Prof. R. Hiptmair
G. Alberti,
F. Leonardi

AS 2015

Numerical Methods for CSE

ETH Zürich
D-MATH

Problem Sheet 14

Problem 1 Implicit Runge-Kutta method (core problem)

This problem is the analogon of Problem 1, Problem Sheet 12, for general implicit Runge-Kutta methods [1, Def. 12.3.18]. We will adapt all routines developed for the explicit method to the implicit case. This problem assumes familiarity with [1, Section 12.3], and, especially, [1, Section 12.3.3] and [1, Rem. 12.3.24].

(1a) By modifying the class `RKIntegrator`, design a header-only C++ class `implicit_RKIntegrator` which implements a general implicit RK method given through a Butcher scheme [1, Eq. (12.3.20)] to solve the autonomous initial value problem $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$, $\mathbf{y}(0) = \mathbf{y}_0$. The stages $\mathbf{g}_i$ as introduced in [1, Eq. (12.3.22)] are to be computed with the damped Newton method (see [1, Section 2.4.4]) applied to the nonlinear system of equations satisfied by the stages (see [1, Rem. 12.3.21] and [1, Rem. 12.3.24]). Use the provided code `dampnewton.hpp`, that is a simplified version of [1, Code 2.4.50]. Note that we do not use the simplified Newton method as discussed in [1, Rem. 12.3.24].

In the code template `implicit_rkintegrator_template.hpp` you will find all the parts from `rkintegrator_template.hpp` that you should reuse. In fact, you only have to write the method `step` for the implicit RK.

(1b) Examine the code in `implicit_rk3prey.cpp`. Write down the complete Butcher scheme according to [1, Eq. (12.3.20)] for the implicit Runge-Kutta method defined there. Which method is it? Is it A-stable [1, Def. 12.3.32], L-stable [1, Def. 12.3.38]?

HINT: Scan the particular implicit Runge-Kutta single step methods presented in [1, Section 12.3].

(1c) Test your implementation `implicit_RKIntegrator` of general implicit RK SSMs with the routine provided in the file `implicit_rk3prey.cpp` and comment on the observed order of convergence.

Problem 2 Initial Value Problem With Cross Product

We consider the initial value problem

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}) := \mathbf{a} \times \mathbf{y} + c\mathbf{y} \times (\mathbf{a} \times \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0 = [1,1,1]^\top, \tag{61}$$

where $c > 0$ and $\mathbf{a} \in \mathbb{R}^3$, $\|a\|_2 = 1$.

NOTE: $\mathbf{x} \times \mathbf{y}$ denotes the cross product between the vectors $\mathbf{x}$ and $\mathbf{y}$. It is defined by

$$\mathbf{x} \times \mathbf{y} = [x_2y_3 - x_3y_2, x_3y_1 - x_1y_3, x_1y_2 - x_2y_1]^\top.$$

It satisfies $\mathbf{x} \times \mathbf{y} \perp \mathbf{x}$. In Eigen, it is available as `x.cross(y)`.

(2a) Show that $\|\mathbf{y}(t)\|_2 = \|\mathbf{y}_0\|_2$ for every solution $\mathbf{y}$ of (61).

HINT: Target the time derivative $\frac{d}{dt}\|\mathbf{y}(t)\|_2^2$ and use the product rule.

(2b) Compute the Jacobian $D\mathbf{f}(\mathbf{y})$. Compute also the spectrum $\sigma(D\mathbf{f}(\mathbf{y}))$ in the stationary state $\mathbf{y} = \mathbf{a}$, for which $\mathbf{f}(\mathbf{y}) = 0$. For simplicity, you may consider only the case $\mathbf{a} = [1,0,0]^\top$.

(2c) For $\mathbf{a} = [1,0,0]^\top$, (61) was solved with the standard MATLAB integrators `ode45` and `ode23s` up to the point $T = 10$ (default Tolerances). Explain the different dependence of the total number of steps from the parameter c observed in Figure 8.

(2d) Formulate the non-linear equation given by the implicit mid-point rule for the initial value problem (61).

(2e) Solve (61) with $\mathbf{a} = [1,0,0]^\top$, $c = 1$ up to $T = 10$. Use the implicit mid-point rule and the class developed for Problem 1 with $N = 128$ timesteps (use the template `cross_template.cpp`). Tabulate $\|\mathbf{y}_k\|_2$ for the sequence of approximate states generated by the implicit midpoint method. What do you observe?

(2f) The linear-implicit mid-point rule can be obtained by a simple linearization of the incremental equation of the implicit mid-point rule around the current solution value.

Give the defining equation of the linear-implicit mid-point rule for the general autonomous differential equation

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$$

with smooth f.

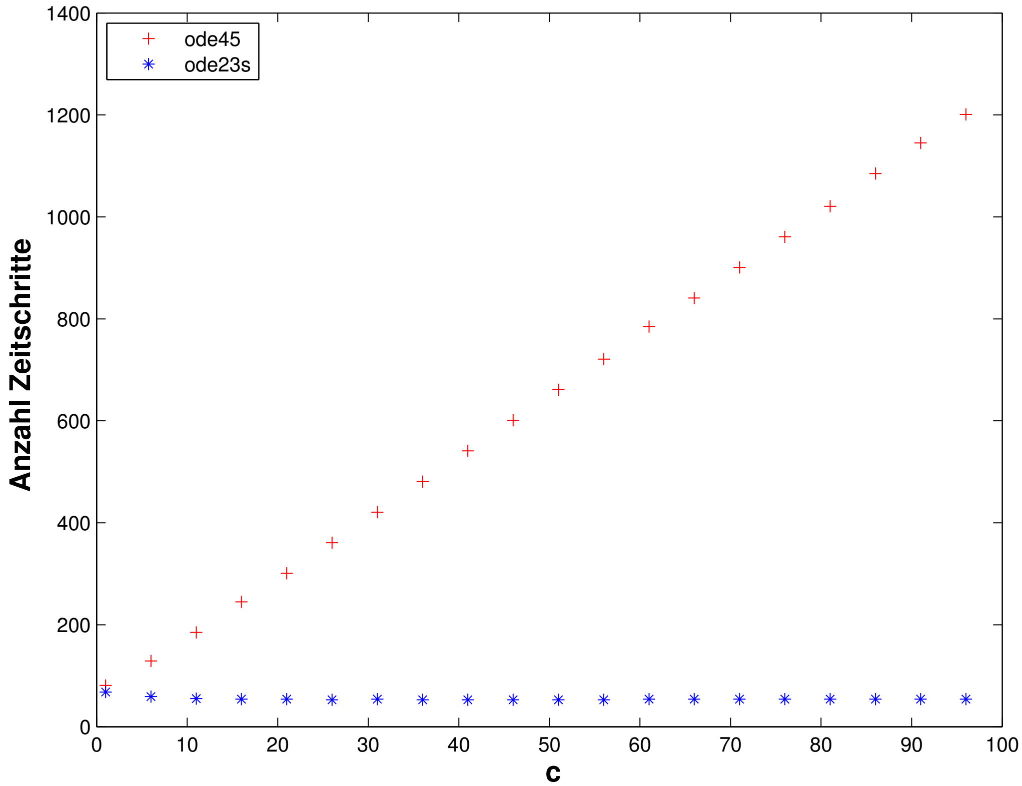

Figure 8: Subproblem (2c): number of steps used by standard MATLAB integrators in relation to the parameter c.

(2g) Implement the linear–implicit midpoint rule using the template provided in `cross_template.cpp`. Use this method to solve (61) with $\mathbf{a} = [1, 0, 0]^\top$, $c = 1$ up to $T = 10$ and $N = 128$. Tabulate $\|\mathbf{y}_k\|_2$ for the sequence of approximate states generated by the linear implicit midpoint method. What do you observe?

Problem 3 Semi-implicit Runge-Kutta SSM (core problem)

General implicit Runge-Kutta methods as introduced in [1, Section 12.3.3] entail solving systems of non-linear equations for the increments, see [1, Rem. 12.3.24]. Semi-implicit Runge-Kutta single step methods, also known as Rosenbrock-Wanner (ROW) methods [1, Eq. (12.4.6)] just require the solution of linear systems of equations. This problem deals with a concrete ROW method, its stability and aspects of implementation.

We consider the following autonomous ODE

$$\mathbf{y} = \mathbf{f}(\mathbf{y}) \tag{62}$$

and discretize it with a *semi-implicit* Runge-Kutta SSM (*Rosenbrock method*):

$$\begin{aligned}
\mathbf{W}\mathbf{k}_1 &= \mathbf{f}(\mathbf{y}_0) \\
\mathbf{W}\mathbf{k}_2 &= \mathbf{f}(\mathbf{y}_0 + \frac{1}{2}h\mathbf{k}_1) - ah\mathbf{J}\mathbf{k}_1 \\
\mathbf{y}_1 &= \mathbf{y}_0 + h\mathbf{k}_2
\end{aligned} \tag{63}$$

where

$$\begin{aligned}
\mathbf{J} &= D\mathbf{f}(\mathbf{y}_0) \\
\mathbf{W} &= \mathbf{I} - ah\mathbf{J} \\
a &= \frac{1}{2+\sqrt{2}}.
\end{aligned}$$

(3a) Compute the stability function S of the Rosenbrock method (63), that is, compute the (rational) function $S(z)$, such that

$$y_1 = S(z)y_0, \quad z := h\lambda,$$

when we apply the method to perform one step of size h, starting from y_0, of the linear scalar model ODE $\dot{y} = \lambda y, \lambda \in \mathbb{C}$.

(3b) Compute the first 4 terms of the Taylor expansion of $S(z)$ around $z = 0$. What is the maximal $q \in \mathbb{N}$ such that

$$|S(z) - \exp(z)| = O(|z|^q)$$

for $|z| \to 0$? Deduce the maximal possible order of the method (63).

HINT: The idea behind this sub-problem is elucidated in [1, Rem. 12.1.19]. Apply [1, Lemma 12.1.21].

(3c) Implement a C++ function:

```
template <class Func, class DFunc, class StateType>
std::vector<StateType> solveRosenbrock(
                    const Func & f, const DFunc & df,
                    const StateType & y0,
                    unsigned int N, double T)
```

taking as input function handles for $\mathbf{f}$ and $D\mathbf{f}$ (e.g. as lambda functions), an initial data (vector or scalar) `y0` $= \mathbf{y}(0)$, a number of steps N and a final time T. The function returns the sequence of states generated by the single step method up to $t = T$, using N equidistant steps of the Rosenbrock method (63).

HINT: See `rosenbrock_template.cpp`.

(3d) ⚂ Explore the order of the method (63) empirically by applying it to the IVP for the limit cycle [1, Ex. 12.2.5]:

$$\mathbf{f}(\mathbf{y}) := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{y} + \lambda(1 - \|\mathbf{y}\|^2)\mathbf{y}, \tag{64}$$

with $\lambda = 1$ and initial state $\mathbf{y_0} = [1, 1]^\top$ on $[0, 10]$. Use fixed timesteps of size $h = 2^{-k}, k = 4, \ldots, 10$ and compute a reference solution with $h = 2^{-12}$ step size. Monitor the maximal mesh error:

$$\max_j \|\mathbf{y}_j - \mathbf{y}(t_j)\|_2.$$

(3e) ⚃ Show that the method (63) is L-stable (cf. [1, § 12.3.37]).

HINT: To investigate the A-stability, calculate the complex norm of $S(z)$ on the imaginary axis $\operatorname{Re} z = 0$ and apply the following maximum principle for holomorphic functions:

Theorem (Maximum principle for holomorphic functions)**.** Let

$$\mathbb{C}^- := \{z \in \mathbb{C} \mid Re(z) < 0\}.$$

Let $f : D \subset \mathbb{C} \to \mathbb{C}$ be non-constant, defined on $\overline{\mathbb{C}^-}$, and analytic in $\mathbb{C}^-$. Furthermore, assume that $w := \lim_{|z| \to \infty} f(z)$ exists and $w \in \mathbb{C}$, then:

$$\forall z \in \mathbb{C}^- |f(z)| < \sup_{\tau \in \mathbb{R}} |f(i\tau)|.$$

Problem 4 Singly Diagonally Implicit Runge-Kutta Method

SDIRK-methods (**S**ingly **D**iagonally **I**mplicit **R**unge-**K**utta methods) are distinguished by Butcher schemes of the particular form

$$\begin{array}{c|c} \mathbf{c} & \mathfrak{A} \\ \hline & \mathbf{b}^T \end{array} := \begin{array}{c|cccc} c_1 & \gamma & & \cdots & 0 \\ c_2 & a_{21} & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{s,s-1} & \gamma \\ \hline & b_1 & \cdots & b_{s-1} & b_s \end{array}, \tag{65}$$

with $\gamma \neq 0$.

More concretely, in this problem the scalar linear initial value problem of second order

$$\ddot{y} + \dot{y} + y = 0, \qquad y(0) = 1, \quad \dot{y}(0) = 0 \tag{66}$$

should be solved numerically using a SDIRK-method (**S**ingly **D**iagonally **I**mplicit **R**unge-**K**utta Method). It is a Runge-Kutta method described by the Butcher scheme

$$\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline & 1/2 & 1/2 \end{array}. \tag{67}$$

(4a) ⊡ Explain the benefit of using SDIRK-SSMs compared to using Gauss-Radau RK-SSMs as introduced in [1, Ex. 12.3.44]. In what situations will this benefit matter much?

HINT: Recall that in every step of an implicit RK-SSM we have to solve a non-linear system of equations for the increments, see [1, Rem. 12.3.24].

(4b) ⊡ State the equations for the increments $\mathbf{k}_1$ and $\mathbf{k}_2$ of the Runge-Kutta method (67) applied to the initial value problem corresponding to the differential equation $\dot{\boldsymbol{y}} = \mathbf{f}(t, \boldsymbol{y})$.

(4c) ⊡ Show that, the stability function $S(z)$ of the SDIRK-method (67) is given by

$$S(z) = \frac{1 + z(1-2\gamma) + z^2(1/2 - 2\gamma + \gamma^2)}{(1-\gamma z)^2}$$

and plot the stability domain using the template `stabdomSDIRK.m`.

For $\gamma = 1$ is this method:

- A–stable?
- L–stable?

HINT: Use the same theorem as in the previous exercise.

(4d) ⊡ Formulate (66) as an initial value problem for a linear first order system for the function $\boldsymbol{z}(t) = (y(t), \dot{y}(t))^\top$.

(4e) ⊡ Implement a C++-function

```
template <class StateType>
StateType sdirtkStep(const StateType & z0, double h,
    double gamma);
```

that realizes the numerical evolution of one step of the method (67) for the differential equation determined in subsubsection (4d) starting from the value `z0` and returning the value of the next step of size `h`.

HINT: See `sdirk_template.cpp`.

(4f) ⊡ Use your C++ code to conduct a numerical experiment, which gives an indication of the order of the method (with $\gamma = \frac{3+\sqrt{3}}{6}$) for the initial value problem from subsubsection (4d). Choose $\boldsymbol{y}_0 = [1, 0]^\top$ as initial value, `T=10` as end time and `N=20,40,80,...,10240` as steps.

Issue date: 17.12.2015

Hand-in: – (in the boxes in front of HG G 53/54).

Version compiled on: January 22, 2016 (v. 1.0).

References

[1] R. Hiptmair. *Lecture slides for course "Numerical Methods for CSE"*. http://www.sam.math.ethz.ch/~hiptmair/tmp/NumCSE/NumCSE15.pdf. 2015.