Technical University of Denmark

02685 Scientific Computing for Differential Equations
2017

Assignment 1

Authors:
Miguel Suau de Castro (s161333)
Michal Baumgartner (s161636)

March 16, 2017

Contents

1 The Test Problem and DOPRI54

In this first section we are going to implement a set of numerical methods for solving ordinary differential equations. Since the algorithms are only approximations to the real solution, we shall also test their accuracy and discuss their performance by comparing the results obtained when solving the two following initial value problems:

EQUATIONS

As a first approach, we are going to implement the Explicit Euler's method. The algorithm makes use of finite different methods to replace the derivatives in the differential equation. The independent variable is discretized and the solution is computed based on cosequtive approximations to the real function values.

TALK ABOUT STEP LENGTH

EQUATION FORWARD EULER

Instead of using the previous iterate one could also look at future values to approximate a solution. This method is called backward or implicit Euler:

EQUATION BACKWARD EULER

However, for some problems the solution of the previous equation may require the use of numerical solvers, and thus the algorithm becomes computationally more demanding than the explicit Euler's method. We shall see in the next section the advantange of using this method.

Besides, the trapezodial method can be seen as a combination of both methods:

DESCRIBE TRAPEZOIDAL

Figure **??** shows the solution of the two initial value problems given by explicit, implicit Euler and trapezoidal, along with the true solution. It is easy to see, especially in the graph on the right, that, since we base the solution at one point on previous approximations, the further the points are from the initial value the more inaccurate they become and the greater the distance to the true solution is. This distance is called global error, whilst the error made in every iteration is known as local error.

2 Question 3

2.1 Order conditions, coefficients for the error estimator and the Butcher tableau

Using the excerpt from the book provided in the lecture 10 folder we will write up the order conditions for an embedded Runge-Kutta method with 3 stages. The solution will have order 3 and the embedded method used for error estimation will have order 2.

Firstly the Butcher tableau for our ERK will have the following schema (henceforth the upper triangular shape where the a_{ij} coefficients are 0 and $c_1 = 0$):

$$
\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
c_2 & a_{21} & 0 & 0 \\
c_3 & a_{31} & a_{32} & 0 \\
\hline
x & b_1 & b_2 & b_3 \\
\widehat{x} & \widehat{b}_1 & \widehat{b}_2 & \widehat{b}_3 \\
\hline
e & d_1 & d_2 & d_3
\end{array}
$$

Table 1: Butcher tableau for ERK with 3 stages and embedded method

Order conditions (one for first order, one for second order and two for third order) derived from our Butcher tableau:

$$\mathcal{O}(h^1): \quad b^T e = 1 \quad b_1 + b_2 + b_3 = 1 \quad \tau_1 \rightarrow \tag{1a}$$

$$\mathcal{O}(h^2): \quad b^T C e = \frac{1}{2} \quad \underbrace{b_1 c_1}_{0} + b_2 c_2 + b_3 c_3 = \frac{1}{2} \quad \tau_2 \rightarrow \tag{1b}$$

$$\mathcal{O}(h^3): \quad b^T C^2 e = \frac{1}{3} \quad \underbrace{b_1 c_1^2}_{0} + b_2 c_2^2 + b_3 c_3^2 = \frac{1}{3} \quad \tau_3 \rightarrow \tag{1c}$$

$$b^T A C e = \frac{1}{6} \quad \underbrace{b_2 a_{21} c_1}_{0} + \underbrace{b_3 a_{31} c_1}_{0} + b_3 a_{32} c_2 = \frac{1}{6} \quad \tau_4 \rightarrow \tag{1d}$$

values of c_2 and c_3 will be set to $\frac{1}{4}$ and 1 respecively. This leaves us with 6 unknown variables (3 as and 3 bs) and only 4 equations so we will add the so called consistency conditions in order for the system to be solvable.

$$c_2 = a_{21} \tag{1e}$$

$$c_3 = a_{31} + a_{32} \tag{1f}$$

Using Matlab to solve the system we get the following results:

$$b_1 = -\tfrac{1}{6},\ b_2 = \tfrac{8}{9},\ b_3 = \tfrac{5}{18},\ a_{21} = \tfrac{1}{4},\ a_{31} = -\tfrac{7}{5},\ a_{32} = \tfrac{12}{5}.$$

Next we will solve the system defined for second order embedded method with one first order and one second order condition where c_2 and c_3 are known thus giving 2 equations with 3 unknowns. In order to find a solution, $\widehat{b}_2$ is set to be $\frac{1}{2}$[1].

$$\widehat{b}_1 + \widehat{b}_2 + \widehat{b}_3 = 1 \tag{2a}$$

$$\widehat{b}_2 c_2 + \widehat{b}_3 c_3 = \frac{1}{2} \tag{2b}$$

The above system yields $\widehat{b}_1 = \frac{1}{8}$ and $\widehat{b}_3 = \frac{3}{8}$. Going back to the Butcher tableau we know that last row $e = (d_1, d_2, d_3)$ is just the difference of the previous two rows by definition.

$c_1 = 0$	0	0	0
$c_2 = \frac{1}{4}$	$1/4$	0	0
$c_3 = 1$	$-7/5$	$12/5$	0
x	$-1/6$	$8/9$	$5/18$
$\widehat{x}$	$1/8$	$1/2$	$3/8$
e	$-7/24$	$7/18$	$-7/72$

Table 2: Butcher tableau with error estimators for our method

[1] According to the book excerpt given in Lecture 10 folder. Otherwise any real value < 1 could have been selected.

2.2 Testing on the test equation

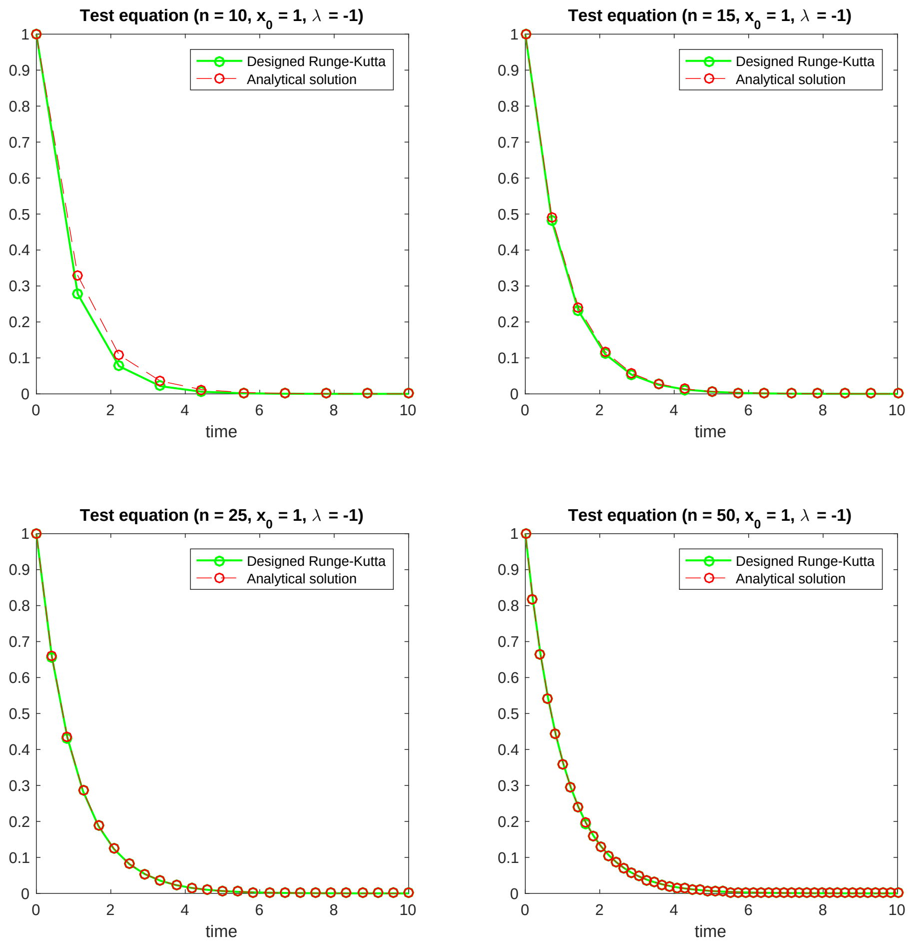

Figure 1: Comparison with the test equation for different step sizes

2.3 Verifying the order

2.4 R(λh) and stability plot

The solution to the test equation obtained by a Runge-Kutta method is defined as $x(t_n+h) = R(\lambda h)x(t_n)$ and $R(z) = 1 + zb^T(I - zA)^{-1}e$. From the Butcher tableau with error estimators for our method vector b and the A matrix are plugged in to

$R(z)$ resulting in

$$R_m(z) = 1 + z + \frac{1}{2}z^2 + \frac{3}{18}z^3$$

where $z = \lambda h$ for the third order method. The second order embedded method yields

$$R_e(z) = z + \frac{1}{2}z^2 + \frac{9}{40}z^3$$

where $z = \lambda h$. Note that $R(z)$ can be calculated with Matlab's Symbolic Toolbox

```
syms z;
R = 1 + z*b'*inv(eye(length(b)) - z*A)*ones(length(b),1);
```

then `collect(R, z)` is used display powers of z and the respective coefficients.

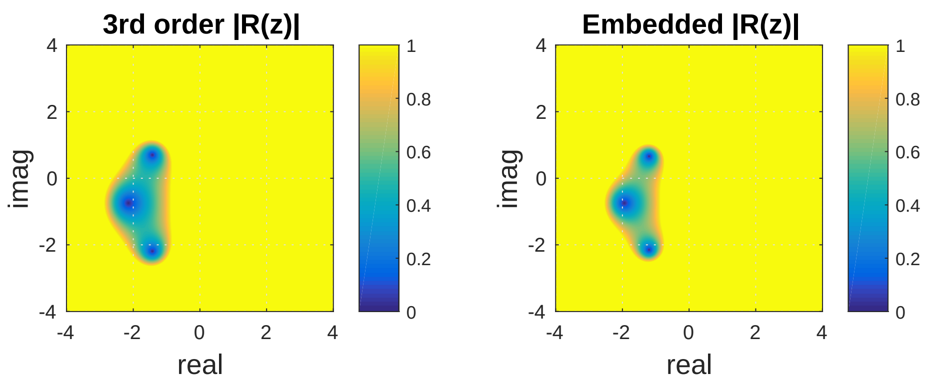

Figure 2: Stability plots of the third order ERK with second order embedded method

2.5 Testinging on the Van der Pol problem and comparison with `ode15s`

Matlab's `ode15s` was used with the default ODE-options and user defined Jacobian, the error for our method with with step size of 10^{-3} is roughly around 10^{-8} and for step size 10^{-2} it is around 10^{-5}. Even though our choice of $c_2 = 1/4$ might look strange, the method performs reasonably well.

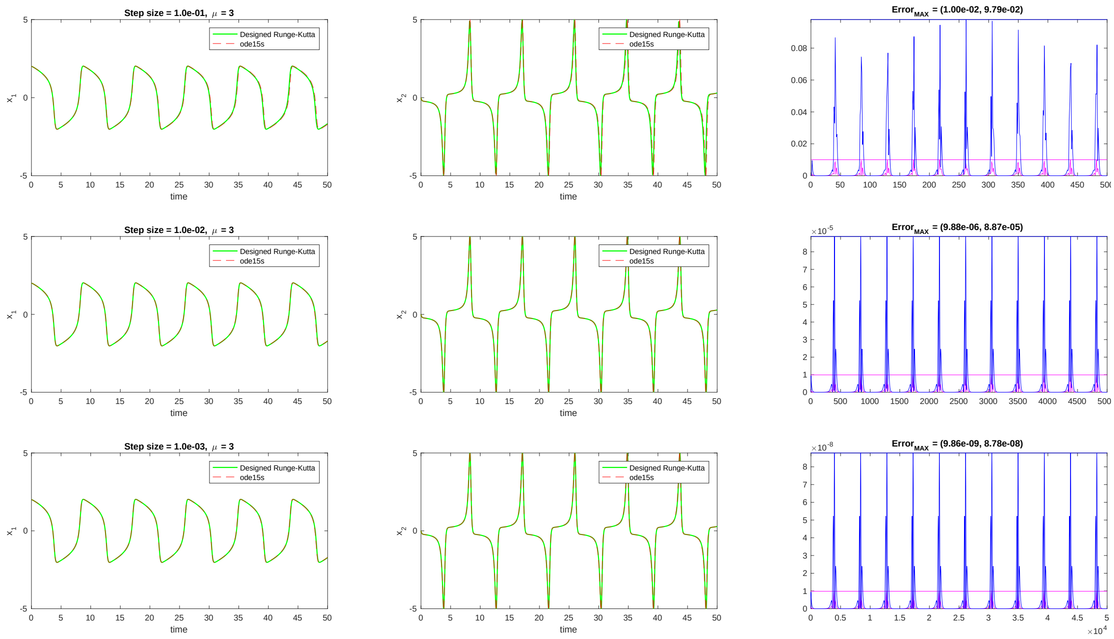

Figure 3: Comparison with `ode15s` on Van der Pol problem ($\mu = 3$). Each row depicts different step size (0.1, 0.01 and 0.001) and the maximal error from ERK is shown in the plot title as well as on a reference line (x_1 - magenta, x_2 - blue).