

## Differential Algebraic Equations Exercise Sheet 4 – One Step Methods

We consider the initial value problems given by an ODE

$$\dot{x} = f(t, x), \quad x(t_0) = x_0, \quad (\text{ODE})$$

and a DAE

$$F(t, x, \dot{x}) = 0, \quad x(t_0) = x_0, \quad (\text{DAE})$$

on the time interval  $[t_0, t_e]$ . On the grid

$$t_0 < t_1 < \dots < t_N = t_e,$$

where  $N \in \mathbb{N}$ , where  $t_k = t_0 + kh$ , and where  $h = \frac{t_e - t_0}{N}$ , let  $x_k$  denote a numerical approximation to  $x(t_k)$ , where  $x$  is a solution to (ODE) or (DAE). We use the general form

$$x_{k+1} = \phi(t_k, h, x_k, x_{k+1})$$

to refer to a given numerical scheme.

**A Effect of Rounding Errors** Consider the *Explicit Euler* scheme

$$x_{k+1} = x_k + hf(t_k, x_k)$$

and show that the error  $e_k = x_k - y_k$ , where  $y_k$  are the iterates of the perturbed scheme

$$y_{k+1} = y_k + hf(t_k, y_k) + \varepsilon_k,$$

and where  $\varepsilon_k$  denotes the rounding error, behaves like  $\|e_N\| \approx \max_k \{\varepsilon_k\} h^{-1}$ .

**B Order of Consistency** Determine the order of consistency of the Runge–Kutta scheme

$$\begin{array}{c|cc} 0 & 0 & \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array}$$

if applied to (ODE) by directly estimating  $\|x(t_{k+1}) - \phi(t_k, h, x(t_k), x(t_{k+1}))\|$ . For simplicity, assume that the ODE is *autonomous*, i.e., that the right hand side  $f$  depends only on  $x(t)$  but not explicitly on  $t$ .

**C Two-stage Gauss method for ODEs and DAEs** Consider the *Gauss* method of order 2:

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

1. Determine the order of convergence for ODEs using *Butcher's Theorem* (lecture: Thm. 5.1).
2. Show that  $\kappa_1 = 2$  and  $\kappa_2 = 2$  as defined in the lecture: (to be announced). What does this mean for the numerical approximation of DAEs?

**D Kronecker and Runge–Kutta** Show that for the DAE with constant coefficients  $E\dot{x} = Ax + f(t)$ , the stage derivatives  $\dot{X}_{ij}$ ,  $j = 1, \dots, s$  at time step  $i$  of an  $s$ -stage Runge–Kutta method  $(\mathcal{A}, \beta, \gamma)$ , are defined through the linear system

$$(I_s \otimes E - h\mathcal{A} \otimes A)\dot{X}_i = Z_i,$$

where  $\dot{X}_i := [\dot{X}_{il}]_{l=1, \dots, s}$  and  $Z_i = [Ax_i + f(t_i + \gamma lh)]_{l=1, \dots, s}$ .

Please turn the sheet.

## Coding Exercises

1. Implement the *explicit Euler* scheme for constant step sizes and test it on the ODE:

$$\begin{aligned}\dot{x}_1 &= e^t x_2, & x_1(0) &= \sin(1), \\ \dot{x}_2 &= -e^t x_1, & x_2(0) &= \cos(1),\end{aligned}$$

through numerically computing  $u_1(3)$  for stepsizes  $h = 3/2^k$ ,  $k = 5, 6, \dots$ . What do you observe?

2. Implement the *implicit Euler* scheme for DAEs for constant step sizes as a method that takes

- a function  $F(t, x, \dot{x})$ ,
- an initial value  $x(t_0) = x_0$ ,
- an interval  $\mathbb{I} = [t_0, T]$ ,
- and a number of discretization points  $N$ .

It should return an array of the computed solution and a plot of it.

Test your implementation on the following cases:

- (a)  $F = E\dot{x} - Ax - f$ ,  $x_0 = [0, 1]^T$ ,  $\mathbb{I} = [0, 10]$ , and

$$E(t) = \begin{bmatrix} -t & t^2 \\ -1 & t \end{bmatrix}, \quad A(t) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad f(t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

- (b)  $F = E\dot{x} - Ax - f$ ,  $x_0 = [1, 1]^T$ ,  $\mathbb{I} = [0, 10]$ , and

$$E(t) = \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix}, \quad A(t) = \begin{bmatrix} -1 & t \\ 0 & 0 \end{bmatrix}, \quad f(t) = \begin{bmatrix} \sin(t) \\ \cos(t) \end{bmatrix}.$$

- (c)  $F = E\dot{x} - Ax - f$ ,  $x_0 = [0, 1]^T$ ,  $\mathbb{I} = [0, 10]$ , and

$$E(t) = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \quad A(t) = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, \quad f(t) = \begin{bmatrix} e^t + \cos(t) \\ e^t \end{bmatrix}.$$

Compare the outcomes to the actual solutions (cf. the introducing examples of Section 4 in the lecture). Are the initial values consistent? If not, determine consistent ones? What happens for inconsistent initial values?

**Remark:** The implementation involves the solution of a (nonlinear) system of equations. Typically, one would implement a *Newton iteration* for that. You may, however, simply call a builtin function like *Matlab's* `fsolve` or *SciPy's* `optimize.fsolve`.