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.

Solution: See `implicit_rkintegrator.hpp`.

(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].

Solution: The Butcher scheme used corresponds to the Radau RK-SSM of order 3 (see [1, Ex. 12.3.44] for the complete scheme). Thus, the method is A-stable and L-stable.

(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.

Solution: As expected, the observed order of convergence is 3 (see [1, Ex. 12.3.44]).

Problem 2 Initial Value Problem With Cross Product

We consider the initial value problem

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}) := \mathbf{a} \times \mathbf{y} + c\mathbf{y} \times (\mathbf{a} \times \mathbf{y}), \quad \mathbf{y}(0) = \mathbf{y}_0 = [1,1,1]^\top, \tag{106}$$

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 (106).

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

Solution: We have

$$\begin{aligned}
&(\mathbf{a} \times \mathbf{y} + c(\mathbf{y} \times (\mathbf{a} \times \mathbf{y}))) \cdot \mathbf{y} = 0 \\
&\Rightarrow D_t \|\mathbf{y}(t)\|_2^2 = 2\dot{\mathbf{y}}(t) \cdot \mathbf{y}(t) = 0 \\
&\Rightarrow \|\mathbf{y}(t)\|_2^2 = \text{const. } \forall t.
\end{aligned}$$

Thus $\|\mathbf{y}(t)\|_2 = \|\mathbf{y}(0)\|_2 = \|\mathbf{y}_0\|_2$.

(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 $\mathbf{a} = [1,0,0]^\top$.

Solution: Using the definition of cross product, a simple but tedious calculation shows that

$$D\mathbf{f}(\mathbf{y}) = \begin{bmatrix} -ca_2y_2 - ca_3y_3 & -a_3 + 2ca_1y_2 - ca_2y_1 & a_2 - ca_3y_1 + 2ca_1y_3 \\ a_3 - ca_1y_2 + 2ca_2y_1 & -ca_3y_3 - ca_1y_1 & -a_1 + 2ca_2y_3 - ca_3y_2 \\ -a_2 + 2ca_3y_1 - ca_1y_3 & a_1 - ca_2y_3 + 2ca_3y_2 & -ca_1y_1 - ca_2y_2 \end{bmatrix}.$$

Thus for $\mathbf{y} = \mathbf{a}$ we obtain

$$D\mathbf{f}(\mathbf{a}) = \begin{bmatrix} -c(a_2^2 + a_3^2) & -a_3 + ca_1a_2 & a_2 + ca_3a_1 \\ a_3 + ca_1a_2 & -c(a_1^2 + a_3^2) & -a_1 + ca_2a_3 \\ -a_2 + ca_1a_3 & a_1 + ca_2a_3 & -c(a_1^2 + a_2^2) \end{bmatrix}.$$

A direct calculation gives that the spectrum is given by

$$\sigma(D\mathbf{f}(\mathbf{a})) = \{0, -c\,\|a\|_2^2 \pm i\,\|a\|_2\} = \{0, -c \pm i\}.$$

Therefore, the problem is stiff for large c (see [1, § 12.2.14]).

(2c) ⚀ For $\mathbf{a} = [1, 0, 0]^\top$, (106) 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 25.

Solution: In the plot, we see that, for the solver `ode45`, the number of steps rises with c. On the other hand, `ode23s` uses roughly the same amount of steps, irregardless of the value of c chosen.

As `ode45` is an explicit solver, it suffers under stability-based step-size limits for large $c > 0$. The implicit solver `ode23s` must however not take smaller step-sizes to satisfy the tolerance for big c. In other words: the problem becomes stiffer, the greater the parameter c is. This is expected from what we saw above.

(2d) ⚀ Formulate the non-linear equation given by the implicit mid-point rule for the initial value problem (106).

Solution: With the formula for the implicit mid-point rule

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{f}\left(\frac{\mathbf{y}_k + \mathbf{y}_{k+1}}{2}\right)$$

the formulation of (106) is given as

$$\frac{\mathbf{y}_{k+1} - \mathbf{y}_k}{h} = \mathbf{a} \times \left(\frac{\mathbf{y}_k + \mathbf{y}_{k+1}}{2}\right) + c\left(\frac{\mathbf{y}_k + \mathbf{y}_{k+1}}{2}\right) \times \left(\mathbf{a} \times \left(\frac{\mathbf{y}_k + \mathbf{y}_{k+1}}{2}\right)\right)$$

(2e) ⚀ Solve (106) 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?

Solution: See file `cross.cpp`. As expected from (2a), the norm of the approximate states is constant.

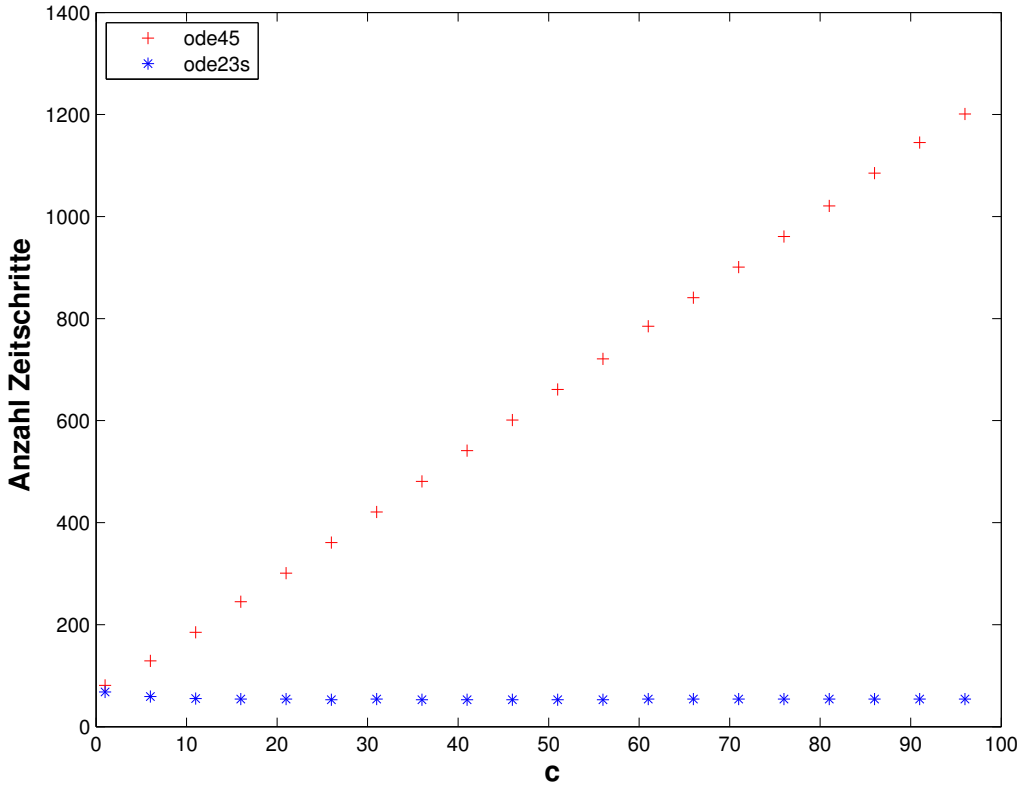

Figure 25: Subproblem (2c): number of steps used by standard MATLAB integrators in relation to the parameter c.

(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.

Solution: The linear implicit mid-point rule is obtained by developing the increment $\mathbf{k}_1$ of the implicit mid point rule by its Taylor series

$$\mathbf{k}_1 = f\left(\mathbf{y}_k + \frac{h}{2}\mathbf{k}_1\right) = \mathbf{f}(\mathbf{y}_k) + \frac{h}{2}D\mathbf{f}(\mathbf{y}_k)\mathbf{k}_1 + O(h^2)$$

and only taking the linear terms. Since the non-linear method is given by $\mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{k}_1$, the linearization reads

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{k}_1^{\text{lin.}}, \qquad \mathbf{k}_1^{\text{lin.}} := \left(I - \frac{h}{2} D\mathbf{f}(\mathbf{y}_k)\right)^{-1} \mathbf{f}(\mathbf{y}_k).$$

(2g) ⚂ Implement the linear–implicit midpoint rule using the template provided in `cross_template.cpp`. Use this method to solve (106) 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?

Solution: See file `cross.cpp`. The sequence of the norms is not exactly constant: this is due to the approximation introduced with the linearization.

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

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}) \tag{107}$$

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

$$\begin{aligned} \mathbf{W}\mathbf{k}_1 &= \mathbf{f}(\mathbf{y}_0) \\ \mathbf{W}\mathbf{k}_2 &= \mathbf{f}(\mathbf{y}_0 + \frac{1}{2}h\mathbf{k}_1) - ah\mathbf{J}\mathbf{k}_1 \\ \mathbf{y}_1 &= \mathbf{y}_0 + h\mathbf{k}_2 \end{aligned} \tag{108}$$

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 (108), 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 $\dot{y} = \lambda y, \lambda \in \mathbb{C}$.

Solution: For a scalar ODE, the Jacobian is just the derivative w.r.t. y, whence $J = Df(y) = f'(y) = (\lambda y)' = \lambda$. The quantity $\mathbf{W}$ is, therefore, a scalar quantity as well:

$$W = 1 - ahJ = 1 - ah\lambda$$

If we plug everything together and assume h is small enough:

$$\begin{aligned}
y_1 &= y_0 + h\frac{\mathbf{f}(\mathbf{y}_0 + \frac{1}{2}h\mathbf{k}_1) - ah\mathbf{J}\mathbf{k}_1}{W} \\
&= y_0 + h\frac{\mathbf{f}(\mathbf{y}_0 + \frac{1}{2}h\frac{\mathbf{f}(\mathbf{y}_0)}{W}) - ah\mathbf{J}\frac{\mathbf{f}(\mathbf{y}_0)}{W})}{W} \\
&= y_0 + h\frac{\lambda(\mathbf{y}_0 + \frac{1}{2}h\frac{\lambda\mathbf{y}_0}{1-ah\lambda}) - \frac{ah\lambda^2 y_0}{1-ah\lambda}}{1 - ah\lambda} \\
&= \frac{(1-ah\lambda)^2 + h\lambda(1-ah\lambda) + \frac{1}{2}h^2\lambda^2 - ah^2\lambda^2}{(1-ah\lambda)^2} y_0 \\
&= \frac{(1 + a^2z^2 - 2az) + z(1-az) + \frac{1}{2}z^2 - az^2}{(1-az)^2} y_0 \\
&= \frac{1 + (1-2a)z + (\frac{1}{2} - 2a + a^2)z^2}{(1-az)^2} y_0
\end{aligned}$$

Since $\frac{1}{2} - 2a + a^2 = 0$, it follows

$$S(z) = \frac{1 + (1-2a)z}{(1-az)^2}$$

(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 (108).

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

Solution: We compute:

$$\begin{aligned} S(0) &= 1, \\ S'(0) &= 1, \\ S''(0) &= 4a - 2a^2, \\ S'''(0) &= 18a^2 - 12a^3, \end{aligned}$$

therefore

$$S(z) = 1 + z + (2a - a^2)z^2 + (3a^2 - 2a^3)z^3 + O(z^4).$$

We can also compute the expansion of exp:

$$\exp(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + O(z^4)$$

and since $2a - a^2 = \frac{1}{2}$ but $3a^2 - 2a^3 = \frac{1}{2}(\sqrt{2} - 1) \neq \frac{1}{6}$, we have:

$$|S(z) - \exp(z)| = O(|z|^3)$$

Using [1, Lemma 12.1.21], we deduce that the maximal order q of the scheme is $q = 2$.

(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 (108).

HINT: See `rosenbrock_template.cpp`.

Solution: See `rosenbrock.cpp`.

(3d) ⚂ Explore the order of the method (108) empirically by applying it to the IVP for the limit cycle [1, Ex. 12.2.5]:

$$\mathbf{f}(\mathbf{y}) := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{y} + \lambda(1 - \|\mathbf{y}\|^2)\mathbf{y}, \tag{109}$$

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:

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

Solution: The Jacobian Df is:

$$D\mathbf{f}(\mathbf{y}) = \begin{bmatrix} \lambda(1 - \|\mathbf{y}\|^2) - 2\lambda y_0^2 & -1 - 2\lambda y_1 y_0 \\ 1 - 2\lambda y_1 y_0 & \lambda(1 - \|\mathbf{y}\|^2) - 2\lambda y_1^2 \end{bmatrix}.$$

For the implementation, cf. `rosenbrock.cpp`.

(3e) ⚄ Show that the method (108) 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:

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

Solution: We start by proving that the method is A-stable [1, § 12.3.30], meaning that $S(z) < 1, \forall z \in \mathbb{C}, \operatorname{Re}(z) < 0$. First of all, we can compute the complex norm of the stability function at $z = iy, y \in \mathbb{R}$ as:

$$|S(iy)|^2 = \frac{|1 + (1 - 2a)iy|^2}{|(1 - aiy)^2|^2} = \frac{1 + (1 - 2a)^2 y^2}{(1 + a^2 y^2)^2} = \frac{1 + (1 - 4a + 4a^2)y^2}{(1 + 2a^2 y^2 + a^4 y^4)}.$$

Notice that $1 - 4a + 4a^2 = 2a^2$, therefore:

$$|S(iy)|^2 = \frac{1 + 2a^2y^2}{1 + 2a^2y^2 + a^4y^4} < 1.$$

The norm of the function S is bounded on the imaginary axis. Observe that $|S(z)| \to 0, |z| \to \infty$, which, follows from the fact that the degree of the denominator of S is bigger than the polynomial degree of the numerator. Notice that $1 - 2a = 1 - 2\frac{1}{\sqrt{(2)}+2} = \sqrt{2} - 2$.

The only pole of the function is at $z = 1/a$, therefore the function is holomorphic on the left complex plane.

Applying the theorem of the hint on concludes that the absolute value of the function is bounded by 1 on the left complex plane.

The L-stability follows immediately with A-stability and the fact that the absolute value of the function converges to zero as $|z| \to \infty$.

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

$$\begin{array}{c|c} \mathbf{c} & \mathfrak{A} \\ \hline & \mathbf{b}^T \end{array} := \begin{array}{c|ccc} c_1 & \gamma & \cdots & 0 \\ c_2 & a_{21} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{s,s-1} \ \gamma \\ \hline & b_1 & \cdots & b_{s-1} \ b_s \end{array}, \tag{110}$$

with $\gamma \neq 0$.

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

$$\ddot{y} + \dot{y} + y = 0, \qquad y(0) = 1, \quad \dot{y}(0) = 0 \tag{111}$$

should be solved numerically using a SDIRK-method (**S**ingly **D**iagonally **I**mplicit **R**unge-

Kutta Method). It is a Runge-Kutta method described by the Butcher scheme

$$\begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline & 1/2 & 1/2 \end{array}. \tag{112}$$

(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 (112) applied to the initial value problem corresponding to the differential equation $\dot{\boldsymbol{y}} = \mathbf{f}(t, \boldsymbol{y})$.

Solution: The increments $\boldsymbol{k}_i$ are given by

$$\begin{aligned} \boldsymbol{k}_1 &= f\left(t_0 + h\gamma, \boldsymbol{y}_0 + h\gamma \boldsymbol{k}_1\right), \\ \boldsymbol{k}_2 &= f\left(t_0 + h(1-\gamma), \boldsymbol{y}_0 + h(1-2\gamma)\boldsymbol{k}_1 + h\gamma \boldsymbol{k}_2\right). \end{aligned}$$

(4c) ⊡ Show that, the stability function $S(z)$ of the SDIRK-method (112) 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.

Solution: The stability function $S(z)$ of a method is derived by applying the method to the scalar, linear test equation

$$\dot{y}(t) = \lambda y(t)$$

The solution can be written as

$$y_{k+1} = S(z) y_k$$

where $S(z)$ is the stability function and $z := h\lambda$.

In the case of the SDIRK-method, we get

$$\begin{aligned} k_1 &= \lambda(y_k + h\gamma k_1) \\ k_2 &= \lambda(y_k + h(1-2\gamma)k_1 + h\gamma k_2), \end{aligned}$$

therefore

$$\begin{aligned} k_1 &= \frac{\lambda}{1-h\lambda\gamma} y_k, \\ k_2 &= \frac{\lambda}{1-h\lambda\gamma}(y_k + h(1-2\gamma)k_1). \end{aligned}$$

Furthermore

$$y_{k+1} = y_k + \frac{h}{2}(k_1 + k_2),$$

with $z := h\lambda$ and by plugging in k_1 and k_2 we arrive at

$$y_{k+1} = \underbrace{\left(1 + \frac{z}{2(1-\gamma z)}\left(2 + \frac{z(1-2\gamma)}{1-\gamma z}\right)\right)}_{=:S(z)} y_k.$$

Hence

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

and after collecting the powers of z in the numerator we get

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

For $\gamma = 1$ the stability function is therefore

$$S_1(z) := \frac{1 - z - \frac{z^2}{2}}{(1-z)^2}. \tag{113}$$

Verification of the A-stability of (113)**:**
By definition [1, § 12.3.30], we need to show that $|S_1(z)| \leq 1$ for all $z \in \mathbb{C}^- := \{z \in$

$\mathbb{C} \mid \operatorname{Re} z < 0\}$. In order to do this we consider the stability function on the imaginary axis

$$\begin{aligned} |S_1(iy)|^2 &= \frac{|1 - iy - (iy)^2/2|^2}{|1 - iy|^4} \\ &= \frac{|1 - iy + y/2|^2}{|1 - iy|^4} \\ &= \frac{(1 + y^2/2)^2 + y^2}{(1 + y^2)^2} \\ &= \frac{1 + 2y^2 + y^4/4}{1 + 2y^2 + y^4} \leq 1, \quad y \in \mathbb{R}. \end{aligned}$$

Since the only pole ($z = 1$) of the rational function $S_1(z)$ lies in the positive half plane of $\mathbb{C}$, the function S_1 is holomorphic in the left half plane. Furthermore S_1 is bounded by 1 on the boundary of this half plane (i.e. on the imaginary axis). So by the maximum principle for holomorphic functions (hint) S_1 is bounded on the entire left half plane by 1. This implies in particular that S_1 is A-stable.

Verification of the L-stability of (113)**:**
S_1 is not L-stable (cf. definition [1, § 12.3.37]), because

$$\lim_{\operatorname{Re} z \to -\infty} |S_1(z)| = \lim_{\operatorname{Re} z \to -\infty} \left| \frac{1 - z - z^2/2}{1 - 2z + z^2} \right| = \frac{1}{2} \neq 0.$$

(4d) ⊡ Formulate (111) as an initial value problem for a linear first order system for the function $\boldsymbol{z}(t) = (y(t), \dot{y}(t))^\top$.

Solution: Define $z_1 = y$, $z_2 = \dot{y}$, then the initial value problem

$$\ddot{y} + \dot{y} + y = 0, \qquad y(0) = 1, \quad \dot{y}(0) = 0 \tag{114}$$

is equivalently to the first order system

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= -z_1 - z_2, \end{aligned} \tag{115}$$

with initial values $z_1(0) = 1$, $z_2(0) = 0$.

(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 (112) 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`.

Solution: Let

$$\mathbf{A} := \begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}.$$

Then

$$\begin{aligned} k_1 &= \mathbf{A}\boldsymbol{y}_0 + h\gamma\mathbf{A}k_1 \\ k_2 &= \mathbf{A}\boldsymbol{y}_0 + h(1-2\gamma)\mathbf{A}k1 + h\mathbf{A}k_2, \end{aligned}$$

so

$$\begin{aligned} k_1 &= (\mathbf{I} - h\gamma\mathbf{A})^{-1}\mathbf{A}\boldsymbol{y}_0 \\ k_2 &= (\mathbf{I} - h\gamma\mathbf{A})^{-1}(\mathbf{A}\boldsymbol{y}_0 + h(1-2\gamma)\mathbf{A}k_1). \end{aligned}$$

See the implementation in `sdirk.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 $\boldsymbol{y}_0 = [1,0]^\mathsf{T}$ as initial value, `T=10` as end time and `N=20,40,80,...,10240` as steps.

Solution: The numerically estimated order of convergence is 3, see `sdirk.cpp`.

Issue date: 17.12.2015

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

Version compiled on: February 14, 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.