Problem Sheet 6

Problem 6.1 Invertible Collocation Matrix

Show that, the matrix $\mathfrak{A} \in \mathbb{R}^{s\times s}$ with the entries $(\mathfrak{A})_{ij} = a_{ij}$ is invertible, given that $c_1 > 0$.

HINT: Show that, $\mathfrak{A}\mathbf{z} = 0$ implies $\mathbf{z} = 0$. In order to do this consider the polynomial

$$q(\xi) = \int_0^\xi \sum_{j=1}^s z_j L_j(\tau)\,\mathrm{d}\tau \in \mathcal{P}_s.$$

Solution: The matrix $\mathfrak{A} \in \mathbb{R}^{s\times s}$ is invertible if and only if its kernel is trivial. So let $\mathbf{z}$ be in the kernel of $\mathfrak{A}$. Then

$$\begin{aligned}
& \mathfrak{A}\mathbf{z} = 0 \\
\iff\quad & (\mathfrak{A}\mathbf{z})_i = 0 && \forall i = 1,\dots s \\
\iff\quad & \sum_{j=1}^s a_{ij}(\mathbf{z})_j = 0 && \forall i = 1,\dots s \\
\iff\quad & \sum_{j=1}^s \int_0^{c_i} L_j(\tau)(\mathbf{z})_j\,\mathrm{d}\tau = 0 && \forall i = 1,\dots s \\
\iff\quad & \int_0^{c_i} \sum_{j=1}^s L_j(\tau)(\mathbf{z})_j\,\mathrm{d}\tau = 0 && \forall i = 1,\dots s
\end{aligned}$$

We now consider the polynomial

$$q(\xi) = \int_0^\xi \sum_{j=1}^s L_j(\tau)(\mathbf{z})_j\,\mathrm{d}\tau.$$

Since $L_j(\tau) \in \mathcal{P}_{s-1}$, $q(\xi) \in \mathcal{P}_s$. Obviously $q(0) = 0$ and $q(c_i) = 0$, $i = 1,\dots s$, i.e. $q(\xi) \in \mathcal{P}_s$ has $s+1$ different roots, because $c_1 > 0$ by assumption. Therefore $q(\xi) \equiv 0$. And consequently also $q'(\xi) \equiv 0$, i.e., it holds

$$\sum_{j=1}^s L_j(\xi)(\mathbf{z})_j \equiv 0 \quad\iff\quad (\mathbf{z})_j = 0 \quad \forall j = 1,\dots s.$$

$\mathfrak{A}$ has a trivial kernel.

Problem 6.2 Composition of Runge–Kutta Methods

This exercise investigates the composition of Runge-Kutta methods. It turns out that such a composition is itself always another single step method of Runge-Kutta type:

Let $\hat{\Psi}^{t_0,t_0+h}$ and $\tilde{\Psi}^{t_0,t_0+h}$ be the discrete evolutions for two 2-stage Runge-Kutta single step methods defined by the Butcher-tableaux:

$$\begin{array}{c|cc} 0 & & \\ \widehat{c}_2 & \widehat{a}_{21} & \\ \hline & \widehat{b}_1 & \widehat{b}_2 \end{array} \quad \text{and} \quad \begin{array}{c|cc} 0 & & \\ \widetilde{c}_2 & \widetilde{a}_{21} & \\ \hline & \widetilde{b}_1 & \widetilde{b}_2 \end{array} .$$

Show that their composition

$$\Psi^{t,t+2h} := \tilde{\Psi}^{t+h,t+2h} \circ \hat{\Psi}^{t,t+h}$$

can be interpreted as a discrete evolution of a Runge-Kutta method with step size $2h$, and determine the coefficients of this Runge-Kutta method.

Solution: The first method $\hat{\Psi}$ yields an approximation

$$\mathbf{y}_h(t+h) := \mathbf{y}_h(t) + h\Big(\widehat{b}_1\widehat{k}_1 + \widehat{b}_2\widehat{k}_2\Big)$$

where

$$\begin{aligned} \widehat{k}_1 &= f(t, \mathbf{y}_h(t)), \\ \widehat{k}_2 &= f(t + \widehat{c}_2 h, \mathbf{y}_h(t) + h\widehat{a}_{21}\widehat{k}_1). \end{aligned}$$

Applying the second method $\tilde{\Psi}$, we obtain

$$\begin{aligned} \mathbf{y}_h(t+2h) &:= \tilde{\Psi}^{t+h,t+2h}\mathbf{y}_h(t+h) \\ &= \mathbf{y}_h(t+h) + h\Big(\widetilde{b}_1\widetilde{k}_1 + \widetilde{b}_2\widetilde{k}_2\Big) \\ &= \mathbf{y}_h(t) + 2h\Big(\tfrac{\widehat{b}_1}{2}\widehat{k}_1 + \tfrac{\widehat{b}_2}{2}\widehat{k}_2 + \tfrac{\widetilde{b}_1}{2}\widetilde{k}_1 + \tfrac{\widetilde{b}_2}{2}\widetilde{k}_2\Big) \end{aligned}$$

where the last line has been written in the form of a single step method with step size $2h$ and

$$\begin{aligned} \widetilde{k}_1 &:= f(t+h, \mathbf{y}_h(t+h)), \\ \widetilde{k}_2 &:= f(t+h+\widetilde{c}_2 h, \mathbf{y}_h(t+h) + h\widetilde{a}_{21}\widetilde{k}_1). \end{aligned}$$

Their composition is therefore again a Runge-Kutta method with step size $2h$ and the following Butcher tableau

$$\begin{array}{c|cccc} 0 & & & & \\ \widehat{c}_2/2 & \widehat{a}_{21}/2 & & & \\ 1/2 & \widehat{b}_1/2 & \widehat{b}_2/2 & & \\ (1+\widetilde{c}_2)/2 & \widehat{b}_1/2 & \widehat{b}_2/2 & \widetilde{a}_{21}/2 & \\ \hline & \widehat{b}_1/2 & \widehat{b}_2/2 & \widetilde{b}_1/2 & \widetilde{b}_2/2 \end{array} .$$

Problem 6.3 Newton's Method as a Single-Step Method

It is a surprising realisation that Newton's method for solving non-linear equation systems can be introduced as a single-step method to solve suitable initial value problems: The damped Newton method for solving $F(\mathbf{y}) = 0$, $F \in \mathcal{C}^2(D, \mathbb{R}^d)$, $D \subset \mathbb{R}^d$ is given as

$$\mathbf{y}_{k+1} = \mathbf{y}_k - s_k(\mathrm{D}F(\mathbf{y}_k))^{-1}F(\mathbf{y}_k), \tag{6.3.1}$$

where $\mathrm{D}F(\mathbf{y}_k)$ is the Jacobian of F and s_k is the damping parameter.

(6.3a) Interpret (6.3.1) as an explicit Euler method for a suitable autonomous ODE. What does the corresponding time grid look like?

Solution: The damped Newton method (6.3.1) can be interpreted as explicit Euler-approximation

$$\mathbf{y}_{k+1} = \mathbf{y}_k + s_k\mathbf{g}(\mathbf{y}_k)$$

for the differential equation

$$\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y}) := -(\mathrm{D}F(\mathbf{y}))^{-1}F(\mathbf{y}), \qquad \mathbf{y}(0) = \mathbf{y}_0,$$

where the step-size of the time grid $\mathcal{G}$ is equal to the damping parameter, i.e. $\mathcal{G} = \{t_k\} = \left\{\sum_{i=0}^{k-1} s_i\right\}$.

(6.3b) Show: If $F(\mathbf{y}^*) = 0$, then $\mathbf{y}^*$ is an attractive fixed point of the differential equation in subproblem (6.3a).

Definition: A fixed point $\mathbf{y}^*$ is called attractive, if there exists an $\epsilon > 0$, such that the solution $t \mapsto \mathbf{y}(t)$ of the initial value problem to every initial value $\mathbf{y}_0$ with $\|\mathbf{y}_0 - \mathbf{y}^*\| < \epsilon$ tends to $\mathbf{y}^*$ for $t \to \infty$.

HINT: Use following theorem:

Let $\mathbf{y}^* \in D$ be a fixed point of the autonomous differential equation $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$ with $\mathbf{f} \in C^1(D, \mathbb{R}^d)$. If the spectral abscissa of the Jacobian of $\mathbf{f}$ in the fixed point $\mathbf{y}^*$ is negative, meaning, if

$$\nu(D\mathbf{f}(\mathbf{y}^*)) := \max_{\lambda \in \sigma(D\mathbf{f}(\mathbf{y}^*))} Re(\lambda) < 0,$$

then $\mathbf{y}^*$ is an attractive fixed point.

Solution: Let $F(\mathbf{y}^*) = 0$. To show that $\mathbf{y}^*$ is an attractive fixed point of the differential equation $\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y})$, we observe the Jacobian

$$\begin{aligned} \mathrm{D}\mathbf{g}(\mathbf{y}^*) = &\quad \mathrm{D}\big(-(\mathrm{D}F(\mathbf{y}^*))^{-1}F(\mathbf{y}^*)\big) \\ &= -\mathrm{D}\big((\mathrm{D}F(\mathbf{y}^*))^{-1}\big)\underbrace{F(\mathbf{y}^*)}_{=0} - (\mathrm{D}F(\mathbf{y}^*))^{-1}\mathrm{D}F(\mathbf{y}^*) \\ &= -I. \end{aligned}$$

Because $\max_{\lambda \in \sigma(D\mathbf{g}(\mathbf{y}^*))} Re(\lambda) = -1 < 0$, the spectral abscissa of the Jacobian is negative. The result follows from the theorem in the hint.

Problem 6.4 A MATLAB Function for Implicit Runge-Kutta Methods

Gauss collocation methods are in general implicit methods, therefore a non-linear system of equations must be solved in each step. To this end we use a damped Newton-method (`dampnewton.m`). The construction of collocation methods from [NODE, Sect. 2.2.1] of the lecture gives us a piecewise polynomial approximation $\mathbf{y}_h$ of the solution of the initial value problem, which is given by the collocation conditions on each interval of the time grid, as

$$\mathbf{y}_h(t_k + \tau h) = \mathbf{y}_0 + h \sum_{j=1}^{s} \mathbf{k}_j \int_0^\tau L_j(\xi)\,\mathrm{d}\xi, \quad 0 \le \tau \le 1,$$

with increments $\mathbf{k}_j$.

(6.4a) Extend the function `rkimplss.m`, such that it returns the approximations $\mathbf{y}_h(kh)$ for a given Butcher scheme, the time grid $\{0, h, 2h, \ldots, 1\}$, $h = \frac{1}{N}$ and $k = 0, ..., N$ as well as the values $\mathbf{y}_h$ at the points $\{t_k + \tau_1 h, \ldots, t_k + \tau_l h\}$, $0 < \tau_1 \le \cdots \le \tau_l = 1$.

HINT: Extend `collcoeffs.m`, such that for given $\tau_1, \ldots, \tau_l$, the return value `b` has the components $b_{ji} = \int_0^{\tau_i} L_j(\xi)\,\mathrm{d}\xi$ and then modify `rkimplssm.m`.

Solution: For the modification of `collCoeffs.m`, see Listing 6.1.

For the modification of `rkimplssm.m`, see Listing 6.2.

Listing 6.1: Modified `collCoeffs.m`

```
function [A,b] = collCoeffs(c,tau)
% COLLCOEFFS Calculate the coefficients of RK-collocation
   methods.
%
%   [A,B] = COLLCOEFFS(C) given the collocation points in the
   vector C,
%   calculate the matrix A and vector B from the Butcher
   Table of the
%   corresponding Collocation Method.
%
% Example Usage, Calculate the Butcher Table of the Implicit
   Trapezium Method
% (collocation points c1=0, c2=1):
% [A,b] = collCoeffs([0 1])

  s = length(c);
  is = length(tau);
  % jth column are coefficiants of jth Lagrange-Polynom
  alpha = inv(vander(c));

  A = nan(s,s);
  b = nan(s,is);
  for j=1:s
    A(:,j) = polyval(polyint(alpha(:,j).'),c);
    b(j,:) = polyval(polyint(alpha(:,j).'),tau);
```

```
22    end
```

Listing 6.2: Modified `rkimplssm.m`

```
1  function [vals] = rkimplssm(fun, Jac, y0, N, c, b, A, tol)
2  % In implicit Runge-Kutta method  we need to solve a possibly
       nonlinear
3  % system of equations. Here we will use the damped Newton
       method.
4  % INPUTS:
5  % fun: Right hand side function
6  % Jac: Jacobian of fun
7  % y0: initial value
8  % N: number of steps
9  % c, b, A: Coefficients characterising the RK-Method
10 % tol: tolerance for the damped Newton
11
12 % Precheck input arguments for consistency
13 s = size(b, 1);
14 if ~isequal(size(A),[s s]), error('Size mismatch.
       Double-check A and b'); end
15 if (length(c) ~= s), error('c of wrong size'); end
16
17 % Initialise the variables
18 h = 1/N; % Stepsize
19 y = y0;  % Set to initial value
20 vals = y;
21 % Loop over all the points in the mesh
22 for k = 1:N
23     % Solve the equation fv(x) = 0
24     % As the initial guess use [0, ... , 0]
25     % Functions NFun and NJac are defined below
26     [gv, ~] = DampedNewton(zeros(s, 1), @NFun, @NJac, 0.5,
           tol);
27     % Check the definition of NFun to infer what does gv
           correspond to
28     yo = y(end);
29     % intial value for y
30     y = y(end);
31     % Compute (all entries of) y by doing one step of the
           given RK method
32     % (with s stages, s being the dimension of A).
33     % Notice that if b has more than one column then we ought
           to compute
34     % the intermediate values as well.
35      for i=1:s
36          y = y + h*feval(fun,yo+gv(i))*b(i,:);
37      end
```

```
38
39      % Notice that if the size of b is not of the form n x 1
           (that is, if
40      % the we are computing some intermediate values as well)
           but rather of
41      % the form n x r, then the y computed above will not be a
           scalar but a
42      % r x 1 vector. The last entry in y is the value you
           should use as yo
43      % in the next iteration. The other entries are
           intermediary values
44      vals = [vals, y];
45  end
46  % Definition of function NFunc
47      function fv = NFun(gv)
48          fv = gv;
49          for j=1:s
50              fg = feval(fun,y(end)+gv(j));
51              fv = fv - h*A(:,j)*fg;
52          end
53      end
54  % Definition of function NJac (Jacobi Matrix for NFunc)
55      function DF = NJac(gv)
56          DF = [];
57          for j=1:s
58              Jacj = feval(Jac,y(end)+gv(j));
59              KP = A(:,j);
60              DF = [DF,KP*Jacj];
61          end
62          DF = eye(s) - h*DF;
63      end
64  end
```

(6.4b) Use the modified functions from subproblem (6.4a) and numerically show that an s-step Gauss collocation method on an equidistant timegrid need not adhere to the error estimate

$$\epsilon(h) := \max_{0\leq t\leq T} \|\mathbf{y}_h(t) - \mathbf{y}(t)\| = \mathcal{O}(h^{2s}).$$

To do so, complete the template `GaussCollLogRate.m` in which we estimate the convergence rate of a collocation method of order s, with $s = 1, \ldots, 4$, by computing $\epsilon(h)$ on a grid with $N = 2^i$ points, where $i = 2, \ldots, 6$ (i.e. as $h \to 0$).

HINT: As an example, use the scalar logistic differential equation

$$\dot{y} = 10y(1 - y)$$

on $[0, 1]$ with $\mathbf{y}(0) = 0.01$.

Solution: See Listing 6.3 for the modification of `GaussCollLogRate.m`.

Listing 6.3: Modified `GaussCollLogRate.m`

```
function  GaussCollLogRate
% In this code we compute and plot the convergence rates of
   approximate
% solutions to the logistic IVP
%                    dy/dt = lambda*y*(1-y) , y(0) = 0.01
% acquired by RK collocation methods of order 1-4.

% Refinement steps, used for 2^[-2:nsteps]
nsteps=6;
% Source term
lambda=10;
% Right hand side of the ODE
fun = @(x) lambda*x.*(1-x);
% Jacobian of the right hand side
jac = @(x) lambda*(1-2*x);
% Initial value
y0 = 0.01;
% Exact solution of the differential equtation
sol = @(t) y0./(y0+(1-y0)*exp(-lambda.*t));

% Intermediate evaluation points
tau = [0.5,1];

% Initialising the error
err_t = zeros(4,nsteps-1);
tol=10^-6;
% Make the figure
figure('Name','Collocation single step methods');
hold on;
% Set markers and initialise the legend and the output
marker = {'b+','m*','c^','rp'};
leg = cell(4, 1);
results = cell(4, 1);

for p = 1:4
    % Get nodes for the appropriate collocation method
    [c] = GaussNodes(p-1);

    % Get the vector/matrix b and matrix A of the collocation
       method
    % Beware that b is of size p x length(tau)
    [A, b] = collCoeffs(c, tau);

    for i = 2:nsteps
        % Number of steps
        N = 2^i;
```

```
        % Time grid
        t = 0:1/(N*length(tau)):1;
        % Compute the solution using rkimplssm
        [vals] = rkimplssm(fun, jac, y0, N, c, b, A, tol);
        err_t(p, i-1) = norm(vals-sol(t), Inf);
    end
    plot(2.^-[2:nsteps], err_t(p, :), marker{p});
    % Compute the approximate order of convergence
    fit = polyfit(log(2.^-[2:nsteps]), log(err_t(p, :)), 1);
    % Assigning the legend and the output
    leg{p} = sprintf('s=%d',p);
    results{p} = sprintf('For s = %d the estimated order is
       %f', p, fit(1));

end
% Display the results
sprintf('%s\n', results{:})

% Final touches for the plot
set(gca,'fontsize',14);
set(gca,'Xscale','log','Yscale','log');
xlabel('t');
ylabel('y');
legend(leg, 4);

% Write to file
print -dpsc2 'rkkolllograte.eps';
```

We get the convergence rates $2.03\,(s=1), 4.06\,(s=2), 3.97\,(s=3), 5.89\,(s=4)$, i.e. for $s=3,4$, $\epsilon(h) \neq \mathcal{O}(h^{2s})$.

(6.4c) Prove that the $\mathbf{y}_h$ computed with the 1-step Gauss collocation method satisfy

$$\max_{0\leq t\leq T} \|\mathbf{y}_h(t) - \mathbf{y}(t)\| = \mathcal{O}(h^2).$$

Solution: We want to show that with $s=1$

$$\max_{0\leq t\leq T} \|\mathbf{y}_h(t) - \mathbf{y}(t)\| = \mathcal{O}(h^2).$$

We know that $\mathbf{y}(t)$ is a piecewise linear function, i.e. $\mathbf{y}_h|_{[t_k,t_{k+1}]}$ is linear. Now, for a given time grid, we define the interpolation operator I_h, with $(I_h\mathbf{y})(t_k) = \mathbf{y}(t_k)$ and $I_h\mathbf{y}|_{[t_k,t_{k+1}]}$ is linear. We

know that $\max_t \|I_h \mathbf{y}(t) - \mathbf{y}(t)\| = \mathcal{O}(h^2)$. Now, we have

$$\begin{aligned}
\max_{0\le t\le T} \|\mathbf{y}_h(t) - \mathbf{y}(t)\| &= \max_{0\le t\le T} \|\mathbf{y}_h(t) - I_h\mathbf{y}(t) + I_h\mathbf{y}(t) - \mathbf{y}(t)\| \\
&\le \max_{0\le t\le T} \|\mathbf{y}_h(t) - I_h\mathbf{y}(t)\| + \max_{0\le t\le T} \|I_h\mathbf{y}(t) - \mathbf{y}(t)\| \\
&= \max_{0\le t\le T} \|\mathbf{y}_h(t) - I_h\mathbf{y}(t)\| + \mathcal{O}(h^2) \\
&\le \max_{t_k} \|\mathbf{y}_h(t_k) - I_h\mathbf{y}(t_k)\| + \mathcal{O}(h^2) \\
&= \mathcal{O}(h^2).
\end{aligned}$$

Note that we use the fact that the maximum of a linear function on an interval is taken at its boundary. The last step is the consistency quality $\|\mathbf{y}_h(t_k) - \mathbf{y}(t_k)\| = \mathcal{O}(h^2)$ of the 1-step Gauss collocation method.

Published on 28 March 2016.

To be submitted by 5 April 2016.

References

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

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

Last modified on April 22, 2016