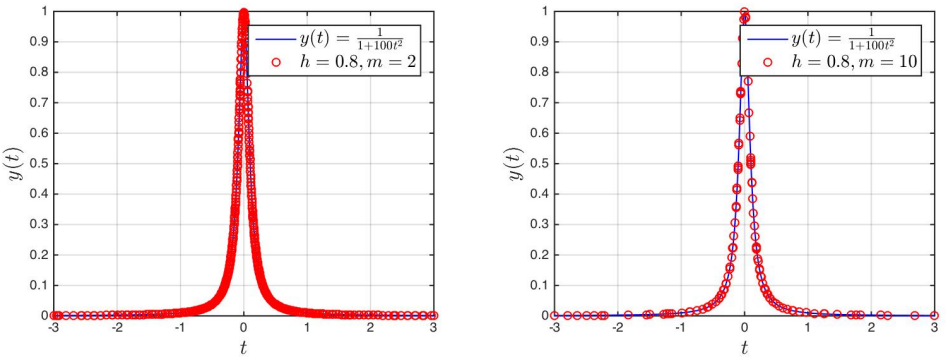

FIGURE 7. Approximation of the exact function $y(t) = \frac{1}{1+100t^2}$ on the interval $[-3, 3]$ using a step-size controlled Runge-Kutta method of order four with damp-factor 2 respectively 10 and $h = 0.8$ for $\varepsilon = 10^{-11}$.

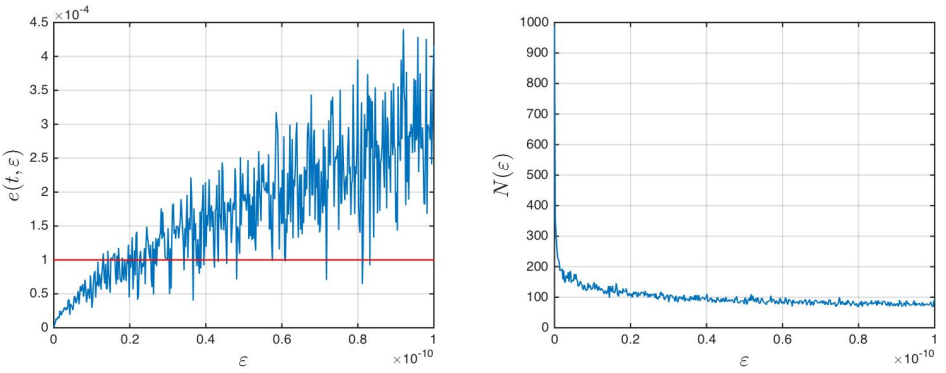

FIGURE 8. On the left: Plot of $\max_j |e(t_j, h)|$ for $h = 0.8$ and $m = 10$ against $\varepsilon \in [10^{-14}, 10^{-10}]$. On the right: The number of steps used by the step-size controlled Runge-Kutta method for precision $\varepsilon \in [10^{-14}, 10^{-10}]$.

Yannis Bähni
yannis.baehni@uzh.ch

```
function [ t,y,successfull ] = SSCRK( f,t0,tN,y0,H,m,epsilon )
%Classical Runge-Kutta method of order four with step-size control.
%Implementation of the classical Runge-Kutta method of order four with a
%step-size controller based on asymptotic expansions of the global discret-
%isation error as seen in _[1], page 130 - 133.

%Author:
%------
%Yannis B?hni - baehni.yannis@uzh.ch

%References:
%----------
%_[1]   Stoer/Bulirsch, Numerische Mathematik II, Springer Verlag, 5. Auf-
%       lage, 2005.
p = 4;
%Storage
y(1) = y0;
t(1) = t0;
successfull = 0;
rejected = 0;
%Main loop
while t(end) < tN
    %Calculating exact stepsize to reach endpoint tN
    if (t(end) + H) >= tN
        H = tN - t(end);
    end
    %Discretisation with stepsize H at t + H
    [~,EtaH] = rk4(f,t(end),t(end) + H,y(end),H);
    %Discretisation with stepsize H/m at t + H
    [~,EtaHm] = rk4(f,t(end),t(end) + H,y(end),H/m);
    %Check if factor is not too small (may result in division by zero)
    if abs(EtaH(end) - EtaHm(end)) < realmin
        factor = 1;
    else
        factor = m^p/(m^p - 1) * abs(EtaH(end) - EtaHm(end))/epsilon;
    end
    %Calculating optimised stepsize h
    h = H/factor^(1/(p+1));
    if H/h >= m + 1
        H = m * h; rejected = rejected + 1;
    else
        t(end + 1) = t(end) + H; y(end + 1) = EtaHm(end); H = m * h;
        successfull = successfull + 1;
    end
end
    disp(['Number of successfull steps: ', num2str(successfull)]);
    disp(['Number of rejected steps: ', num2str(rejected)]);
end
```

Listing 1. src/SSCRK.m