P. Grohs
W. Wu

Winter Term 2017

Numerical Analysis II

ETH Zürich
D-MATH

Exam Winter 2017

Last Name		Note
First Name		
Degree Programme		
Legi Number		
Date		

1	2	3	4	Marks

- First fill out the cover sheet and place your Legi on the edge of the desk.
- Begin each problem on a separate sheet of paper. Please write out the problem ID in a striking font.
- Every sheet must bear your name and Legi number.
- Write with neither red nor green pens nor with a pencil.
- Please write out your ideas clearly and show your reasoning rigorously.
- You may not start to read the questions printed on the subsequent pages until instructed to do so by the Invigilator.

Good luck!

Exam Winter 2017

Problem 1 Implementation of the Gaussian Collocation Method [25.5 Marks]

As explained in the section [NODE, Def. 2.2.1] of the lecture notes, a one-step collocation scheme $\mathbf{y}_1 = \Psi^{t_0,t_0+h}\mathbf{y}_0$ for the solution of the ODE $\dot{\mathbf{y}} = \mathbf{f}(t,\mathbf{y})$, with collocation points

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

can be described by

$$\begin{aligned} \mathbf{k}_i &= \mathbf{f}(t_0 + c_i h, \mathbf{y}_0 + h \sum_{j=1}^{s} a_{ij}\mathbf{k}_j) \\ \mathbf{y}_1 := \mathbf{y}_h(t_1) &= \mathbf{y}_0 + h\sum_{i=1}^{s} b_i \mathbf{k}_i \end{aligned} \qquad \text{with} \qquad \begin{aligned} a_{ij} &= \int_0^{c_i} L_j(\tau)\,\mathrm{d}\tau \\ b_i &= \int_0^1 L_i(\tau)\,\mathrm{d}\tau\ . \end{aligned} \tag{1.1}$$

Here

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

are the Lagrange polynomials. The coefficients a_{ij}, $1 \le i,j \le s$ are collected in the matrix $\mathbf{A} \in \mathbb{R}^{s\times s}$.

(1a) ⚃ 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 $(\mathbf{b})_i = b_i$.

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

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

```
function c = GaussNodes(n)
```

which returns the roots of the Legendre polynomial of n^{th} degree on the interval $[0, 1]$.
HINT: The Golub-Welsch algorithm returns the roots of the Legendre polynomial of n^{th} degree on the interval $[-1, 1]$. To be specific, The roots $c_1, \ldots, c_n$ of the Legendre polynomial of degree n for the interval $[-1, 1]$ are the eigenvalues of the matrix

$$\begin{pmatrix} 0 & b_1 & & \\ b_1 & 0 & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & 0 \end{pmatrix},$$

where $b_j := j(4j^2 - 1)^{-1/2}$.

The eigenvalues of the matrix mentioned in the hint can be calculated with the MATLAB command `eig`. Notice you may need scaling and translation to get the eigenvalue on $[0, 1]$.

(1c) ⚁ 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)` provided in the template, so that the function performs `nNewton` steps of Newton's method.

(1d) ⚃ 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 (1b) and rephrase the method as a root-finding problem. Apply your implementation of Newton's method from subproblem (1c) to it. Complete the template `impGauss.m` with inputs: the initial value `y0` of the IVP, the right hand side `f` of the initial value problem, the Jacobian of the right hand side `Df`, the end point `T`, the number of steps `Nh` and `nNewton`, the number of iterations of Newton's method.

(1e) ⚂ 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 `T=0.5` by varying both the number of steps $N_h = 2^i$, $i = 4, \ldots, 8$ and the number of Newton iterations `nNewton`$= 1, 2, 3$. Plot the error against the number of steps in logarithmic scale and estimate 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 2 Stability Domain of a Rational Single Step Method [24.5 Marks]

Consider the rational function

$$R(z) = \frac{2 - z^2}{2(1 - z)}.$$

(2a) ⚄ Determine the maximal $p \in \mathbb{N}$ such that

$$|\exp(z) - R(z)| = \mathcal{O}(|z|^{p+1}) \qquad \text{for } z \to 0.$$

HINT: Compute the first three derivatives of $R(z)$ and use them to compare the Taylor series of $\exp(z)$ and $R(z)$ around the point 0.

(2b) ⚁ Consider $R(z)$ as a stability function of a Runge-Kutta single step method and plot its stability domain in MATLAB by completing the template `StabilityRegion.m`.

(2c) ⚂ Show that a Runge-Kutta method with stability function $R(z)$ is of convergence order 2 when applied to linear ODEs, that is, to problems of the form $\dot{y} = \lambda y,\ y(0) = y_0$.

(2d) ⚁ Write down (in detail) the discrete evolution of the single step method (whose stability function is $R(z)$), when applied to the autonomous linear differential equation

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y}, \qquad \mathbf{A} \in \mathbb{R}^{d \times d}. \tag{2.1}$$

(2e) ⚄ Implement the method (in MATLAB) for the approximate solution of (2.1) by completing the template `RationalSSM.m` to solve the initial value problem

$$\dot{\mathbf{y}} = \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix} \mathbf{y}, \quad \mathbf{y}(0) = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

for $t \in [0, 10]$ with the values

	(i)	(ii)	(iii)	(iv)	(v)	(vi)
α	-2	-2	-2	1.5	1.5	1.5
β	-1	-2	-2	0	0	0
h	1	1	0.5	0.5	1	1.5

where h is the step size. Plot your results and compare them with the exact solution. **Explain the behaviour of the method with all the six different sets of parameters with the help of the stability domain of $R(z)$.**

Problem 3 ODEs for Matrix-Valued Functions **[23 Marks]**

Let the matrix-valued function $\mathbf{Y}:\mathbb{R}\to\mathbb{R}^{d\times d}$ be a solution of the (matrix) differential equation

$$\dot{\mathbf{Y}} = \mathbf{A}\mathbf{Y} \quad \text{with} \quad \mathbf{A}\in\mathbb{R}^{d\times d}. \tag{3.1}$$

(3a) Assume $\mathbf{A}^\top\mathbf{H} = -\mathbf{H}\mathbf{A}$. Show that $\mathbf{Y}(t)^\top\mathbf{H}\mathbf{Y}(t) = \mathbf{H}$ for all $t > 0$ provided $\mathbf{Y}(0)^\top\mathbf{H}\mathbf{Y}(0) = \mathbf{H}$.

HINT: You might want to compute $\frac{\mathrm{d}}{\mathrm{d}t}$ of $\mathbf{Y}^\top\mathbf{H}\mathbf{Y}$.

(3b) Implement the following functions in MATLAB

(i) `function Y = ExplEulStep(A, Y0, h)`,

(ii) `function Y = ImplEulStep(A, Y0, h)`,

(iii) `function Y = ImplMidpStep(A, Y0, h)`,

which, for a given initial value $\mathbf{Y}(t_0) = \mathbf{Y}_0$ and for a given step size h, compute approximations to $\mathbf{Y}(t_0+h)$ for the solution of (3.1) using a (*single*) step of

(i) the explicit Euler method,

(ii) the implicit Euler method,

(iii) the implicit mid-point method.

For (ii) and (iii), write out the closed form for $\mathbf{Y}_{k+1}$ instead of using Newton's method. Explain how you get the formula on your answer sheet.

(3c) Take now $\mathbf{A} = \begin{pmatrix} -3 & -6 \\ 6 & 3 \end{pmatrix}$, $\mathbf{Y}(0) = \frac{1}{\sqrt{3}}\begin{pmatrix} 2 & 1 \\ -1 & -2 \end{pmatrix}$, and $\mathbf{H} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$. Complete the template `CompareNorms.m` where, using the functions from subproblem (3b), you should compute discrete approximations $\mathbf{Y}_k$ of $\mathbf{Y}(kh)$, for $k = 1,\dots,20$ with $h = 1/20$. Compare the norms $\left\|\mathbf{Y}_k^\top\mathbf{H}\mathbf{Y}_k - \mathbf{H}\right\|_F$, for $k = 1,\dots,20$ and all three methods, and comment on your observations with regards to the invariant from subproblem (3a).

HINT: The Frobenius norm $\|\cdot\|_F$ of a matrix can be computed using the command `norm(A,'fro')`.

(3d) Show that the solution $\mathbf{Y}_k$ computed via the implicit mid-point rule satisfies:

$$\text{if} \quad \mathbf{Y}_0^\top\mathbf{H}\mathbf{Y}_0 = \mathbf{H} \quad \text{then} \quad \mathbf{Y}_k^\top\mathbf{H}\mathbf{Y}_k = \mathbf{H} \quad \text{for all } k \geq 1.$$

HINT: You might find the identity $\mathbf{Y}_1 - \mathbf{Y}_0 = \frac{h}{2}\mathbf{A}(\mathbf{Y}_0+\mathbf{Y}_1)$ useful.

Problem 4 Extrapolating Implicit Trapzoidal **[27 Marks]**

In this problem we will apply the extrapolation method to the implicit Trapzoidal method. Consider the logistic ODE

$$\dot{y} = \lambda y(1-y), \quad \lambda > 0, \tag{4.1}$$

with the initial value $y(0) = y_0 > 0$

(4a) ⚃ Find the fixed points of the ODE (4.1) and determine if any of them are attractive. Explain why given $1 > y_0 > 0$ and $y(t)$ is a smooth solution to the IVP, it follows that $1 > y(t) > 0$ for all $t > 0$. □

(4b) ⚃ The implicit trapezoidal rule for solving the autonomous differential equation $\dot{y} = f(y)$ is given by

$$y_1 = \Psi^{t_0,t_0+h} y_0 := y_0 + \tfrac{1}{2}h[f(y_0) + f(y_1)]. \tag{4.2}$$

Give the closed form of the discrete evolution $\Psi^{t_0,t_0+h}y_0$ of the implicit trapezoidal rule when applied to the logistic differential equation (4.1), and argue whether the solution is admissible assuming the initial value satisfies $1 > y(0) > 0$ given stepsize h small enough. □

HINT: The discrete evolution of the implicit trapezoidal rule leads to a quadratic equation which admits an explicit solution. Then use (4a) to conclude which of the two expressions makes sense for $1 > y(0) > 0$, or none of them may not be admissble.

(4c) ⚂ The method (4.2) can be interpreted as a Runge-Kutta-method. Write down the corresponding Butcher-tableau.

(4d) ⚁ Complete the templates

```
function y = ImplicitTrapzoidal(y0, lambda, h)
```

to carry out the implicit trapzoidal method for (4.1) where the parameters include a given initial value `y0`, positive parameter `lambda` and step size `h`. Use the result in (4b) directly for implicit trapzoidal method. □

(4e) ⚂ Suppose that for ODE (4.1), we have performed a chosen single step method n times with different step sizes $h = (h_1, \cdots, h_n)$ on time interval $[0, t_0]$, where $h_1 < h_2 < \cdots < h_n$. Let T_i be the approximation of $y(t_0)$ for step size h_i, $i = 1, \cdots, n$. Let $T := (T_1, \cdots, T_n)$. Implement a MATLAB function using the template

```
function y = Extrapolation(T, h).
```

that performs Aitken-Neville extrapolation method to compute the extrapolated value for $y(t_0)$. □

(4f) ⚂ Consider again the ODE (4.1), where we take $y(0) = 0.03$, $\lambda = 5$, and complete the template `ExtrapolatedSingleStep.m` which performs a series of the implicit trapzoidal methods with different step sizes and calculate the extrapolated result using `Extrapolation(T,h)` in (4e).

In the program we take `2.^(3:8)` subdivisions of the time interval `[0 1]`. Print out the result at the end of program. □

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.