Research Paper
Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. 10, No. 1, 2022, pp. 44-60
DOI:10.22034/cmde.2020.39203.1720

An interval version of the Kuntzmann-Butcher method for solving the initial value problem

Andrzej Marciniak[1,2], **Barbara Szyszka**[3,*], **and Tomasz Hoffmann**[4]
[1]Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60-965 Poznan, Poland.
[2]Department of Computer Science, Higher Vocational State School in Kalisz Poznanska 201-205, 62-800 Kalisz, Poland.
[3]Institute of Mathematics, Poznan University of Technology, Piotrowo 3A, 60-965 Poznan, Poland.
[4]Poznan Supercomputing and Networking Center, Jana Pawła II 10, 61-139 Poznan, Poland.

Abstract

The Kutzmann-Butcher method is the unique implicit four-stage Runge-Kutta method of order 8. In many problems in ordinary differential equations this method realized in floating-point arithmetic gives quite good approximations to the exact solutions, but the results obtained do not contain any information on rounding errors, representation errors and the error of the method. Thus, we describe an interval version of this method, which realized in floating-point interval arithmetic gives approximations (enclosures in the form of an interval) containing all these errors. The described method can also include data uncertainties in the intervals obtained.

Keywords. Initial value problem, Runge-Kutta methods, Kuntzmann-Butcher method, Interval Runge-Kutta methods, Floating-point interval arithmetic.

2010 Mathematics Subject Classification. 65G30, 65G40, 65L05, 65L06, 65Y04.

1. Introduction

It is well-known that there are two kinds of errors caused by floating-point arithmetic: representation errors and rounding errors. While solving ordinary differential equations on a computer (in the form of an initial value problem), we usually apply approximate methods, which in turn introduce the third kind of errors – the errors of methods, usually called truncation errors. To take into consideration these errors we can use interval arithmetic (see, e.g., [1, 15, 31, 32, 37]). Applying interval methods for solving the initial value problem in floating-point interval arithmetic (see, e.g., [14]), we can obtain enclosures of the solutions in the form of intervals which contain all possible numerical errors. These intervals may also include data uncertainties.

There are a number of interval methods for approximating the initial value problem. The first one was described by R. E. Moore in 1965 [30, 31]. There are also known interval methods based on high-order Taylor series (see, e.g., [2–4, 7, 16, 21, 34–36]), explicit Runge-Kutta methods [20, 25, 37], implicit ones [10, 11, 22, 25, 29], and explicit and implicit multistep methods [17–19, 23–25, 28, 37]. In recent years many studies have been conducted especially on a variety of the interval method based on high-order Taylor series.

In this paper we propose an interval version of the Kuntzmann-Butcher method, which is implicit and of high order. The main purpose to consider this method separately is the fact that this is the only one fourth-stage and eight-order method between other fourth-stage methods of Runge-Kutta type (interval versions of which we developed in our previous papers). Our numerical experiments show also that this method is at least comparable with the methods based on the high-order Taylor series, giving even better enclosures of the exact solutions.

The paper is divided into six sections. In section 2 we recall the well-known conventional implicit Runge-Kutta methods. The Kutzmann-Butcher method of order 8 is recalled in section 3. Section 4 is the main section of this

Received: 15 April 2020 ; Accepted: 12 December 2020.
∗ Corresponding author. Email: Barbara.Szyszka@put.poznan.pl.

paper, in which we describe an interval version of the Kutzmann-Butcher method. In this section we also point to some important theorems proved in our previous papers. In section 5 we present six numerical examples, which confirm the usefulness of the proposed method. We compare our results with those obtained by the VNODE-LP package based on high-order Taylor series [33, 34]. In the last section, some conclusions are given.

2. Implicit Runge-Kutta methods

Let us consider the initial value problem

$$y' = f(t, y(t)), \quad y(0) = y_0, \tag{2.1}$$

where $t \in [0, a]$, $y \in \mathbb{R}$ and $f : [0, a] \times \mathbb{R} \to \mathbb{R}$. The implicit m-stage Runge-Kutta methods for solving the problem (2.1) are given by the formula [5, 6, 12, 13]

$$y_{k+1} = y_k + h \sum_{i=1}^{m} w_i \kappa_{ik}, \quad k = 0, 1, \ldots,$$

where

$$\kappa_{ik} = f(t_k + c_i h, y_k + h \sum_{j=1}^{m} a_{ij} \kappa_{jk}), \quad i = 1, 2, \ldots, m, \tag{2.2}$$

and

$$c_i = \sum_{j=1}^{m} a_{ij},$$

where $h = t_{k+1} - t_k$ is a step-size, and the coefficients w_i, c_i and a_{ij} are some parameters. It is convenient to present these coefficients in a form of an array, called the Butcher table [6]:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1m} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ c_m & a_{m1} & a_{m2} & \cdots & a_{mm} \\ \hline & w_1 & w_2 & \cdots & w_m \end{array}$$

The local truncation error of step $k + 1$ for any Runge-Kutta method of order p can be written in the form

$$\begin{aligned} r_{k+1}(h) &= y(t_k + h) - \left(y(t_k) + h \sum_{i=1}^{m} w_i \kappa_{ik}(h) \right) \\ &= \psi(t_k, y(t_k)) h^{p+1} + O(h^{p+2}) \\ &= r_{k+1}^{(p+1)}(0) \frac{h^{p+1}}{(p+1)!} + r_{k+1}^{(p+2)}(\theta h) \frac{h^{p+2}}{(p+2)!}, \quad 0 < \theta < 1, \end{aligned}$$

where $y(t_k + h)$ and $y(t_k)$ denote the exact solutions at $t_k + h$ and t_k, respectively, and $\kappa_{ik}(h) \equiv \kappa_{ik}$ is given by (2.2) for the exact value $y(t_k)$. From the conditions $r_{k+1}^{(l)} = 0$ (for $l = 1, 2, ..., p$) follow the equations for determining the coefficients w_i, c_i and a_{ij}. There are fewer equations than the number of unknowns and usually we consider some special cases. For each m there exists a method with maximum order $p = 2m$.

3. The Kuntzmann-Butcher method of order 8

The Kuntzmann-Butcher method is the implicit four-stage Runge-Kutta method with $p = 8$. This method can be written in the form (compare the formulas (4.2) and (4.3) in Sec.4)

$$\begin{aligned} y_{k+1} &= y_k + h(w_1\kappa_{1k} + w_2\kappa_{2k} + w_3\kappa_{3k} + w_4\kappa_{4k}),\\ \kappa_{ik} &= f(t_k + c_i h, y_k + h(a_{i1}\kappa_{1k} + a_{i2}\kappa_{2k} + a_{i3}\kappa_{3k} + a_{i4}\kappa_{4k})),\\ c_i &= a_{i1} + a_{i2} + a_{i3} + a_{i4}, \quad i = 1, 2, 3, 4, \end{aligned} \tag{3.1}$$

where the coefficients are as follows [12]:

$$\begin{array}{c|cccc} \frac{1}{2} - \omega_2 & \omega_1 & \omega_1' - \omega_3 + \omega_4' & \omega_1' - \omega_3 - \omega_4' & \omega_1 - \omega_5 \\ \frac{1}{2} - \omega_2' & \omega_1 - \omega_3' + \omega_4 & \omega_1' & \omega_1' - \omega_5' & \omega_1 - \omega_3' - \omega_4 \\ \frac{1}{2} + \omega_2' & \omega_1 + \omega_3' + \omega_4 & \omega_1' + \omega_5' & \omega_1' & \omega_1 + \omega_3' - \omega_4 \\ \frac{1}{2} + \omega_2 & \omega_1 + \omega_5 & \omega_1' + \omega_3 + \omega_4' & \omega_1' + \omega_3 - \omega_4' & \omega_1 \\ \hline & 2\omega_1 & 2\omega_1' & 2\omega_1' & 2\omega_1 \end{array}$$

and where

$$\begin{aligned} \omega_1 &= \tfrac{1}{8}\left(1 - \tfrac{\sqrt{30}}{18}\right), & \omega_1' &= \tfrac{1}{8}\left(1 + \tfrac{\sqrt{30}}{18}\right),\\ \omega_2 &= \tfrac{1}{2}\sqrt{\tfrac{15+2\sqrt{30}}{35}}, & \omega_2' &= \tfrac{1}{2}\sqrt{\tfrac{15-2\sqrt{30}}{35}},\\ \omega_3 &= \tfrac{\omega_2}{6}\left(1 + \tfrac{\sqrt{30}}{4}\right), & \omega_3' &= \tfrac{\omega_2'}{6}\left(1 - \tfrac{\sqrt{30}}{4}\right),\\ \omega_4 &= \tfrac{\omega_2}{21}\left(1 + \tfrac{5\sqrt{30}}{8}\right), & \omega_4' &= \tfrac{\omega_2'}{21}\left(1 - \tfrac{5\sqrt{30}}{8}\right),\\ \omega_5 &= \omega_2 - 2\omega_3, & \omega_5' &= \omega_2' - 2\omega_3'. \end{aligned}$$

The local truncation error is

$$r_{k+1}(h) = \psi(t_k, y(t_k))h^9 + O(h^{10}). \tag{3.2}$$

The form of $\psi(t, y)$ is very complicated and cannot be written in a general form for an arbitrary p. Since this form is very important from the point of view of the interval method developed in the next section, below we present some useful formulas for $p = 8$ (in general, m can be equal not only to 4, as in our case, but also to 5, 6, 7 or 8).

To simplify further notation, we denote

$$f = f(t, y), \quad f^{(l)}_{t^p y^q} = \frac{\partial^l f}{\partial t^p \partial y^q},$$

where $l = p + q$, and

$$y^{(l)} = y^{(l)}(t), \quad \kappa_i^{(l)} = \kappa_i^{(l)}(0), \quad \lambda_i^{(l)} = \sum_{j=1}^{m} a_{ij}\kappa_j^{(l)},$$

where

$$\kappa_i \equiv \kappa_i(h) = f\left(t + c_i h, y(t) + h\sum_{j=1}^{m} a_{ij}\kappa_j(h)\right).$$

We have

$$\psi(t, y) = \frac{1}{9!}\left(y^{(9)} - 9\sum_{i=1}^{m} w_i\kappa_i^{(8)}\right), \tag{3.3}$$

where

$$
\begin{aligned}
\kappa_i^{(1)} =& c_i\left(f_t^{(1)}+f_y^{(1)}f\right),\\
\kappa_i^{(2)} =& c_i^2\left(f_{t^2}^{(2)}+2f_{ty}^{(2)}f+f_{y^2}^{(2)}f^2\right)+2f_y^{(1)}\lambda_i^{(1)},\\
\kappa_i^{(3)} =& c_i^3\left(f_{t^3}^{(3)}+3f_{t^2y}^{(3)}f+3f_{ty^2}^{(3)}f^2+f_{y^3}^{(3)}f^3\right)+6c_i\left(f_{ty}^{(2)}+f_{y^2}^{(2)}f\right)\lambda_i^{(1)}+3f_y^{(1)}\lambda_i^{(2)},\\
\kappa_i^{(4)} =& c_i^4\left(f_{t^4}^{(4)}+4f_{t^3y}^{(4)}f+6f_{t^2y^2}^{(4)}f^2+4f_{ty^3}^{(4)}f^3+f_{y^4}^{(4)}f^4\right)\\
&+12c_i^2\left(f_{t^2y}^{(3)}+2f_{ty^2}^{(3)}f+f_{y^3}^{(3)}f^2\right)\lambda_i^{(1)}+12c_i\left(f_{ty}^{(2)}+f_{y^2}^{(2)}f\right)\lambda_i^{(2)}\\
&+12f_{y^2}^{(2)}\left(\lambda_i^{(1)}\right)^2+4f_y^{(1)}\lambda_i^{(3)},\\
\kappa_i^{(5)} =& c_i^5\left(f_{t^5}^{(5)}+5f_{t^4y}^{(5)}f+10f_{t^3y^2}^{(5)}f^2+10f_{t^2y^3}^{(5)}f^3+5f_{ty^4}^{(5)}f^4+f_{y^5}^{(5)}f^5\right)\\
&+20c_i^3\left(f_{t^3y}^{(4)}+3f_{t^2y^2}^{(4)}f+3f_{ty^3}^{(4)}f^2+f_{y^4}^{(4)}f^3\right)\lambda_i^{(1)}\\
&+30c_i^2\left(f_{t^2y}^{(3)}+2f_{ty^2}^{(3)}f+f_{y^3}^{(3)}f^2\right)\lambda_i^{(2)}\\
&+60c_i\left(f_{ty^2}^{(3)}+f_{y^3}^{(3)}f\right)\left(\lambda_i^{(1)}\right)^2\\
&+20c_i\left(f_{ty}^{(2)}+f_{y^2}^{(2)}f\right)\lambda_i^{(3)}+60f_{y^2}^{(2)}\lambda_i^{(1)}\lambda_i^{(2)}+5f_y^{(1)}\lambda_i^{(4)},
\end{aligned}
$$

$$
\begin{aligned}
\kappa_i^{(6)} =& c_i^6\left(f_{t^6}^{(6)}+6f_{t^5y}^{(6)}f+15f_{t^4y^2}^{(6)}f^2+20f_{t^3y^3}^{(6)}f^3+15f_{t^2y^4}^{(6)}f^4+6f_{ty^5}^{(6)}f^5+f_{y^6}^{(6)}f^6\right)\\
&+30c_i^4\left(f_{t^4y}^{(5)}+4f_{t^3y^2}^{(5)}f+6f_{t^2y^3}^{(5)}f^2+4f_{ty^4}^{(5)}f^3+f_{y^5}^{(5)}f^4\right)\lambda_i^{(1)}\\
&+60c_i^3\left(f_{t^3y}^{(4)}+3f_{t^2y^2}^{(4)}f+3f_{ty^3}^{(4)}f^2+f_{y^4}^{(4)}f^3\right)\lambda_i^{(2)}\\
&+180c_i^2\left(f_{t^2y^2}^{(4)}+2f_{ty^3}^{(4)}f+f_{y^4}^{(4)}f^2\right)\left(\lambda_i^{(1)}\right)^2\\
&+60c_i^2\left(f_{t^2y}^{(3)}+2f_{ty^2}^{(3)}f+f_{y^3}^{(3)}f^2\right)\lambda_i^{(3)}\\
&+360c_i\left(f_{ty^2}^{(3)}+f_{y^3}^{(3)}f\right)\lambda_i^{(1)}\lambda_i^{(2)}\\
&+120f_{y^3}^{(3)}\left(\lambda_i^{(1)}\right)^3+30c_i\left(f_{ty}^{(2)}+f_{y^2}^{(2)}f\right)\lambda_i^{(4)}\\
&+30f_{y^2}^{(2)}\left(4\lambda_i^{(1)}\lambda_i^{(3)}+3\left(\lambda_i^{(2)}\right)^2\right)+6f_y^{(1)}\lambda_i^{(5)},\\
\kappa_i^{(7)} =& c_i^7\left(f_{t^7}^{(7)}+7f_{t^6y}^{(7)}f+21f_{t^5y^2}^{(7)}f^2+35f_{t^4y^3}^{(7)}f^3+35f_{t^3y^4}^{(7)}f^4+21f_{t^2y^5}^{(7)}f^5+7f_{ty^6}^{(7)}f^6+f_{y^7}^{(7)}f^7\right)\\
&+42c_i^5\left(f_{t^5y}^{(6)}+5f_{t^4y^2}^{(6)}f+10f_{t^3y^3}^{(6)}f^2+10f_{t^2y^4}^{(6)}f^3+5f_{ty^5}^{(6)}f^4+f_{y^6}^{(6)}f^5\right)\lambda_i^{(1)}\\
&+105c_i^4\left(f_{t^4y}^{(5)}+4f_{t^3y^2}^{(5)}f+6f_{t^2y^3}^{(5)}f^2+4f_{ty^4}^{(5)}f^3+f_{y^5}^{(5)}f^4\right)\lambda_i^{(2)}\\
&+420c_i^3\left(f_{t^3y^2}^{(5)}+3f_{t^2y^3}^{(5)}f+3f_{ty^4}^{(5)}f^2+f_{y^5}^{(5)}f^3\right)\left(\lambda_i^{(1)}\right)^2\\
&+140c_i^3\left(f_{t^3y}^{(4)}+3f_{t^2y^2}^{(4)}f+3f_{ty^3}^{(4)}f^2+f_{y^4}^{(4)}f^3\right)\lambda_i^{(3)}\\
&+1260c_i^2\left(f_{t^2y^2}^{(4)}+2f_{ty^3}^{(4)}f+f_{y^4}^{(4)}f^2\right)\lambda_i^{(1)}\lambda_i^{(2)}
\end{aligned}
$$

$$
\begin{aligned}
&+ 840c_i \left(f^{(4)}_{ty^3} + f^{(4)}_{y^4} f\right) \left(\lambda_i^{(1)}\right)^3 \\
&+ 105c_i^2 \left(f^{(3)}_{t^2y} + 2f^{(3)}_{ty^2} f + f^{(3)}_{y^3} f^2\right) \lambda_i^{(4)} \\
&+ 210c_i \left(f^{(3)}_{ty^2} + f^{(3)}_{y^3} f\right) \left(4\lambda_i^{(1)}\lambda_i^{(3)} + 3\left(\lambda_i^{(2)}\right)^2\right) \\
&+ 1260 f^{(3)}_{y^3} \left(\lambda_i^{(1)}\right)^2 \lambda_i^{(2)} \\
&+ 42c_i \left(f^{(2)}_{ty} + f^{(2)}_{y^2} f\right) \lambda_i^{(5)} \\
&+ 210 f^{(2)}_{y^2} \left(\lambda_i^{(1)}\lambda_i^{(4)} + 2\lambda_i^{(2)}\lambda_i^{(3)}\right) + 7f^{(1)}_y \lambda_i^{(6)},
\end{aligned}
$$

$$
\begin{aligned}
\kappa_i^{(8)} =& c_i^8 \left(f^{(8)}_{t^8} + 8f^{(8)}_{t^7y} f + 28f^{(8)}_{t^6y^2} f^2 + 56f^{(8)}_{t^5y^3} f^3 + 70f^{(8)}_{t^4y^4} f^4 + 56f^{(8)}_{t^3y^5} f^5 + 28f^{(8)}_{t^2y^6} f^6 + 8f^{(8)}_{ty^7} f^7 + f^{(8)}_{y^8} f^8\right) \\
&+ 56c_i^6 \left(f^{(7)}_{t^6y} + 6f^{(7)}_{t^5y^2} f + 15f^{(7)}_{t^4y^3} f^2 + 20f^{(7)}_{t^3y^4} f^3 + 15f^{(7)}_{t^2y^5} f^4 + 6f^{(7)}_{ty^6} f^5 + f^{(7)}_{y^7} f^6\right) \lambda_i^{(1)} \\
&+ 168c_i^5 \left(f^{(6)}_{t^5y} + 5f^{(6)}_{t^4y^2} f + 10f^{(6)}_{t^3y^3} f^2 + 10f^{(6)}_{t^2y^4} f^3 + 5f^{(6)}_{ty^5} f^4 + f^{(6)}_{y^6} f^5\right) \lambda_i^{(2)} \\
&+ 840c_i^4 \left(f^{(6)}_{t^4y^2} + 4f^{(6)}_{t^3y^3} f + 6f^{(6)}_{t^2y^4} f^2 + 4f^{(6)}_{ty^5} f^3 + f^{(6)}_{y^6} f^4\right) \left(\lambda_i^{(1)}\right)^2 \\
&+ 280c_i^4 \left(f^{(5)}_{t^4y} + 4f^{(5)}_{t^3y^2} f + 6f^{(5)}_{t^2y^3} f^2 + 4f^{(5)}_{ty^4} f^3 + f^{(5)}_{y^5} f^4\right) \lambda_i^{(3)}
\end{aligned}
$$

$$
\begin{aligned}
&+ 3360c_i^3 \left(f^{(5)}_{t^3y^2} + 3f^{(5)}_{t^2y^3} f + 3f^{(5)}_{ty^4} f^2 + f^{(5)}_{y^5} f^3\right) \lambda_i^{(1)}\lambda_i^{(2)} \\
&+ 3360c_i^2 \left(f^{(5)}_{t^2y^3} + 2f^{(5)}_{ty^4} f + f^{(5)}_{y^5} f^2\right) \left(\lambda_i^{(1)}\right)^3 \\
&+ 280c_i^3 \left(f^{(4)}_{t^3y} + 3f^{(4)}_{t^2y^2} f + 3f^{(4)}_{ty^3} f^2 + f^{(4)}_{y^4} f^3\right) \lambda_i^{(4)} \\
&+ 840c_i^2 \left(f^{(4)}_{t^2y^2} + 2f^{(4)}_{ty^3} f + f^{(4)}_{y^4} f^2\right) \left(4\lambda_i^{(1)}\lambda_i^{(3)} + 3\left(\lambda_i^{(2)}\right)^2\right) \\
&+ 10080c_i \left(f^{(4)}_{ty^3} + f^{(4)}_{y^4} f\right) \left(\lambda_i^{(1)}\right)^2 \lambda_i^{(2)} + 1680 f^{(4)}_{y^4} \left(\lambda_i^{(1)}\right)^4 \\
&+ 168c_i^2 \left(f^{(3)}_{t^2y} + 2f^{(3)}_{ty^2} f + f^{(3)}_{y^3} f^2\right) \lambda_i^{(5)} \\
&+ 1680 \left(f^{(3)}_{ty^2} + f^{(3)}_{y^3} f\right) \left(\lambda_i^{(1)}\lambda_i^{(4)} + 2\lambda_i^{(2)}\lambda_i^{(3)}\right) \\
&+ 1680 f^{(3)}_{y^3} \left(2\lambda_i^{(1)}\lambda_i^{(3)} + 3\left(\lambda_i^{(2)}\right)^2\right) \lambda_i^{(1)} + 56c_i \left(f^{(2)}_{ty} + f^{(2)}_{y^2} f\right) \lambda_i^{(6)} \\
&+ 56 f^{(2)}_{y^2} \left(6\lambda_i^{(1)}\lambda_i^{(5)} + 15\lambda_i^{(2)}\lambda_i^{(4)} + 10\left(\lambda_i^{(3)}\right)^2\right) + 8f^{(1)}_y \lambda_i^{(7)}.
\end{aligned}
$$

The analytical forms of derivatives of y with respect to t can be obtained with Mathematica, Matlab, Derive or similar software. One can also try to obtain with such a software the analytical formulas for $\kappa_i^{(l)}$, presented above, but in our opinion the application of symbols $\lambda_i^{(l)}$ shorts these formulas significantly.

Although the analytical forms of the functions $\kappa_i^{(8)}$ and $y^{(9)}$ are complicated, having such formulas the interval extensions of them can be found immediately. Hence, the interval extension of $\psi(t, y)$, necessary in our interval method, developed in the next section, can be determined easily. It should be noted that the necessity of calculating $\kappa_i^{(8)}$ and $y^{(9)}$ follows directly from the definition of classical Runge-Kutta methods (see, e.g., [6]) and it causes that our method is very expensive. Its effectiveness cannot be compared with very effective methods based on the Taylor series (see, e.g., [2–4, 7, 16, 21, 34–36]), where automatic differentiation is used to compute $y^{(l)}$.

4. An interval version of Kuntzmann-Butcher method

Let us denote:

- Δ_t and Δ_y – bounded sets in which the function $f(t, y)$, occurring in (2.1), is defined, i.e.,
  $\Delta_t = \{t \in \mathbb{R} : 0 \le t \le a\}, \quad \Delta_y \quad = \{y \in \mathbb{R} : \underline{b} \le y \le \overline{b}\},$
- $F(T, Y)$ – an interval extension of $f(t, y)$, where an interval extension of the function
  $f : \mathbb{R} \times \mathbb{R} \supset \Delta_t \times \Delta_y \to \mathbb{R}$
  we call a function
  $F : \mathbb{IR} \times \mathbb{IR} \supset \mathbb{I}\Delta_t \times \mathbb{I}\Delta_y \to \mathbb{IR}$
  such that
  $(t, y) \in (T, Y) \Rightarrow f(t, y) \in F(T, Y),$
  and where $\mathbb{IR}$ denotes the space of real intervals,
- $\Psi(T, Y)$ – an interval extension of $\psi(t, y)$ (see (3.3)).

Let us assume that:

- the function $F(T, Y)$ is defined and continuous for all $T \subset \Delta_t$ and $Y \subset \Delta_y$[1],
- the function $F(T, Y)$ is monotonic with respect to inclusion, i.e.,
  $T_1 \subset T_2 \wedge Y_1 \subset Y_2 \Rightarrow F(T_1, Y_1) \subset F(T_2, Y_2),$
- for each $T \subset \Delta_t$ and for each $Y \subset \Delta_y$ there exists a constant $\Lambda > 0$ such that
  $$w(F(T, Y)) \le \Lambda (w(T) + w(Y)), \tag{4.1}$$
  where $w(A)$ denotes the width of the interval A,
- the function $\Psi(T, Y)$ is defined for all $T \subset \Delta_t$ and $Y \subset \Delta_y$,
- the function $\Psi(T, Y)$ is monotonic with respect to inclusion.

Taking into account (3.1) and (3.2), for $t_0 = 0$ and $y_0 \in Y_0$, where the interval Y_0 is given, we propose the following interval version of Kuntzmann-Butcher method:

$$\begin{aligned} Y_{k+1} &= Y_k + h(w_1 K_{1k} + w_2 K_{2k} + w_3 K_{3k} + w_4 K_{4k}) + \\ &\quad + (\Psi(T_k, Y_k) + [-\alpha, \alpha]) h^9, \qquad k = 0, 1, \ldots, n-1, \end{aligned} \tag{4.2}$$

where

$$\begin{aligned} K_{ik} =& F(T_k + c_i h, \\ & \quad Y_k + h(a_{i1} K_{1k} + a_{i2} K_{2k} + a_{i3} K_{3k} + a_{i4} K_{4k})), \\ & \alpha = M h_0, \quad \left| \frac{r_{k+1}^{(10)}(\theta h)}{10!} \right| \le M, \\ & 0 < \theta < 1, \quad 0 < h \le h_0, \end{aligned} \tag{4.3}$$

and where h_0 denotes a given number (initial value of step size). Note that the constant M, and hence also α, is calculated a priori at the start of the integration. Such an assumption occurs also in explicit interval Runge-Kutta methods developed by Shokin [37]. One can consider an evaluation of M at each step of integration, but our numerical experiments show that the influence of α for enclosures is very small and it is not worth trying to do it (taking into account that such an approach increases the number of calculations).

[1] The function $F(T, Y)$ is continuous at (T_0, Y_0) if for every $\epsilon > 0$ there is a positive number $\delta = \delta(\epsilon)$ such that $d(F(T, Y), F(T_0, Y_0)) < \epsilon$ whenever $d(T, T_0) < \delta$ and $d(Y, Y_0) < \delta$. Here, d denotes the interval metric defined by $d(X_1, X_2) = \max\{|\underline{X}_1 - \underline{X}_2|, |\overline{X}_1 - \overline{X}_2|\}$, where $X_1 = [\underline{X}_1, \overline{X}_1]$ and $X_2 = [\underline{X}_2, \overline{X}_2]$ are two intervals.

The step size h of the method (4.2)–(4.3), which fulfills the condition $0 < h \leq h_0$, is

$$h = \frac{\eta}{n},$$

where

$$\eta = \min\{\eta_0, \eta_1, \eta_2, \eta_3, \eta_4\}, \tag{4.4}$$

and where for $Y_0 \subset \Delta_y$ and $y_0 \in Y_0$ the numbers $\eta_i > 0$ $(i = 1, 2, 3, 4)$ are such that

$$Y_0 + \eta_i c_i F(\Delta_t, \Delta_y) \subset \Delta_y, \quad i = 1, 2, 3, 4,$$

and the number $\eta_0 > 0$ fulfills the condition

$$Y_0 + \eta_0 \sum_{i=1}^{4} w_i F(\Delta_t, \Delta_y) + (\Psi(\Delta_t, \Delta_y) + [-\alpha, \alpha])\, h_0^8 \subset \Delta_y.$$

In [25] we have described a procedure, which in interval floating-point arithmetic calculates the number $\eta = t_{\max}$ for any interval Runge-Kutta method (explicit or implicit). Unfortunately, in many problems it appears that η is very small (see examples in Sec. 5). Although one can try to use the method (4.2)–(4.3) over η, but the value of η given by (4.4) is necessary in the proof of Theorem 4.1 (see the end of this section).

Finally, we divide the interval $[0, \eta]$ into n parts by the points $t_k = kh$ $(k = 0, 1, ..., n)$, whereas the intervals T_k, which appear in the method (4.2)–(4.3), are selected in such a way that

$$t_k = kh \in T_k \subset [0, \eta].$$

Of course, the above formula does not define T_k. In practice, we usually take $T_k = [\underline{kh}, \overline{kh}]$ or $T_k = T_{k-1} + [\underline{h}, \overline{h}]$ with $T_0 = [0, 0]$, where $\underline{x}$ denotes the largest machine number less or equal to x, and $\overline{x}$ denotes the smallest machine number greater or equal to x. We assume here a constant step size h, but one can consider a variable step size to control the widths of enclosures for solution (this problem will be taken into account in our further research).

From (4.3) it follows that in each step k we have to solve a nonlinear equation of the form

$$X = G(T, X),$$

where

$$T \in \mathbb{I}\Delta_t \subset \mathbb{IR}, \quad X \in \mathbb{I}\Delta_y \subset \mathbb{IR}, \quad G : \mathbb{I}\Delta_t \times \mathbb{I}\Delta_y \to \mathbb{IR}.$$

If we assume that G is a contraction (contractive) mapping[2], then the well-known fixed-point theorem implies that the iteration

$$X^{(l+1)} = G(T, X^{(l)}), \quad l = 0, 1, \ldots, \tag{4.5}$$

is convergent to X^*, i.e., $\lim_{l \to \infty} X^{(l)} = X^*$, for an arbitrary choice of $X^{(0)} \in \mathbb{I}\Delta_y$. For equation (4.3), the process (4.5) is of the form

$$\begin{aligned} &K_{ik}^{(l+1)} = F\left(T_k + c_i h,\ Y_k + h \sum_{j=1}^{4} a_{ij} K_{jk}^{(l)}\right), \\ &i = 1, 2, 3, 4 \quad k = 0, 1, \ldots, n-1, \quad l = 0, 1, \ldots, \end{aligned} \tag{4.6}$$

where

$$K_{ik}^{(0)} = F(T_k + c_i h,\ Y_k).$$

[2]Let us recall that G is called a contraction mapping if

$$d(G(T, X_{(1)}), G(T, X_{(2)})) \leq \alpha d(X_{(1)}, X_{(2)}),$$

where d is metric, $\alpha < 1$ denotes a constant, and where $X_{(1)}$ and $X_{(2)}$ are two arbitrary intervals.

The process (4.6) may be modified to

$$K_{ik}^{(l+1)} = F\left(T_k + c_i h,\ \ Y_k + h\left(\sum_{j=1}^{i-1} a_{ij} K_{jk}^{(l+1)} + \sum_{j=i}^{4} a_{ij} K_{jk}^{(l)}\right)\right),$$

which should reduce the number of calculations. The above processes are stopped when

$$\frac{\left|\underline{K}_{ik}^{(l+1)} - \underline{K}_{ik}^{(l)}\right|}{\underline{K}_{ik}^{(l+1)}} < \epsilon \ \text{ and } \ \frac{\left|\overline{K}_{ik}^{(l+1)} - \overline{K}_{ik}^{(l)}\right|}{\overline{K}_{ik}^{(l+1)}} < \epsilon, \quad \left|\underline{K}_{ik}^{(l+1)}\right|, \left|\overline{K}_{ik}^{(l+1)}\right| \neq 0,$$

for $K_{ik}^{(p)} = \left[\underline{K}_{ik}^{(p)}, \overline{K}_{ik}^{(p)}\right]$ $(p = l, l+1)$. Here, ϵ denotes a prescribed accuracy.

For method (4.2), we have

Theorem 4.1. *For the exact solution $y(t)$ of the initial value problem (2.1) we have $y(t_k) \in Y_k$ $(k = 0, 1, ..., n)$, where Y_k are obtained from (4.2).*

The proof of this theorem for the four-stage implicit interval Runge-Kutta method (4.2)–(4.3) can be found in [11, 25]. Similar theorems for other implicit and explicit interval Runge-Kutta methods are presented in [11, 22, 25, 29, 37]. It should be noted that Theorem 4.1 gives only a theoretical aspect of the method (the rounding-errors and the iteration to compute $K_{ik}^{(l+1)}$ to an accuracy ϵ, occurring into practice, are not taken into account).

In [11, 25] we have also proved theorems regarding estimations of $w(Y_k)$ for explicit and implicit interval Runge-Kutta methods. In these theorems for implicit methods, the initial step size h_0 must fulfill some additional conditions connected with the coefficients a_{ij}. Since for our four-stage method the values of these coefficients are known, we can formulate the theorem as follows:

Theorem 4.2. *If Y_k $(k = 1, 2, ..., n)$ are obtained on the basis of the method (4.2) − (4.3), then for h_0 such that*

$$h_0 < \min\left\{1, \frac{1}{0.502\Lambda + 0.025\Lambda^2 + 0.010\Lambda^3 + 0.001\Lambda^4}\right\},$$

where Λ is a constant occurring in (4.1), we have

$$w\left(Y_k\right) \le Qh^8 + Rw\left(Y_0\right) + S \max_{l=1,2,\dots,n} w\left(T_l\right),$$

where Q, R and S denote some nonnegative constants.

Theorem 4.2 gives a theoretical estimation of the width of Y_k. In practice, this width can be calculated easily for each Y_k found, as we do in all examples presented in the next section.

5. Numerical examples

The coefficients w_i, c_i and a_{ij} in (3.1) are real numbers, and they are not exactly represented in floating-point arithmetic. In the method (4.2) − (4.3) they are represented in the form of the intervals:

$$c_1 = 0.0694318442029737[1,2],\ \ c_2 = 0.3300094782075718[6,7],$$
$$c_3 = 0.6699905217924281[3,4],\ \ c_4 = 0.9305681557970262[8,9],$$
$$w_1 = w_4 = 0.1739274225687269[2,3],$$
$$w_2 = w_3 = 0.3260725774312730[7,8],$$
$$a_{11} = a_{44} = 0.0869637112843634[6,7],$$
$$a_{12} = -0.026604180084998[80,79],\ \ a_{21} = 0.1881181174998680[7,8],$$
$$a_{13} = 0.0126274626894047[2,3],\ \ a_{14} = -0.0035551496857956[9,8],$$
$$a_{22} = a_{33} = 0.1630362887156365[3,4],$$

TABLE 1. The interval solution of the problem (5.1)

$t = kh \in T_k$	Y_k	*Width*
0.2	[8.1873075307798185E−0001, 8.1873075307798187E−0001][3]	$\approx 1.08 \cdot 10^{-17}$
0.6	[5.4881163609402641E−0001, 5.4881163609402645E−0001]	$\approx 3.54 \cdot 10^{-17}$
1.0	[3.6787944117144228E−0001, 3.6787944117144236E−0001]	$\approx 6.54 \cdot 10^{-17}$

$$a_{23} = -0.027880428602470[90, 89],\ a_{24} = 0.0067355005945381[5, 6],$$
$$a_{31} = 0.1671919219741887[7, 8],\ a_{32} = 0.3539530060337439[6, 7],$$
$$a_{34} = -0.0141906949311411[5, 4],\ a_{41} = 0.1774825722545226[1, 2],$$
$$a_{42} = 0.3134451147418683[4, 5],\ a_{43} = 0.3526767575162718[6, 7].$$

This notation means that in our computer calculations we have taken, for example,

$$\begin{aligned} w_1 =& 0.1739274225687269[2, 3] \\ =& [\underline{0.17392742256872692}, \overline{0.17392742256872693}], \end{aligned}$$

where, as previously, $\underline{x}$ denotes the largest machine number less or equal to x (similarly, $\overline{x}$ denotes the smallest machine number greater or equal to x). Such intervals enclosure the exact (not representable) values. Although we cannot assert that the left and the right ends of intervals differ internally by one unit in the last (binary) place, but such intervals are satisfactory for the Delphi Pascal *Extended* type used in our calculations.

In the examples presented below, we compare results obtained by our interval version of Kutzmann-Butcher method with exact solutions (if such solutions are known) and with results obtained by the VNODE-LP package [34], which uses an interval method based on high-order Taylor series. We have used our own implementation of floating-point interval arithmetic in Delphi Pascal. This implementation has been written in the form of a unit called $IntervalArithmetic32and64$ (the current version of this unit is presented in [27]). This unit takes advantage of the Delphi Pascal floating-point *Extended* type. All programs written in Delphi Pascal for the examples presented can be found in [26]. In [26] it is also included a Delphi Pascal program for solving any initial value problem by our interval version of Kutzmann-Butcher method. This program requires the user to write a dynamic link library with definitions of appropriate interval functions.

In the examples considered we start with a simple initial value problem, and then we consider more complicated problems, including stiff differential equations.

Example 5.1. Let us consider the simple initial value problem

$$y' = -y, \quad y(0) = 1, \tag{5.1}$$

with the exact solution $y = \exp(-t)$. On the basis of (4.4) for

$$\begin{aligned} \Delta_t &= \{t \in \mathbb{R} : 0 \le t \le 10\}, \\ \Delta_y &= \{y \in \mathbb{R} : \underline{0.000046} \le y \le 1\}, \\ h_0 &= 0.01, \quad M = 2.81 \cdot 10^{-10}, \end{aligned}$$

we have found $t_{\max} = \eta = 1$. Taking $h = 0.01$, and assuming $\epsilon = 10^{-18}$ we have obtained intervals presented in Table 1. The number of iterations (in the equation (4.6)) in each step has not exceeded 8.

Comparing these intervals with the exact solution we see high compatibility. For instance, the exact solution at $t = 1$ is equal to

$$\exp(-1) \approx 0.36787944117144232.$$

The VNODE-LP package at $t = 1$ produces very quickly the output

$$0.367879441171442[1, 6],$$

what can be written in the form

$$[3.678794411714421\mathrm{E}{-}0001, 3.678794411714426\mathrm{E}{-}0001].$$

The width of this interval is $5 \cdot 10^{-16}$. Comparing this result with the widths presented in Table 1 we see that our method gives tighter enclosures of the exact solution. Unfortunately, our method is multiple more expensive from the point of view of execution time. But "something for something".

TABLE 2. Widths of interval solutions for different step sizes for the problem (5.1) and the method (4.2)–(4.3)

h	k	$Y_k(1.0)$	$Width$
0.0001	10000	[3.6787944117144132E−0001, 3.6787944117144305E−0001]	$\approx 1.71 \cdot 10^{-15}$
0.0005	2000	[3.6787944117144210E−0001, 3.6787944117144249E−0001]	$\approx 3.80 \cdot 10^{-16}$
0.001	1000	[3.6787944117144220E−0001, 3.6787944117144242E−0001]	$\approx 2.15 \cdot 10^{-16}$
0.005	200	[3.6787944117144227E−0001, 3.6787944117144236E−0001]	$\approx 8.15 \cdot 10^{-17}$
0.01	100	[3.6787944117144228E−0001, 3.6787944117144236E−0001]	$\approx 6.54 \cdot 10^{-17}$
0.05	20	[3.6787944117144229E−0001, 3.6787944117144235E−0001]	$\approx 5.22 \cdot 10^{-17}$
0.1	10	[3.6787944117144228E−0001, 3.6787944117144234E−0001]	$\approx 5.23 \cdot 10^{-17}$

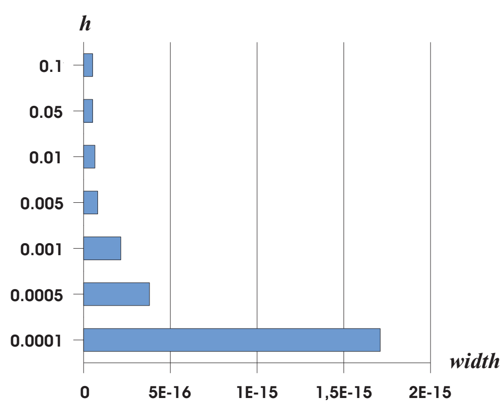

FIGURE 1. Widths of interval solutions for different step sizes for the problem (5.1) and the method (4.2)–(4.3)

[3]All the results are presented in the form obtained by our programs [26]

It may be interesting that in this example for smaller step sizes h we obtain intervals with greater widths, while for some h greater than 0.01 the widths are smaller (see Table 2 and Figure 1, and compare $h = 0.1$ and 0.05 with other values of step sizes). This can be explained by a greater number of calculations for smaller h, which causes a growth of rounding errors. Thus, the step size h should be suitably selected for each problem considered. Of course, too small step size connected with a greater number of calculations has an effect on increase the execution time (see Figure 2, from which it follows that, for example, for $h = 0.0001$ the CPU time is approximately 1000 times longer than for $h = 0.1$). Moreover, it can be observed that the execution time is approximately proportional to the number of steps.

□

In the above simple example, we have $f(t,y) = -y$, and hence a lot of derivatives $f^{(l)}_{t^p y^q}$ $(l = p+q)$, which are needed to find $\Psi(T_k, Y_k)$, are equal to zero (only $f^{(1)}_y$ equals -1). In general, one can put a lot of effort into finding all these derivatives. Mathematical software (e.g., Derive, Matlab, Mathematica) can be very helpful to find analytical forms of them, and then their interval extensions can be determined easily.

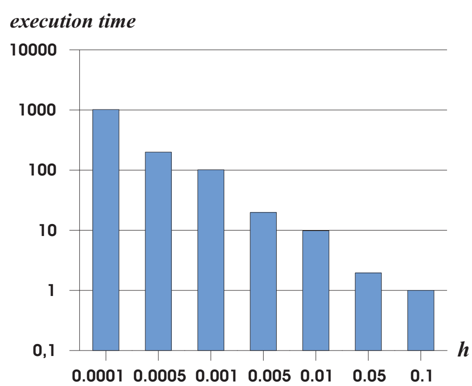

FIGURE 2. Execution times for different step sizes for the problem (5.1) (execution time = 1 unit for $h = 0.1$; on Lenovo Z51 computer with Intel®Core$^{\mathrm{TM}}$ i7-5500U 2,4 GHz processor; 1 unit means 0.633 sec)

Example 5.2. The initial value problem

$$y' = \frac{1}{\exp(t/4)}\left(2\cos 2t - \frac{\sin^2 2t}{4y\exp(t/4)} - \frac{\sin 2t}{4y}\right), \quad y(0) = 1, \tag{5.2}$$

has the exact solution

$$y = 1 + \frac{\sin 2t}{\exp(t/4)}.$$

If we take

$$\Delta_t = \{t \in \mathbb{R} : 0 \le t \le 2\},$$
$$\Delta_y = \left\{y \in \mathbb{R} : \underline{0.4} \le y \le \overline{1.9}\right\},$$
$$h_0 = 0.02, \quad M = 3.87 \cdot 10^{-4},$$

than using the procedure described in Sec. 4, we can find $t_{\max} \approx 0.18$. For $h = 0.01$ and $\epsilon = 10^{-18}$ we have obtained the results presented in Table 3 (with the maximum number of iterations not greater then 5 in each step).
At $t = 0.18$ the exact solution (which is, of course, within our interval) is equal to

$$1 + \sin(0.36)\,/\exp(0.045) \approx 1.33677327992567203,$$

TABLE 3. The interval solution of the problem (5.2)

$t = kh \in T_k$	Y_k	*Width*
0.06	[1.1179299247165512E+0000, 1.1179299247165513E+0000]	$\approx 5.96 \cdot 10^{-18}$
0.12	[1.2306774521289623E+0000, 1.2306774521289624E+0000]	$\approx 1.20 \cdot 10^{-17}$
0.18	[1.3367732799256720E+0000, 1.3367732799256721E+0000]	$\approx 1.79 \cdot 10^{-17}$

while the VNODE-LP package gives

$$1.33677327992567[08, 36].$$

As in Example 5.1, we can observe that our method gives tighter enclosures.

□

One can find many examples for comparing numerical methods for the initial value problem in [8] and [9]. Let us consider two problems from these references (chosen at random).

Example 5.3. For the initial value problem (the problem A5 from [9, p. 23])

$$y' = \frac{y-t}{y+t}, \quad y(0) = 4, \tag{5.3}$$

let us take

$$\Delta_t = \{t \in \mathbb{R} : 0 \le t \le 4\},$$
$$\Delta_y = \left\{y \in \mathbb{R} : 4 \le y \le \overline{6.3}\right\},$$
$$h_0 = 0.01, \quad M = 1.96 \cdot 10^{-3}.$$

For these data and our method, we have found $t_{\max} \approx 1.46$. Using our interval version of Kutzmann-Butcher method with $h = 0.01$ and $\epsilon = 10^{-18}$ we have obtained intervals presented in Table 4 (with the maximum number of iterations not greater than 6 in each step).

TABLE 4. The interval solution of the problem (5.3)

$t = kh \in T_k$	Y_k	*Width*
0.2	[4.1906101485118331E+0000, 4.1906101485118332E+0000]	$\approx 3.51 \cdot 10^{-17}$
0.6	[4.5241459756042591E+0000, 4.5241459756042593E+0000]	$\approx 1.10 \cdot 10^{-16}$
1.0	[4.8075923778847061E+0000, 4.8075923778847064E+0000]	$\approx 1.91 \cdot 10^{-16}$
1.4	[5.0513616875327934E+0000, 5.0513616875327937E+0000]	$\approx 2.79 \cdot 10^{-16}$

At $t = 1.4$ the VNODE-LP package produces the output

$$5.051361687532[7871, 8014],$$

what means that the interval width is approximately equal to 10^{-13}, while in our method we have 10^{-16}. But, on the other hand, the VNODE-LP package gives results much faster than our method.

□

The interval version of Kutzmann-Butcher method can be also applied in the case of data uncertainties.

Example 5.4. Let us consider Example 5.3, but now let us assume that $Y_0 = [\underline{3.99}, \overline{4.01}]$. The intervals obtained are presented in Table 5.

TABLE 5. The interval solution of the problem (5.3) with $Y_0 = [\underline{3.99}, \overline{4.01}]$

$t = kh \in T_k$	Y_k	$Width$
0.2	[4.1796358815223235E+0000, 4.2015893070901247E+0000]	$\approx 2.20 \cdot 10^{-2}$
0.6	[4.5113421918621374E+0000, 4.5369639637013887E+0000]	$\approx 2.56 \cdot 10^{-2}$
1.0	[4.7930841568533574E+0000, 4.8221237507052542E+0000]	$\approx 2.90 \cdot 10^{-2}$
1.4	[5.0352481519527752E+0000, 5.0675071277936416E+0000]	$\approx 3.23 \cdot 10^{-2}$

Taking into account that the width of Y_0 is 0.02, we see that the method (4.2)–(4.3) gives quite good enclosures.

□

Implicit methods are recognized as appropriate for stiff differential equations (see, e.g., [6] and [13]). Thus, in the last example, we consider such a problem. It should be mentioned that although in section 4 we have presented our method in the scalar case (for simplicity), there is no problem to expand the method to the multi-dimensional case.

Example 5.5. Let us take the initial value problem E2 given in [8, p. 32] and [9, p. 21]:

$$\begin{aligned} y_1' = y_2, \quad y_2' = 5(1-y_1^2)y_2 - y_1, \\ y_1(0) = 2, \quad y_2(0) = 0. \end{aligned} \tag{5.4}$$

Assuming

$$\Delta_t = \{t \in \mathbb{R} : 0 \le t \le 1\},$$
$$\Delta_y = \left\{(y_1, y_2) \in \mathbb{R}^2 : \underline{1.8} \le y_1 \le \overline{2.1},\ \underline{-0.2} \le y_2 \le \overline{0.1}\right\},$$
$$h_0 = 0.001, \quad M_1 = 3.32 \cdot 10^3, \quad M_2 = 1.82 \cdot 10^5,$$

we have found $t_{\max} \approx 0.053$. Taking $h = 0.001$ and $\epsilon = 10^{-18}$ at $t = 0.05$, i.e. after 50 steps, we have obtained (with the number of iterations not exceeding 8 at each step)

$$\begin{aligned} Y1 =& [1.9980234267738453\text{E}+0000, 1.9980234267738454\text{E}+0000], \\ Y2 =& [-7.0355564016027207\text{E}-0002, -7.0355564016027200\text{E}-0002], \end{aligned} \tag{5.5}$$

with widths $1.13 \cdot 10^{-17}$ and $5.83 \cdot 10^{-18}$, respectively, while the VNODE-LP package produces

$$\begin{aligned} Y_1 =& 1.99802342677384[48, 55], \\ Y_2 =& -0.070355564016027[1, 3]. \end{aligned}$$

As in the previous examples, our enclosures are tighter than those from VNODE-LP.
It may be interesting that the conventional Kutzmann-Butcher method gives results placed inside the intervals (5.5), namely

$$\begin{aligned} Y_1 =& 1.99802342677384539\text{E}+0000, \\ Y_2 =& -7.03555640160272031\text{E}-0002, \end{aligned} \tag{5.6}$$

but, of course, from these results, we have no information on rounding errors. If in our interval method we assume that $\Psi = 0$ and $\alpha = 0$ (for both Y_1 and Y_2), then at $t = 0.05$ we obtain

$$\begin{aligned} Y1 =& [1.9980234267738453\mathrm{E}{+}0000, 1.9980234267738454\mathrm{E}{+}0000], \\ Y2 =& [-7.0355564016027206\mathrm{E}{-}0002, -7.0355564016027200\mathrm{E}{-}0002], \end{aligned} \tag{5.7}$$

with widths $5.96 \cdot 10^{-18}$ and $5.08 \cdot 10^{-18}$, respectively. These intervals include rounding errors, but we have still no information on the truncation error included in them. Both these errors contain the intervals presented previously. On the other hand, if we compare the widths of intervals (5.5) and (5.7), we see that the truncation error of the method has an insignificant influence on the results obtained.

It may be also interesting to compare the cost of guaranteed bounds versus estimates. Using floating-point interval arithmetic, to include only the rounding errors, i.e., to obtain (5.7), the cost is no greater than eight times (it follows from the definition of multiplication and division of intervals in this arithmetic see, e.g., [14] for details). This cost is significantly increased due to calculations of Ψ and α, but, on the other hand, to well estimate (5.6) one has to put a lot of efforts into doing it.

Unfortunately, although our method at $t = 0.05$ gives tighter intervals than VNODE-LP, this method cannot be recommended for solving the problems like (5.4). Taking into account that the equations (5.4) present a periodic problem with a period of about 12, the value of $t_{max} \approx 0.053$ is much too short to be useful. On the other hand, the VNODE-LP package can integrate this problem for at least a period. Another problem for solving a system of differential equations by interval methods consists in controlling the wrapping effect (for a definition of wrapping effect see, e.g., [14]). It has not been executed yet in our method.

□

In our method, the widths of intervals are greater in each next integration step. Such a situation cannot be accepted in the case if from a theoretical justification it follows that the interval solution is contractive. But there is a way to manage such a problem with our method.

Example 5.6. Consider the problem

$$y' = -y, \quad y(0) \in [1, 2], \tag{5.8}$$

for which the exact solution at $t = 1$ is $y(1) \in [1/e, 2/e]$. It means that the initial interval $[1, 2]$ at $t = 0$ contracts by the factor $1/e$ to $[1/e, 2/e]$ at $t = 1$. The VNODE-LP package produces for this problem quite good interval at $t = 1$:

$$[0.3678794411714420,\ 0.7357588823428852]. \tag{5.9}$$

Unfortunately, using directly our method with

$$\begin{aligned} \Delta_t &= \{t \in \mathbb{R} : 0 \le t \le 10\}, \\ \Delta_y &= \{y \in \mathbb{R} : \underline{0.000046} \le y \le 2\}, \\ h_0 &= 0.01, \quad M = 2.81 \cdot 10^{-10}, \end{aligned}$$

and taking $h = 0.01$, at $t = 1$ we obtain the interval

$$[-8.0783912487742413\mathrm{E}{-}0001,\ 1.9114774483917511\mathrm{E}{+}0000]$$

of the width 2.72, approximately. Although the exact interval is inside the interval obtained, such a solution is not useful. Nevertheless, we can solve the problem (5.8) twice with initial point intervals $Y_0 = [1, 1]$ and $Y_0 = [2, 2]$. For the first point initial interval the solution at $t = 1$ is (see Example 5.1)

$$[3.6787944117144228\mathrm{E}{-}0001,\ 3.6787944117144236\mathrm{E}{-}0001]. \tag{5.10}$$

Taking Δ_t, h_0, M and h the same as previously, and

$$\Delta_y = \{y \in \mathbb{R} : \underline{0.000095} \le y \le 2\},$$

for the second initial point interval at $t = 1$ we obtain

$$[7.3575888234288457\mathrm{E}{-}0001,\ 7.3575888234288471\mathrm{E}{-}0001]. \tag{5.11}$$

If we take the lower bound of interval (5.10) and upper one of interval (5.11), then we obtain the interval

$$[3.6787944117144228\text{E}{-}0001,\ 7.3575888234288471\text{E}{-}0001],$$

which can be accepted as an enclosure of interval solution to the problem (5.8) at $t = 1$. One can observe that the last interval is a little bit tighter than the interval (5.9).

□

6. Conclusions

The main conclusion from the examples presented in this paper and many others carried out by the authors using this and other interval methods is that the interval methods executed in floating-point interval arithmetic yield enclosures of solutions in the form of intervals which contain all possible numerical errors and data uncertainties. Our interval version of the Kutzmann-Butcher method gives very good enclosures of the exact solutions to the initial value problem. The examples presented show that these enclosures are tighter than those obtained by the methods based on high-order Taylor series, as implemented in the VNODE-LP package. This remark does not depreciate these methods (which in our opinion are the best in general), but only points out that sometimes it is worth to consider other interval methods realized in interval floating-point arithmetic. However, the cost of our method is greater than the cost of those methods, but "something for something". A certain inconvenience of our method concerns the integration interval which cannot be too large (see the last example), but this follows directly from the theory of interval Runge-Kutta methods (see (4.4)). The method also required analytical forms of a number of partial derivatives of the right-hand function occurring in (2.1) (to construct their interval extensions), but these derivatives can be obtained easily by one of the well-known mathematical software packages. Although in our method the widths of intervals are greater in each next integration step, the method can be also used successfully in the case of contractive interval solutions (see the last example).

Acknowledgements

The paper was supported by the Poznan University of Technology (Poland) through the Grant No. 09/91/DSPB/0600.

References

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983, Doi: 10.1016/C2009-0-21898-8.

[2] H. Bauch, *On the iterative inclusion of solutions in initial-value problems for ordinary differential equations*, Computing, *22* (1979), 339–354, Doi: 10.1007/BF02265314.

[3] M. Berz and G. Hoffstätter, *Computation and application of Taylor polynomials with interval remainder bounds*, Reliable Computing, *4* (1998), 83–97, Doi: 10.1023/A:1009958918582.

[4] M. Berz and K. Makino, *Performance of Taylor model methods for validated integration of ODEs*, in: J. Dongarra, and K. Madsen, and J. Wasniewski, J. (eds) Applied Parallel Computing. State of the Art in Scientific Computing, (PARA 2004), Lecture Notes in Computer Science, Springer, Berlin, Heidelberg *3732* (2006), 65–73, Doi: 10.1007/11558958-8.

[5] J. C. Butcher, *Implicit Runge-Kutta processes*, Mathematics of Computation, American Mathematical Society, *18* (1964), 50–64, Doi: 10.1090/S0025-5718-1964-0159424-9.

[6] , J. C. Butcher, *The numerical analysis of Ordinary Differential Equations: Runge-Kutta and general linear methods*, Wiley-Interscience, New York, 1987.

[7] G. F. Corliss and R. Rihm, *Validating an a priori enclosure using high-order Taylor series*, in Scientific Computing, Computer Arithmetic, and Validated Numerics *90* (SCAN-95), Mathematical Research, Akademie Verlag, 1996, 228–238.

[8] W. H. Enright, and T. E. Hull, and B. Lindberg, *Comparing numerical methods for stiff systems of O.D.E:s*, BIT Numerical Mathematics, Springer *15* (1975), 10–48, Doi: 10.1007/BF01932994.

[9] W. H. Enright, and J. D. Pryce, *Two FORTRAN packages for assessing initial value methods*, ACM Transactions on Mathematical Software (TOMS), ACM, *13*(1) (1987), 1–27, Doi: 10.1145/23002.27645.

[10] K. Gajda, M. Jankowska, A. Marciniak, and B. Szyszka, *A survey of interval Runge-Kutta and multistep methods for solving the Initial Value Problem*, in R. Wyrzykowski, and J. Dongarra, and K. Karczewski, and J. Wasniewski, (eds) Parallel Processing and Applied Mathematics, (PPAM 2007), Lecture Notes in Computer Science, Springer, Berlin, Heidelberg *4967* (2008), 1361–1371, Doi: 10.1007/978-3-540-68111-3-144.

[11] K. Gajda, A. Marciniak, and B. Szyszka, *Three-and four-stage implicit interval methods of Runge-Kutta type*, Computational Methods in Science and Technology, *6* (2000), 41–59.

[12] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I – nonstiff problems* , Springer-Verlag Berlin Heidelberg, *8* 1993, Doi: 10.1007/978-3-540-78862-1.

[13] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II – stiff and differential - algerbaic problems*, Springer-Verlag Berlin Heidelberg, *14* 1996, Doi: 10.1007/978-3-642-05221-7.

[14] R. Hammer, M. Hocks, U. Kulisch, and D. Ratz, *Numerical toolbox for verified computing I. Basic numerical problems, theory, algorithms, and Pascal-XSC programs*, Springer-Verlag Berlin Heidelberg, *21* 1993, Doi: 10.1007/978-3-642-78423-1.

[15] E. R. Hansen, *Topics in interval analysis*, Oxford University Press, London, 1969.

[16] K. R. Jackson and N. S. Nedialkov, *Some recent advances in validated methods for IVPs for ODEs*, Applied Numerical Analysis and Computational Mathematics, *42* (2002), 269–284.

[17] M. Jankowska and A. Marciniak, *Implicit interval multistep methods for solving the initial value problem*, Computational Methods in Science and Technology, *8*(1) (2002), 17–30.

[18] M. Jankowska and A. Marciniak, *On explicit interval methods of Adams-Bashforth type*, Computational Methods in Science and Technology, *8*(2) (2002), 46–57.

[19] M. Jankowska and A. Marciniak, *On two families of implicit interval methods of Adams-Moulton type*, Computational Methods in Science and Technology, *12*(2) (2006), 109–114.

[20] S. A. Kalmykov, Y. I. Shokin, and Z. H. Juldashev, *On the solution of ordinary differential equations by interval methods [in Russian]*, Doklady Akad. Nauk SSSR 230, *6* (1976), 1267–1270.

[21] R. J. Lohner, *Computation of guaranteed enclosures for the solutions of ordinary initial and boundary value problems*, in J. R. Cash, and I. Gladwell, (eds), Computational Ordinary Differential Equations, Clarendon Press, Oxford, 1992, 425–435.

[22] A. Marciniak, *Implicit interval methods for solving the initial value problem*, Numerical Algorithms, Springer, *37* (2004), 241–251.

[23] A. Marciniak, *Multistep interval methods of Nyström and Milne-Simpson types*, Computational Methods in Science and Technology, *13*(1) (2007), 23–40.

[24] A. Marciniak, *On multistep interval methods for solving the initial value problem*, Journal of Computational and Applied Mathematics, *199*(2) (2007), 229–237.

[25] A. Marciniak, *Selected interval methods for solving the initial value problem*, http://www.cs.put.poznan.pl/amarciniak/IMforIVP-book/IMforIVP.pdf, Publishing House of Poznan University of Technology, Poznan, 2009.

[26] A. Marciniak, *Delphi Pascal programs for an interval Kutzmann-Butcher method*, http://www.cs.put.poznan.pl/amarciniak/IKBM-Examples/, 2016.

[27] A. Marciniak, *Interval arithmetic module*, http://www.cs.put.poznan.pl/amarciniak/IAUnits/IntervalArithmetic32and64.pas, 2016.

[28] A. Marciniak, M. Jankowska, and T. Hoffmann, *On interval predictor-corrector methods*, Numerical Algorithms, *75*(3) (2017), 777–808.

[29] A. Marciniak and B. Szyszka, *One-and two-stage implicit interval methods of Runge-Kutta type*, Computational Methods in Science and Technology, *5* (1999), 53–65.

[30] R. E. Moore, *The automatic analysis and control of error in digital computation based on the use of interval numbers*, in L. B. Rall (ed) Error in Digital Computation, John Wiley & Sons, New York, *1* 1965, 61–130.

[31] R. E. Moore, *Interval analysis*, Prentice-Hall, Englewood Cliffs, 1966.

[32] R. E. Moore, *Methods and applications of interval analysis*, SIAM Studies in applied and numerical mathematics, Soc. for Industrial & Applied Math, Philadelphia, 1979, Doi: 10.1137/1.9781611970906.

[33] N. S. Nedialkov, *Interval tools for ODEs and DAEs*, Technical Report CAS 06-09-NN, Department of Computing and Software, McMaster University, Hamilton, 2006.

[34] N. S. Nedialkov, *VNODE-LP - a validated solver for initial value problems in Ordinary Differential Equations*, Technical Report CAS 06-06-NN, Department of Computing and Software, McMaster University, Hamilton, 2006.

[35] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss, *Validated solutions of initial value problems for Ordinary Differential Equations*, Applied Mathematics and Computation, Elsevier, *105*(1) (1999), 21–68.

[36] K. Nickel, *Using interval methods for the numerical solution of ODE's*, ZAMM-Journal of applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik, *66* (1986), 513–523.

[37] Y. I. Shokin, *Interval analysis [in Russian]*, Nauka, Novosibirsk, 1981.