Problem Sheet 7

Problem 7.1 Extrapolation of the Implicit Mid-Point Rule.

Taking the implicit mid-point rule as a basis method, we generate another single step method for the autonomous initial value problem $\dot{\mathbf{y}} = f(\mathbf{y})$, $\mathbf{y}(0) = \mathbf{y}_0$ by extrapolation based on 1 and 2 micro-steps. The right hand side is assumed to be "sufficiently smooth".

(7.1a) Denote by $\mathbf{y}_h$, $\mathbf{y}_{h/2}$ the approximations for $\mathbf{y}(h)$ obtained by (successive) application of the implicit mid-point rule with step sizes h and $h/2$, respectively.

Express the approximate value for $\mathbf{y}(h)$ obtained from the extrapolation method in terms of $\mathbf{y}_{h/2}$ and $\mathbf{y}_h$.

HINT: Use the following theorem concerning the asymptotic expansion of the discretization error of single step methods:

Theorem. Let $\mathbf{y}_N$ be the approximate value of $\mathbf{y}(T)$ obtained by an application of $N := T/h$ steps of a single-step method of order p with step size $h > 0$. Here, $\mathbf{y}(t)$ denotes the solution of the initial value problem $\dot{\mathbf{y}} = f(\mathbf{y})$, $\mathbf{y}(t_0) = \mathbf{y}_0$.

Then there exist smooth functions $\mathbf{e}_i : [t_0, T] \mapsto \mathbb{R}^d$, $i = p, p+1, \ldots, p+k$, with $k \in \mathbb{N}$ determined by regularity of f, and a (for sufficiently small h) uniformly bounded function $(T, h) \mapsto \mathbf{r}_{k+p+1}(T, h)$, such that

$$\mathbf{y}_h - \mathbf{y}(T) = \sum_{l=0}^{k} \mathbf{e}_{l+p}(T) h^{l+p} + \mathbf{r}_{k+p+1}(T, h) h^{k+p+1} \quad \text{for } h \to 0 \, .$$

with $\mathbf{r}_{k+p+1}(T, h) = \mathcal{O}(T - t_0)$ for $T - t_0 \to 0$ uniformly in $h < T$, *where additionally* $\mathbf{e}_l(T) = \mathcal{O}(T - t_0)$ *for* $T \to t_0$.

Solution: For autonomous IVP's, the implicit mid-point rule is given by

$$\mathbf{y}_{k+1} = \mathbf{y}_k + hf\left(\frac{\mathbf{y}_k + \mathbf{y}_{k+1}}{2}\right).$$

We use that

$$\mathbf{y}_h - \mathbf{y}(T) = \sum_{l=0}^{k} \mathbf{e}_{l+p}(T) h^{l+p} + \mathbf{r}_{k+p+1}(T, h) h^{k+p+1},$$

and

$$\mathbf{y}_{h/2} - \mathbf{y}(T) = \sum_{l=0}^{k} \mathbf{e}_{l+p}(T) \left(\frac{h}{2}\right)^{l+p} + \mathbf{r}_{k+p+1}(T, h) \left(\frac{h}{2}\right)^{k+p+1}$$

to write the global error of the approximation with a step size h in the form

$$\mathbf{y}_h - \mathbf{y}(T) = \mathbf{e}_2(T)h^2 + \mathbf{e}_3(T)h^3 + \mathbf{r}_4(T,h)h^4 \tag{7.1.1}$$

and the global error of the approximation of two steps with step size $h/2$ in the form

$$\mathbf{y}_{h/2} - \mathbf{y}(T) = \mathbf{e}_2(T)\left(\frac{h}{2}\right)^2 + \mathbf{e}_3(T)\left(\frac{h}{2}\right)^3 + \mathbf{r}_4(T,h)\left(\frac{h}{2}\right)^4. \tag{7.1.2}$$

We see that the error term $\mathbf{e}_2(t)$ which is of order $\mathcal{O}(h^3)$ can be eliminated by combining the two expressions for the global error as (7.1.1) - 4·(7.1.2). We obtain

$$\mathbf{y}_h - 4\mathbf{y}_{h/2} + 3\mathbf{y}(T) = \frac{1}{2}\mathbf{e}_3(T)h^3 + \mathcal{O}(h^4),$$

so

$$\mathbf{y}(T) = -\frac{1}{3}\mathbf{y}_h + \frac{4}{3}\mathbf{y}_{h/2} + \mathcal{O}(h^4),$$

and hence the extrapolated method is

$$\Psi^{0,h} = -\frac{1}{3}\mathbf{y}_h + \frac{4}{3}\mathbf{y}_{h/2}.$$

(7.1b) What is the consistency order of the extrapolation method from subproblem (7.1a)? Justify your answer.

Solution: The extrapolated method has order 3 (cf. [NODE, Def. 2.1.13]).

(7.1c) Give the discrete evolution $\boldsymbol{\Psi}^h(y)$ of the implicit mid-point rule for the logistic differential equation

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

with initial value $y(0) = y_0 > 0$ in closed form.

HINT: The discrete evolution of the implicit mid-point rule leads to a quadratic equation which can be solved explicitely. Try to solve y_{k+1} as function of y_k out of implicit step function $y_{k+1} = g(y_k, y_{k+1})$.

Solution: We apply the implicit mid-point rule to the logistic equation and obtain

$$y_{k+1} = y_k + h\lambda\frac{1}{2}(y_k + y_{k+1})\left(1 - \frac{1}{2}(y_k + y_{k+1})\right).$$

Define $x = (y_k + y_{k+1})/2$. We have

$$y_{k+1} + y_k = 2y_k + h\lambda x(1-x)$$

and hence

$$x = y_k + h\lambda x(1-x).$$

This is a quadratic polynomial in x

$$\frac{h\lambda}{2}x^2 + \left(1 - \frac{h\lambda}{2}\right)x - y_k = 0$$

with solution

$$x = \frac{-\left(1 - \frac{h\lambda}{2}\right) \pm \sqrt{\left(1 - \frac{h\lambda}{2}\right)^2 + 2h\lambda y_k}}{h\lambda}.$$

(7.1d) Which of the two solutions from subproblem (7.1c) is admissible? Justify your answer. HINT: For $f : X \to X$, X be a topological space, define $f^{(n)} = f^{(n-1)}(f(x))$. We say a fixed point x_0 of f is repulsive relative to U, which is an open neighborhood of x_0, if for any open neighborhood V of x_0, there exists a positive integer n_0 such that $f^{(n)}(X - V) \subset X - U$ whenever $n \leqslant n_0$.

Solution: Only the solution with positive sign

$$y_{k+1} = -y_k + 2\frac{-\left(1 - \frac{h\lambda}{2}\right) + \sqrt{\left(1 - \frac{h\lambda}{2}\right)^2 + 2h\lambda y_k}}{h\lambda} > 0$$

is admissible, since $y_0 > 0$ and the logistic equation has an attractive fixed point at $y = 1$ and a repulsive fixed point at $y = 0$ (cf. [NODE, Sect. 1.2.1][NUMODE, Ex. 1.2.1]).

(7.1e) Complete the MATLAB template `IMPExtrapLog.m`, which is supposed to implement the extrapolated mid-point rule for the initial value problem from subproblem (7.1c).

Plot both your numerical result for the time interval $[0, 1]$ for the parameter value $\lambda = 10$ and initial value $y(0) = 0.2$ as well as the exact solution

$$y(t) = \frac{1}{1 + (y_0^{-1} - 1)e^{-\lambda t}}$$

in a figure and save the plot in `logistic.eps`.

Solution: See Figure 7.1. For the code, see `IMPExtrapLog_sol.m`, Listing 7.1 and `PlotLogistic_sol.m` Listing 7.2.

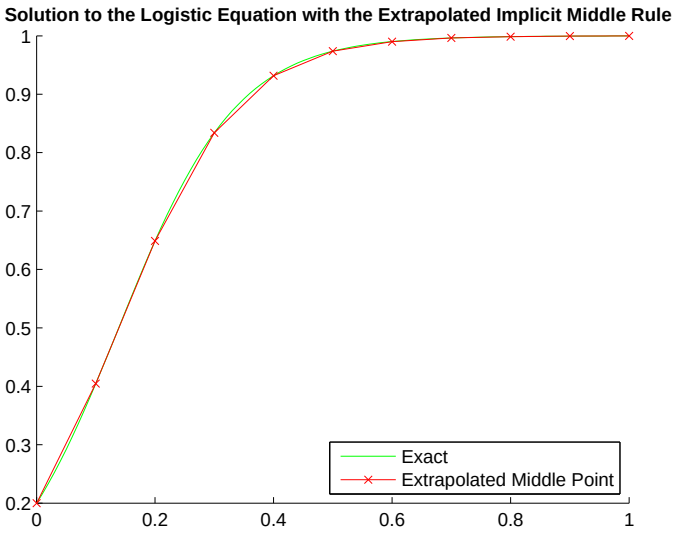

Figure 7.1: subproblem (7.1e).

Listing 7.1: Implementation of `IMPExtrapLog_sol.m`

```
1 extrapconvfunction [t,y,h] = IMPExtrapLog(h,y0,T,lambda)
2 % IMPEXTRAPLOG compute the numerical solution of the logistic
3 % differential equation with an extrapolated mid-point rule.
```

```
%
%   Name:
%
% [T,Y,H] = IMPEXTRAPLOG(H,Y0,T,LAMBDA) solve the logistic differential
% equation y' = LAMBDA*y(1-y) on the time interval T = [t0,tend] with
% initial value y(t0) = y0 with an extrapolated mid-point rule and step
% size (at most) H. As output, the time grid T, the solution Y and (possibly
% altered) step size are returned. The returned value of the step size is
% necessary for the convergence analysis.
%
% Example:
% [t,y,h] = IMPExtrapLog(0.05,0.02,[0,1],10);
%
% See also IMPEXTRAPLOGCONV.

  N = ceil((T(2)-T(1))/h); % number of steps
  h = (T(2)-T(1))/N; % adjust step size to fit interval length
  t = linspace(T(1),T(2),N+1); % time grid
  y = nan(1,N+1); % solution
  y(1) = y0; % set initial value

  for j=1:N

    % Solution at t(j)+h computed with a step of size h
    y1 = IMPLogStep(y(j),h,lambda);

    % Solution at t(j)+h computed with a step of size h/2
    y2 = IMPLogStep(y(j),h/2,lambda);
    y2 = IMPLogStep(y2,h/2,lambda);

    % Extrapolated solution
    y(j+1) = -y1/3 + 4*y2/3;

  end

  return

function y1 = IMPLogStep(y0,h,lambda)
  % IMPLOGSTEP compute the solution Y1 of the logistic differential
  % equation with parameter value LAMBDA after a step H starting
```

```
  % at the initial value Y0 using the implicit mid-point rule

  z = h*lambda;
  y1 = -y0 + 2*( -(1-z/2) + sqrt( (1-z/2)^2 + 2*z*y0 ) )/z;

  return
```

Listing 7.2: Implementation of `PlotLogistic_sol.m`

```
function PlotLogistic

  h=0.1;

  y0=0.2;
  T=[0 1];

  lambda=10;

  tExact = [T(1):0.01:T(2)];
  yExact = 1./( 1 + (y0^-1 - 1)*exp(-lambda*tExact) );

  [t,y] = IMPExtrapLog_sol(h, y0, T, lambda);

  figure;
  hold on;
  plot(tExact,yExact,'g');
  plot(t,y,'x-r');
  legend('Exact','Extrapolated Middle
     Point','Location','SouthEast');
  title('\bf Solution to the Logistic Equation with the
     Extrapolated Implicit Middle Rule');
```

(7.1f) Complete the MATLAB template `IMPExtrapLogConv.m`, to determine the convergence order of the extrapolation method for the logistic differential equation empirically. For this, use the initial value problem from subproblem (7.1e) and the step size $h = 1/2^n$, $n = 4, 5, \ldots, 9$. Save the relevant plot in the file `conv.eps`.

Solution: See Figure 7.2. For the code, see `IMPExtrapLogConv_sol.m`, Listing 7.3. From the plot, we can read off the empirical convergence rate 4.

Listing 7.3: Implementation of `IMPExtrapLogConv_sol.m`

```
function fit = IMPExtrapLogConv_sol
% Compute the empirical convergence order of the extrapolated
   mid-point
% rule (IMPExtrapLog) for the logistic differential equation
% y' = lambda*y*(1-y).
%
% See also IMPEXTRAPLOG.
```

```
7
8   lambda = 10;
9   y0 = [0.2];
10  T = [0 0.2];
11  h = 2.^-[4:9];
12
13  err = nan(size(h));
14
15  for j=1:length(h)
16    [t,y,h(j)] = IMPExtrapLog_sol(h(j), y0, T, lambda);
17    err(j) = norm( y(end) - sol(t(end),y0,lambda) );
18  end
19
20  figure;
21  loglog(h,err,'bx:')
22  grid on;
23  title('\bf log--log plot of error vs. step size');
24  xlabel('h');
25  ylabel('err');
26
27  fit = polyfit(log(h),log(err),1);
28  fprintf('numerically determined convergence order:
      %f\n',fit(1));
29
30  return
31
32 function y = sol(t,y0,lambda)
33 % Exact solution for y' = LAMBDA*y*(1-y) with starting value
     at time(s) T.
34
35  y = 1./( 1 + (y0^-1 - 1)*exp(-lambda*t) );
36
37  return
```

This is better than the order, which we obtained in subproblem (7.1a), since the implicit mid-point rule, a symmetric method, has an asymptotic expansion in even powers of h. See *Hairer, Wanner* S.222 II.8 *Asymptotic Expansion of the Global Error*.

Problem 7.2 Extrapolation methods as Runge-Kutta methods

Extrapolation methods based on explicit methods can be written as explicit Runge–Kutta methods. Here, we will choose the explicit Euler method as our base method. Perform an extrapolation step with step size $h_1 = 2\,h_2$. Then, write the extrapolated method as a Runge–Kutta method (Butcher-tableau) and determine its order of consistency.

HINT: Use the Aitken-Neville algorithm [NUMODE, Eq. (2.4.5)],[NUMODE, Eq. (2.4.6)],[NUMODE, Sect. 2.4.2].

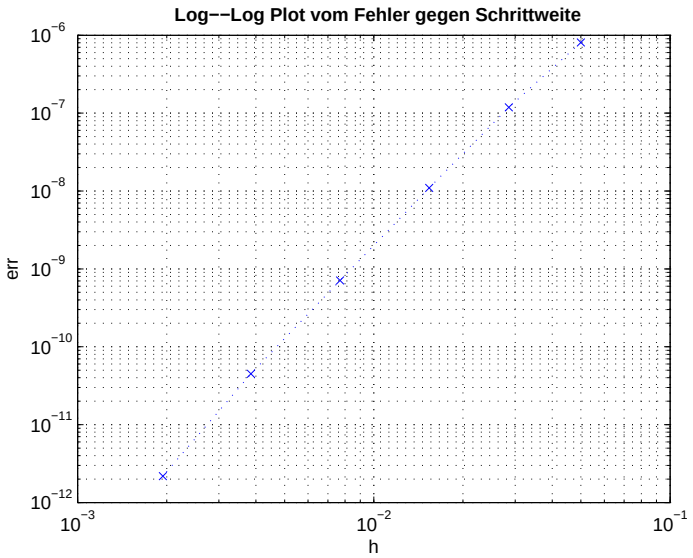

Figure 7.2: subproblem (7.1f).

Solution:

The explicit Euler method is given by

$$\mathbf{y}_1 = \mathbf{y}_0 + h\mathbf{f}(t_0, \mathbf{y}_0).$$

With the Aitken-Neville algorithm we get

$$\begin{aligned} T_{11} &= \mathbf{y}_0 + h_1\mathbf{f}(t_0, \mathbf{y}_0), \\ T_{21} &= \mathbf{y}_0 + \frac{h_1}{2}\mathbf{f}(t_0, \mathbf{y}_0) + \frac{h_1}{2}\mathbf{f}\left(t_0 + \frac{h_1}{2}, \mathbf{y}_0 + \frac{h_1}{2}\mathbf{f}(t_0, \mathbf{y}_0)\right). \end{aligned}$$

The extrapolated method is then given by

$$\begin{aligned} T_{22} &= T_{21} + \frac{T_{21} - T_{11}}{h_1/h_2 - 1}, \\ &= 2T_{21} - T_{11}, \\ &= \mathbf{y}_0 + h_1\mathbf{f}\left(t_0 + \frac{h_1}{2}, \mathbf{y}_0 + \frac{h_1}{2}\mathbf{f}(t_0, \mathbf{y}_0)\right). \end{aligned}$$

Its Butcher-tableau is

0	0	0
0.5	0.5	0
	0	1

A Runge-Kutta method of this form is an autonomisation invariant [NODE, Lemma. 2.2.14], as $\sum_{i=1}^{s} b_i = 1$, and has a consistency order of at least 2, as $\sum_{i=1}^{s} b_i c_i = 0.5$ (cf. [NODE, Def. 2.3.3]). This is as expected, since k steps of extrapolation raise the consistency order by k.

Problem 7.3 Low Storage Runge-Kutta Method

A low storage Runge-Kutta method of order n is an algorithm for the solution of the autonomous IVP $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$, $\mathbf{y}(0) = \mathbf{y}_0$ of the form

$\mathbf{x}_0 = \mathbf{y}_0$
$\mathbf{q}_0 = 0$
for $j = 1$ **to** n **do**
  $\mathbf{q}_j = \alpha_j \mathbf{q}_{j-1} + h\mathbf{f}(\mathbf{x}_{j-1})$
  $\mathbf{x}_j = \mathbf{x}_{j-1} + \beta_j \mathbf{q}_j$
end for
$\mathbf{y}_1 = \mathbf{x}_n$

The algorithm is described by the real coefficients $(\alpha_j)_{j=1}^n$, $(\beta_j)_{j=1}^n$, where $\alpha_1 = 0$.

(7.3a) Show that the 2-stage explicit autonomization invariant Runge-Kutta methods of order 2 form a one-parameter family. Determine the corresponding Butcher-Tableau as a function of this parameter.

HINT: Prove that all coefficients of Butcher tableau are functions of the same single parameter.

Solution: The Butcher Tableau of a 2-stage explicit autonomization invariant Runge-Kutta method has the form

$$\begin{array}{c|cc} 0 & 0 & 0 \\ c_2 & a_{21} & 0 \\ \hline & b_1 & b_2 \end{array}$$

It follows from the definition of autonomization invariant that $c_2 = a_{21}$. For 2-stage methods, the necessary and sufficient conditions on Runge-Kutta methods of order 2.

$$\begin{aligned} b_1 + b_2 &= 1, \\ b_2 c_2 &= \frac{1}{2}, \end{aligned}$$

from which it follows that

$$\begin{aligned} b_2 &= \frac{1}{2a_{21}}, \\ b_1 &= 1 - \frac{1}{2a_{21}}. \end{aligned}$$

Note that $b_2 \neq 0$ and $c_2 \neq 0$.

(7.3b) Show that the stability function of any 2-stage explicit autonomization invariant Runge-Kutta method of order 2 satisfies

$$S(z) = 1 + z + \frac{z^2}{2}.$$

Solution: From the definition of the stability function

$$\begin{aligned}
S(z) &:= 1 + z\mathbf{b}^\top(\mathbf{I} - z\mathfrak{A})^{-1}\mathbf{1} \\
&= 1 + z(b_1\ b_2)\begin{pmatrix} 1 & 0 \\ -a_{21}z & 1 \end{pmatrix}^{-1}\begin{pmatrix}1\\1\end{pmatrix} \\
&= 1 + z(b_1\ b_2)\begin{pmatrix} 1 & 0 \\ a_{21}z & 1 \end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix} \\
&= 1 + z(b_1\ b_2)\begin{pmatrix} 1 \\ 1 + a_{21}z \end{pmatrix} \\
&= 1 + z(\underbrace{b_1 + b_2}_{=1} + \underbrace{b_2 a_{21}}_{=1/2} z) \\
&= 1 + z + \frac{1}{2}z^2.
\end{aligned}$$

Alternatively note that the stability function is a polynomial which approximates $\exp(z)$ with $\mathcal{O}(z^3)$ as $z \to 0$.

(7.3c) Show that every 2-stage explicit autonomization invariant Runge-Kutta method of order 2 is a 2-stage low storage Runge-Kutta method.

Solution: Developing the algorithm, we obtain

$$\begin{aligned}
\mathbf{q}_1 &= h\mathbf{f}(\mathbf{y}_0) \\
\mathbf{x}_1 &= \mathbf{y}_0 + \beta_1 h\mathbf{f}(\mathbf{y}_0) \\
\mathbf{q}_2 &= \alpha_2 h\mathbf{f}(\mathbf{y}_0) + h\mathbf{f}(\mathbf{y}_0 + \beta_1 h\mathbf{f}(\mathbf{y}_0)) \\
\mathbf{x}_2 &= \mathbf{y}_0 + \beta_1 h\mathbf{f}(\mathbf{y}_0) + \beta_2\alpha_2 h\mathbf{f}(\mathbf{y}_0) + \beta_2 h\mathbf{f}(\mathbf{y}_0 + \beta_1 h\mathbf{f}(\mathbf{y}_0)) \\
&= \mathbf{y}_0 + (\beta_1 + \beta_2\alpha_2)h\mathbf{f}(\mathbf{y}_0) + \beta_2 h\mathbf{f}(\mathbf{y}_0 + \beta_1 h\mathbf{f}(\mathbf{y}_0)).
\end{aligned}$$

Comparison with the scheme for a 2-stage explicit Runge-Kutta method

$$\begin{cases}
\mathbf{k}_1 = \mathbf{f}(\mathbf{y}_0) \\
\mathbf{k}_2 = \mathbf{f}(\mathbf{y}_0 + a_{21}h\mathbf{k}_1) \\
\mathbf{y}_1 = \mathbf{y}_0 + hb_1\mathbf{k}_1 + hb_2\mathbf{k}_2
\end{cases}$$

yields

$$\begin{cases}
b_1 = \beta_1 + \beta_2\alpha_2 \\
b_2 = \beta_2 \\
a_{21} = \beta_1
\end{cases}$$

hence

$$\begin{cases}
\beta_1 = a_{21} \\
\beta_2 = b_2 \\
\alpha_2 = \frac{b_1 - a_{21}}{b_2}
\end{cases}$$

Note that $b_2 \neq 0$, since the Runge-Kutta method is of order 2.

(7.3d) A n-stage low storage Runge-Kutta autonomization invariant method is also a Runge-Kutta method in the sense of [NODE, Def. 2.2.2]. Prove this fact for the special case of a 3-stage low storage Runge-Kutta method by determination of the Butcher-Tableau of the corresponding Runge-Kutta method.

Solution: We continue the development from subproblem (7.3c)

$$\begin{aligned}\mathbf{q}_3 &= \alpha_3\alpha_2 h\mathbf{f}(\mathbf{y}_0) + \alpha_3 h\mathbf{f}(\mathbf{y}_0 + \beta_1 h\mathbf{f}(\mathbf{y}_0)) \\ &\quad + h\mathbf{f}(\mathbf{y}_0 + (\beta_1 + \beta_2\alpha_2)h\mathbf{f}(\mathbf{y}_0) + \beta_2 h\mathbf{f}(\mathbf{y}_0 + \beta_1 h\mathbf{f}(\mathbf{y}_0))) \\ \mathbf{x}_3 &= \mathbf{y}_0 + (\beta_1 + \beta_2\alpha_2 + \beta_3\alpha_3\alpha_2)h\mathbf{f}(\mathbf{y}_0) + (\beta_2 + \alpha_3\beta_3)h\mathbf{f}(\mathbf{y}_0 + \beta_1 h\mathbf{f}(\mathbf{y}_0)) \\ &\quad + \beta_3 h\mathbf{f}(\mathbf{y}_0 + (\beta_1 + \beta_2\alpha_2)h\mathbf{f}(\mathbf{y}_0) + \beta_2 h\mathbf{f}(\mathbf{y}_0 + \beta_1 h\mathbf{f}(\mathbf{y}_0)))\end{aligned}$$

and read off

$$\begin{aligned}b_1 &= \beta_1 + \beta_2\alpha_2 + \beta_3\alpha_3\alpha_2 \\ b_2 &= \beta_2 + \alpha_3\beta_3 \\ b_3 &= \beta_3,\end{aligned}$$

as well as

$$\begin{aligned}a_{21} &= \beta_1 \\ a_{31} &= \beta_1 + \alpha_2\beta_2 \\ a_{32} &= \beta_2.\end{aligned}$$

This implies

$$\begin{aligned}c_1 &= 0 \\ c_2 &= \beta_1 \\ c_3 &= \beta_1 + \alpha_2\beta_2 + \beta_2.\end{aligned}$$

Thus

0	0	0	0
β_1	β_1	0	0
$\beta_1 + \alpha_2\beta_2 + \beta_2$	$\beta_1 + \alpha_2\beta_2$	β_2	0
	$\beta_1 + \beta_2\alpha_2 + \beta_3\alpha_3\alpha_2$	$\beta_2 + \alpha_3\beta_3$	β_3

Published on 11 April 2016.
To be submitted by 19 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 May 24, 2016