P. Grohs
S. Hosseini
Ž. Kereta

Spring Term 2015

Numerical Analysis II

ETH Zürich
D-MATH

Problem Sheet 4

Problem 4.1 Linear Independence of Elementary Differentials

Show that the elementary differentials

$$f(\mathbf{y}_0), \quad Df(\mathbf{y}_0)f(\mathbf{y}_0), \quad Df(\mathbf{y}_0)Df(\mathbf{y}_0)f(\mathbf{y}_0), \quad D^2f(\mathbf{y}_0)(f(\mathbf{y}_0), f(\mathbf{y}_0))$$

(cf. example [NUMODE, Ex. 2.3.24]) are linearly independent as mappings $\mathcal{C}^2(\mathrm{U}_\epsilon(\mathbf{y}_0), \mathbb{R}^2)$. In other words, show that from

$$\alpha_1 f(\mathbf{y}_0) + \alpha_2 Df(\mathbf{y}_0)f(\mathbf{y}_0) + \alpha_3 Df(\mathbf{y}_0)Df(\mathbf{y}_0)f(\mathbf{y}_0) + \alpha_4 D^2f(\mathbf{y}_0)(f(\mathbf{y}_0), f(\mathbf{y}_0)) = 0,$$

for all $f \in \mathcal{C}^2(\mathrm{U}_\epsilon(\mathbf{y}_0), \mathbb{R}^2)$, it follows that $\alpha_i = 0$ for all $i = 1, \ldots, 4$.

Problem 4.2 Implementation of the Gaussian Collocation Method

As explained in section [NUMODE, Sect. 2.2] of the lecture, a collocation one-step method $\mathbf{y}_1 = \Psi^{t_0, t_0+h}\mathbf{y}_0$ with collocation points

$$t_0 \le t_0 + c_1 h < \cdots < t_0 + c_s h \le t_0 + h = t_1, \qquad s \in \mathbb{N}$$

for the ODE $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ can be described by

$$\begin{aligned} \mathbf{k}_i &= \mathbf{f}(t_0 + c_i h, \mathbf{y}_0 + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j) \\ \mathbf{y}_1 &:= \mathbf{y}_h(t_1) = \mathbf{y}_0 + h\sum_{i=1}^{s} b_i \mathbf{k}_i \end{aligned} \qquad \text{with} \qquad \begin{aligned} a_{ij} &= \int_0^{c_i} L_j(\tau)\,\mathrm{d}\tau \\ b_i &= \int_0^1 L_i(\tau)\,\mathrm{d}\tau\ . \end{aligned} \tag{4.2.1}$$

Here,

$$L_i(\xi) = \prod_{\substack{j=1 \\ j\neq i}}^{s} \frac{\xi - c_j}{c_i - c_j}, \qquad i = 1, \ldots s$$

denote the Lagrange polynomials, ie. $L_i(\tau_j) = \delta_{ij}$. The coefficients a_{ij}, $1 \le i, j \le s$ are collected in the matrix $\mathfrak{A} \in \mathbb{R}^{s\times s}$.

(4.2a) Write a MATLAB function

```
function [A,b] = collCoeff(c)
```

which takes the relative positions $c_i \in [0, 1]$ of the collocation points as a vector $\mathbf{c} \in \mathbb{R}^s$ and returns the coefficients of the matrix $\mathbf{A} \in \mathbb{R}^{s\times s}$ and the vector $\mathbf{b} \in \mathbb{R}^s$ with $(\mathbf{A})_{ij} = a_{ij}$ and $(\mathbf{b})_i = b_i$ (see equation (4.2.1) in the introduction) for the related collocation method.

HINT: Familiarize yourself with the MATLAB functions `polyint` und `polyval`. `vander` may also be of use.

(4.2b) If we choose the collocation points c_i to be the roots of the Legendre polynomial of n^{th} degree for the interval $[0, 1]$, we call the resulting method *Gaussian collocation one-step method*. This method inherits the convergence properties of the Gaussian quadrature, meaning its convergence order is $2n$. Write a MATLAB function

```
function c = GaussKnoten(n)
```

which returns the roots of the Legendre polynomial of n^{th} degree on the interval $[0, 1]$.
HINT: The algorithm of Golub-Welsch returns the roots of the Legendre polynomial of n^{th} degree for the interval $[-1, 1]$.

(4.2c) The Gaussian collocation one-step method is implicit and is usually used with Newton's method. Let $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ be a function, of which we want to find the roots. Modify the code MATLAB function `newton(x0, F, DF)`, Problem Sheet 3, so that the function performs `nNewton` steps of Newton's method.

The input of the function `newton(x0,F,DF,nNewton)` is the initial data `x0`, function handles `F` and `DF` of $\mathbf{F}$ and its Jacobian $\mathbf{D_y F}$, and the number of iterations `nNewton`.

(4.2d) Implement the implicit Gaussian collocation method of order 4: find the coefficients using the Matlab function `[A,b]=collCoeffs(c)` and the vector $\mathbf{b} \in \mathbb{R}^s$ with $(\mathbf{A})_{ij} = a_{ij}$ and $(\mathbf{b})_i = b_i$ and subproblem (4.2b) and rephrase the method as a root-finding problem. Apply your implementation of Newton's method from subproblem (4.2c) to it. Complete the template `impGauss.m` with inputs: `y0`, the starting value of the initial value problem, `f`, the right hand side of the initial value problem, `Df`, the Jacobian of the right hand side, `T`, the end point, `Nh`, the number of steps and `nNewton`, the number of iterations of Newton's method.

(4.2e) Consider the initial value problem

$$\dot{\mathbf{y}} = \exp(\mathbf{y}) \sin(\mathbf{y}); \quad \mathbf{y}(0) = \pi/4.$$

Find the absolute error of the Gaussian collocation method at the point `T=0.5` for a variation of the number of steps $N_h = 2^i$, $i = 4, \ldots, 9$ and of the number of Newton iterations `nNewton`$= 1, 2, 3$. Plot the error curves vs. the number of steps in logarithmic axes scale and find the algebraic convergence order with the MATLAB function `polyfit`. Use the template `GaussConv.m`.
HINT: You can find a reference solution with `ode45`. Set the relative and absolute tolerance to 10^{-12}.

Problem 4.3 Construction of One-Step Methods

This exercise studies the so called *Taylor series method* to construct one-step methods. For a given autonomous system $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$, we define a family of discrete evolutions $\Psi_n^{t,s}$, $n \in \mathbb{N}$ by

$$\Psi_n^{t,t+h}\mathbf{z} := \mathbf{z} + \sum_{i=1}^{n} \frac{h^i}{i!} \Delta_i \mathbf{z}, \qquad t, t+h \in J(\mathbf{z}),$$

where

$$\Delta_i \mathbf{z} := \frac{\mathrm{d}^{i-1}}{\mathrm{d}s^{i-1}} \mathbf{f}(\mathbf{y}(s))\Big|_{s=t} \qquad \text{with} \qquad \frac{\mathrm{d}}{\mathrm{d}s}\mathbf{y}(s) = \mathbf{f}(\mathbf{y}(s)), \quad \mathbf{y}(t) = \mathbf{z}.$$

(4.3a) Show that the discrete evolutions $\Psi_n^{t,s}$ are consistent with the autonomous system $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$.
HINT: Apply [NUMODE, Lem. 2.1.9].

(4.3b) Express Δ_1, Δ_2 and Δ_3 symbolically in relation to the (sufficiently smooth) right hand side $\mathbf{f}$ and its derivatives.

(4.3c) Find the order of consistency $\boldsymbol{\tau}(t, \mathbf{y}, h)$ of these methods for autonomous systems.

(4.3d) We will now look at the scalar logistic differential equation

$$\dot{y} = 2y(1 - y). \tag{4.3.1}$$

Write three MATLAB functions

```
function y = trv_1(h,T,y0)
function y = trv_2(h,T,y0)
function y = trv_3(h,T,y0)
```

in which, for a given step size `h`, end point `T` and initial value `y0`, you calculate the solution to (4.3.1) using Ψ_1, Ψ_2 and Ψ_3 respectively.

(4.3e) Write a MATLAB function

```
function p = trvconv
```

in which you calculate the convergence order of Ψ_1, Ψ_2 and Ψ_3 for (4.3.1).
HINT: Choose $y(0) = 2$ as initial value and calculate the error at the end point $T = 2$ for different step sizes $h = 2^{-4}, 2^{-5}, \ldots, 2^{-10}$. Then, apply a linear interpolation of the logarithm of the error vs. the logarithm of the stepsize with the MATLAB function `polyfit`.
HINT: The exact solution of (4.3.1) with initial value y_0 is $y(t) = y_0(y_0 + (1 - y_0)\mathrm{e}^{-2t})^{-1}$.

(4.3f) Formulate the discrete evolutions Ψ_n for the linear problem

$$\dot{\mathbf{y}} = \boldsymbol{A}\mathbf{y}, \qquad \mathbf{y} \in \mathbb{R}^d, \quad \boldsymbol{A} \in \mathbb{R}^{d\times d},$$

and general $n \in \mathbb{N}$. Implement Ψ_n, $n = 1, 2, 3$ for $\boldsymbol{A} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ and test which of these methods is length-preserving.

HINT: First show, that $\Delta_i \mathbf{z} = \boldsymbol{A}^i \mathbf{z}$.

Published on 10 March 2015.
To be submitted by 17 March 2015.

References

[NUMODE] Lecture Slides for the course “Numerical Methods for Ordinary Differential Equations”, SVN revision # 63606.

Last modified on March 9, 2015