P. Grohs
W. Wu

Spring Term 2016

Numerical Analysis II

ETH Zürich
D-MATH

Problem Sheet 8

Problem 8.1 Autonomisation and Strang-Splitting

The differential equation

$$\dot{\mathbf{y}} = \mathbf{A}(t)\mathbf{y}, \qquad \mathbf{y}(t_0) = \mathbf{y}_0, \quad \mathbf{A}(t) = t\mathbf{A}_0, \quad \mathbf{A}_0 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{8.1.1}$$

can be autonomised by introducing a variable $\mathbf{z} := \begin{bmatrix} \mathbf{y} \\ t \end{bmatrix}$, which yields the ODE

$$\dot{\mathbf{z}} = F(\mathbf{z}) = \begin{bmatrix} \mathbf{A}(t)\mathbf{y} \\ 1 \end{bmatrix}. \tag{8.1.2}$$

(8.1a) Formulate the discrete evolution of a Strang splitting method for (8.1.2) corresponding to the decomposition

$$F(\mathbf{z}) = \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{=:\mathbf{f}(\mathbf{z})} + \underbrace{\begin{bmatrix} \mathbf{A}(t)\mathbf{y} \\ 0 \end{bmatrix}}_{=:\mathbf{g}(\mathbf{z})}$$

based on the exact evolutions for $\mathbf{f}$ and $\mathbf{g}$.

HINT: Use the representation of the exact solution of an IVP for a homogeneous linear differential equation .

(8.1b) Implement the resulting method by completing the MATLAB template

```
[t, y] = strangaut(y0, t0, T, h)
```

HINT: The matrix exponential function can be computed using `expm`.

(8.1c) Plot the numerical solution obtained by applying the splitting method to the IVP (8.1.1) where $\mathbf{y}_0 = [1, 0]^\top$, $t_0 = 0$, $T = 10$, $h = 0.01$. To do so, complete the template `strangautplot.m`. Use time t as your z-coordinate.

HINT: You might find the function `plot3` to be of use.

Problem 8.2 Lie-Trotter Splitting Method

(8.2a) Write a MATLAB function

```
function [t,y] = lietrotter(psi1, psi2, y0, tspan, N)
```

which implements the Lie-Trotter splitting method on the time interval `tspan`.

(8.2b) Write a MATLAB function

```
function [t,y] = strang(psi1,psi2,psi3,y0,tspan,N)
```

which implements the Strang splitting method on the time interval `tspan`.

Problem 8.3 Extrapolation of Reversible One-step Methods

We will limit ourselves to the autonomous initial value problem

$$\dot{\mathbf{y}} = -\mathbf{y} \quad \mathbf{y}(0) = 1.$$

(8.3a) Implement a MATLAB function

```
y = ImplMpr(y0,h,n),
```

that performs `n` steps of the implicit midpoint rule with step size `h` from the initial value y0.

(8.3b) In a MATLAB function

```
y = extraImplMpr(y0,T,N,n),
```

implement the extrapolated implicit mid point rule. The input should be given as:`y0`, the initial value, `T`, the end point, `N`, the number of macro steps and the vector `n`, which represents the sequence of the number of micro steps.

HINT: Use the function `extrapolate(Y,n,p)` from `extrapolate.m`.

(8.3c) Complete the template `extraImplMprKonv.m`, which performs a convergence study of the extrapolated implicit mid point rule. Let the method run for different sequences of numbers of micro steps (for example:$n = 1, 2$; $n = 1, 2, 3$; $n = 2, 4, 8$). What do you observe? Read [NUMODE, Thm. 2.4.22] and comment on your results.

HINT: Recall that the round-off error can be relevant for methods of high order (f.e. for $n = 1, 2, 3, 4, 5, 6, 7$).

Published on 18 April 2016.

To be submitted by 26 April 2016.

References

[NODE] Lecture Notes for the course "Numerical Methods for Ordinary Differential Equations".

[NUMODE] Lecture Slides for the course "Numerical Methods for Ordinary Differential Equations", SVN revision # 52913.

Last modified on April 22, 2016