Interval Runge-Kutta Methods with Variable Step Sizes

Article *in* Computational Methods in Science and Technology · January 2019

DOI: 10.12921/cmst.2019.0000006

2 authors:

Andrzej Marciniak
Poznan University of Technology
54 PUBLICATIONS **348** CITATIONS
SEE PROFILE

Barbara Szyszka
Poznan University of Technology
17 PUBLICATIONS **110** CITATIONS
SEE PROFILE

CMST 25(1) 17–29 (2019) DOI:10.12921/cmst.2019.0000006

Interval Runge-Kutta Methods with Variable Step Sizes

A. Marciniak[1,2], **B. Szyszka**[3]

[1] *Institute of Computing Science*
Poznan University of Technology
Piotrowo 2, 60-965 Poznan, Poland
E-mail: andrzej.marciniak@put.poznan.pl
[2] *Department of Computer Science*
State University of Applied Sciences in Kalisz
Poznanska 201-205, 62-800 Kalisz, Poland
[3] *Institute of Mathematics*
Poznan University of Technology
Piotrowo 3A, 60-965 Poznan, Poland
E-mail: barbara.szyszka@put.poznan.pl

Received: 25 February 2019; revised: 28 March 2019; accepted: 28 March 2019; published online: 31 March 2019

Abstract: In a number of our previous papers we have presented interval versions of Runge-Kutta methods (explicit and implicit) in which the step size was constant. Such an approach has required to choose manually the step size in order to ensure an interval enclosure to the solution with the smallest width. In this paper we propose an algorithm for choosing automatically the step size which guarantees the best (i.e., the tiniest) interval enclosure. This step size is determined with machine accuracy.

Key words: initial value problem, Runge-Kutta methods, interval Runge-Kutta methods, variable step size, floating-point interval arithmetic

I. INTRODUCTION

Interval arithmetic (see, e.g., [10, 25, 26, 29]) realized in floating-point computer arithmetic is a way to estimate two kinds of errors: representation errors of data (some real numbers cannot be represented exactly in floating-point arithmetic) and rounding errors (the difference between the calculated approximation of a number and its exact mathematical value). Using approximate methods we introduce the third kind of errors – the error of method (often called the truncation error, usually defined as the error that results from using an approximation in place of an exact mathematical procedure). Applying interval methods to approximate the solution of the initial value problem in floating-point interval arithmetic (see, e.g., [9]), we can obtain enclosures of the solution in the form of intervals which contain all possible numerical errors.

Three main kinds of interval methods for solving the initial value problem are known: the methods based on high-order Taylor series (see, e.g., [1, 2, 4, 11, 28]), explicit and implicit methods of Runge-Kutta type [6, 7, 15, 17, 23, 24, 29], and explicit and implicit multistep methods [12–14, 16, 17, 29]. So far, only in the first kind of interval methods the step size correction has been applied. In interval methods based on Runge-Kutta methods and in interval multistep methods a constant step size has been used. Although the methods based on high-order Taylor series seem to be most universal, in [20–22, 24] we have shown that in some cases the interval methods of the second and third kind give better enclosures of solutions.

In this paper we propose an algorithm for finding the optimal step size (taking into account the widths of intervals obtained) for interval methods of Runge-Kutta type. We present an application of this algorithm taking into account an interval method based on the classical Runge-Kutta method of fourth order as an example. But the presented procedure can also be applied to other kinds of interval methods of Runge-Kutta type.

The paper is divided into seven sections. In Sec. II. and III. we shortly recall the well-known conventional Runge-Kutta methods and the classical Runge-Kutta method of fourth order, respectively. An interval version of fourth order Runge-Kutta method [17, 29] is recalled in Sec. IV.. Sec. V. is the main section of this paper, in which we describe an algorithm for finding the optimal step size. In Sec. VI. we present some numerical examples which show the application of this algorithm. In the last section some conclusions are given.

II. RUNGE-KUTTA METHODS

Let us consider the initial value problem

$$y' = f(t, y(t)), \quad y(0) = y_0, \tag{1}$$

where $t \in [0, a]$, $y \in \mathbb{R}$ and $f : [0, a] \times \mathbb{R} \to \mathbb{R}$. As it is well-known, the (explicit) m-stage Runge-Kutta methods for solving the problem (1) are given by the formula [3, 8]

$$y_{k+1} = y_k + h \sum_{i=1}^{m} w_i \kappa_{ik}, \quad k = 0, 1, \ldots, \tag{2}$$

where

$$\kappa_{ik} = f(t_k + c_i h, y_k + h \sum_{j=1}^{i-1} a_{ij} \kappa_{jk}), \quad i = 1, 2, \ldots, m, \tag{3}$$

and

$$c_i = \sum_{j=1}^{i-1} a_{ij}, \tag{4}$$

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

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & \vdots & & \\ c_m & a_{m1} & a_{m2} & \cdots & a_{mm} \\ \hline & w_1 & w_2 & \cdots & w_m \end{array} .$$

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

$$\begin{aligned} r_{k+1}(h) &= y(t_k + h) - \left(y(t_k) + h \sum_{i=1}^{m} w_i \kappa_{ik}(h) \right) \\ &= \psi(t_k, y(t_k)) h^{p+1} + O(h^{p+2}) \\ &= r_{k+1}^{(p+1)}(0) \frac{h^{p+1}}{(p+1)!} + r_{k+1}^{(p+2)}(\theta h) \frac{h^{p+2}}{(p+2)!}, \\ & \qquad 0 < \theta < 1, \end{aligned} \tag{5}$$

where $y(t_k+h)$ and $y(t_k)$ denote the exact solutions at t_k+h and t_k, respectively, and $\kappa_{ik}(h)$ is given by (3) for the exact value $y(t_k)$. From the conditions $r_{k+1}^{(l)}(0) = 0$ (for $l = 1, 2, ..., p$) follow the equations for determining the coefficients w_i, c_i and a_{ij}. Unfortunately, there are fewer equations than the number of unknowns, and usually we consider some special cases. It is known [3, 8] that for each m there exists a method with maximum order $p_{\max}(m) = m$ for $m = 1, 2, 3, 4$, $p_{\max}(m) = m - 1$ for $m = 5, 6, 7$, $p_{\max}(m) = m - 2$ for $m = 8, 9$, and $p_{\max}(m) \leq m - 2$ for $m \geq 10$.

III. THE CLASSICAL RUNGE-KUTTA METHOD

One of the most popular Runge-Kutta methods (simply called the Runge-Kutta method) is the four-stage fourth order method of the form

$$y_{k+1} = y_k + \frac{h}{6} (\kappa_{1k} + 2\kappa_{2k} + 2\kappa_{3k} + \kappa_{4k}), \tag{6}$$

where

$$\begin{aligned} \kappa_{1k} &= f(t_k, y_k), \\ \kappa_{2k} &= f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}\kappa_{1k}\right), \\ \kappa_{3k} &= f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}\kappa_{2k}\right), \\ \kappa_{4k} &= f(t_k + h, y_k + h\kappa_{3k}). \end{aligned} \tag{7}$$

The Butcher table for this method is as follows:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array} .$$

The local truncation error is given by the following formula:

$$r_{k+1}(h) = \psi(t_k, y(t_k))h^5 + O(h^6).$$

The form of $\psi(t,y)$ is rather complicated. Since this form is very important from the point of view of the interval method developed in the next section, below we present the adequate formulas.

Denoting (to short the notations)

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

where $l = p + q$, and

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

we have

$$\psi(t,y) = \frac{1}{120}\left(y^{(5)} - 5\sum_{i=1}^{4} w_i \kappa_i^{(4)}\right), \tag{8}$$

where

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

The derivatives of y with respect to t can be written in the following forms:

$$\begin{aligned}
y^{(2)} =& f_t^{(1)} + f_y^{(1)} f,\\
y^{(3)} =& f_{t^2}^{(2)} + 2f_{ty}^{(2)} f + f_{y^2}^{(2)} f^2 + f_y^{(1)} y^{(2)},\\
y^{(4)} =& f_{t^3}^{(3)} + 3f_{t^2y}^{(3)} f + 3f_{ty^2}^{(3)} f^2 + f_{y^3}^{(3)} f^3\\
&+ 3\left(f_{ty}^{(2)} + f_{y^2}^{(2)} f\right) y^{(2)} + f_y^{(1)} y^{(3)},\\
y^{(5)} =& f_{t^4}^{(4)} + 4f_{t^3y}^{(4)} f + 6f_{t^2y^2}^{(4)} f^2 + 4f_{ty^3}^{(4)} f^3 + f_{y^4}^{(4)} f^4\\
&+ 6\left(f_{t^2y}^{(3)} + 2f_{ty^2}^{(3)} f + f_{y^3}^{(3)} f^2\right) y^{(2)}\\
&+ 4\left(f_{ty}^{(2)} + f_{y^2}^{(2)} f\right) y^{(3)} + f_y^{(1)} y^{(4)}\\
&+ 3f_{y^2}^{(2)}\left(y^{(2)}\right)^2.
\end{aligned}$$

IV. AN INTERVAL VERSION OF RUNGE-KUTTA METHOD

Let us denote:

- Δ_t and Δ_y – bounded sets in which the function $f(t,y)$, occurring in (1), is defined, i.e.,

$$\Delta_t = \{t \in \mathbb{R} : 0 \le t \le a\},$$
$$\Delta_y = \left\{y \in \mathbb{R} : \underline{b} \le y \le \overline{b}\right\},$$

- $F(T,Y)$ – 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 (8)),

and 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

$$w(F(T,Y)) \le \Lambda(w(T) + w(Y)), \tag{9}$$

where $w(A)$ denotes the width of the interval A,

[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 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.

For $t_0 = 0$ and $y_0 \in Y_0$, where the interval Y_0 is given, the interval version of Runge-Kutta method is of the form [17, 29]

$$\begin{aligned} Y_{k+1} =& Y_k + \frac{h}{6}\left(K_{1k} + 2K_{2k} + 2K_{3k} + K_{4k}\right) \\ &+ \left(\Psi\left(T_k, Y_k\right) + [-\alpha, \alpha]\right) h^5, \\ & k = 0, 1, \ldots, n-1, \end{aligned} \tag{10}$$

where

$$\begin{aligned} K_{1k} &= F\left(T_k, Y_k\right), \\ K_{2k} &= F\left(T_k + \frac{h}{2}, Y_k + \frac{h}{2}K_{1k}\right), \\ K_{3k} &= F\left(T_k + \frac{h}{2}, Y_k + \frac{h}{2}K_{2k}\right), \\ K_{4k} &= F\left(T_k + h, Y_k + hK_{3k}\right), \\ \alpha &= Mh_0, \quad \left|\frac{r_{k+1}^{(6)}(\theta h)}{720}\right| \le M, \\ & 0 < \theta < 1, \;\; 0 < h \le h_0, \end{aligned} \tag{11}$$

and where h_0 denotes a given number (an initial value of step size).

According to the theory of interval Runge-Kutta methods [17, 29], the step size h of the method (10)–(11) is given by

$$h = \frac{\eta}{n}, \tag{12}$$

where

$$\eta = \min\left\{\eta_0, \eta_2, \eta_3, \eta_4\right\}, \tag{13}$$

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

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

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

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

In the above relations we have (according to the Butcher table presented in Sec. III.) $c_2 = c_3 = 1/2, c_4 = 1, w_1 = w_4 = 1/6, w_2 = w_3 = 1/3$. In [17] 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).

Finally, we divide the interval $[0, \eta]$ into n parts by the points $t_k = kh \quad (k = 0, 1, ..., n)$, whereas the intervals T_k, which appear in the method (10)–(11), are selected in such a way that

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

For the method (10)–(11) we have the following

Theorem 1 *For the exact solution $y(t)$ of the initial value problem (1) we have $y(t_k) \in Y_k$ $(k = 0, 1, ..., n)$, where Y_k are obtained from (10)–(11).*

The proof of this theorem can be found in [17, 29], where one can also find a theorem regarding estimations of $w(Y_k)$.

Theorem 2 *If Y_k $(k = 1, 2, ..., n)$ are obtained on the basis of the method (10)–(11), then*

$$w\left(Y_k\right) \le Qh^4 + Rw\left(Y_0\right) + S \max_{l=0,1,\ldots,k-1} w\left(T_l\right), \tag{14}$$

where Q, R and S denote some nonnegative constants.

The estimation (14) is true for any explicit interval Runge-Kutta method of fourth order (not only for the method (10)–(11)). In the case of the formulas (10)–(11) the constants Q, R and S are as follows (see the proof of the theorem in [17, 29]):

$$\begin{aligned} Q =& \left[w\left(\Psi\left(\Delta_t, \Delta_y\right)\right) + 2\alpha\right] \frac{S}{\gamma\Lambda}, \\ R =& \exp(\gamma\eta\Lambda), \\ S =& R - 1, \end{aligned} \tag{15}$$

where

$$\begin{aligned} \gamma &= \sum_{i=1}^{4} w_i \sum_{j=0}^{i-1} \mu_{ij} \left(h_0\Lambda\right)^j, \\ \mu_{i0} &= 1, \qquad i = 1, 2, 3, 4, \\ \mu_{i1} &= \frac{1}{2}, \qquad i = 2, 3, 4, \\ \mu_{32} &= \frac{1}{4}, \quad \mu_{42} = \frac{1}{2}, \quad \mu_{43} = \frac{1}{4}, \end{aligned}$$

and where $w_1 = w_4 = 1/6, w_2 = w_3 = 1/3$, and Λ is a constant occurring in (9).

V. CHANGING THE STEP SIZE IN THE INTERVAL RUNGE-KUTTA METHOD

In conventional Runge-Kutta methods the step size is changed to decrease errors of methods. In case of interval methods these errors are included in interval enclosures

of solutions. The only reason to change a given step size for such methods is to decrease the widths of interval enclosures. Since all calculations are performed in floating-point arithmetic which produces rounding errors, for each problem considered there is a step size decreasing of which will not give intervals with smaller widths. A procedure for changing the step size for the method (10)–(11), described below, is based on halving or doubling the size of a given step.

Let $h \leq h_0$ be a step calculated from (12), and let us denote by $Y(h)$ an interval obtained from (10) for $k = 0$, i.e., at T_1 ($t_1 = h \in T_1$). Let $Y\left(2 \times \frac{h}{2}\right)$ be an interval obtained at T_1 by an application of (10) two times. According to (14) we have

$$\begin{aligned} w\left(Y(h)\right) &\leq Qh^4 + Rw\left(Y_0\right) + Sw\left(T_0\right), \\ w\left(Y\left(2 \times \frac{h}{2}\right)\right) &\leq Q\left(\frac{h}{2}\right)^4 + Rw\left(Y\left(\frac{h}{2}\right)\right) \\ &\quad + Sw\left(T_{1/2}\right) \\ &\leq Q\left(\frac{h}{2}\right)^4 + R\Bigg(Q\left(\frac{h}{2}\right)^4 \\ &\quad + Rw\left(Y_0\right) + Sw\left(T_0\right)\Bigg) \\ &\quad + Sw\left(T_{1/2}\right) \\ &\leq Q(R+1)\left(\frac{h}{2}\right)^4 + R^2 w\left(Y_0\right) \\ &\quad + S(R+1)w\left(T_{1/2}\right), \end{aligned} \tag{16}$$

where we take advantage of $w\left(T_0\right) \leq w\left(T_{1/2}\right), t_{1/2} = \frac{h}{2} \in T_{1/2}$.

We should consider two cases:

$$w\left(Y(h)\right) \leq w\left(Y\left(2 \times \frac{h}{2}\right)\right) \tag{17}$$

and

$$w\left(Y\left(2 \times \frac{h}{2}\right)\right) < w\left(Y(h)\right). \tag{18}$$

In case of (17) we see that by halving step size we do not obtain a better interval (i.e., with a smaller width) at T_1 and we can try to double h. In this case we substitute

$$Y\left(\frac{h}{2}\right) := Y(h), \quad h := 2h,$$

and calculate $Y\left(2 \times \frac{h}{2}\right)$ and $Y(h)$ for this new value of h. If (18) occurs, then the step size h should be halved. We substitute

$$Y(h) := Y\left(\frac{h}{2}\right), \quad h := \frac{h}{2},$$

and find $Y\left(2 \times \frac{h}{2}\right)$ for the new h.

Doubling h we should always keep $h \leq h_0$ in mind. If we halve h, we should take into account the inequalities (16). Adding these inequalities by sides we have

$$\begin{aligned} &w\left(Y(h)\right) + w\left(Y\left(2 \times \frac{h}{2}\right)\right) \leq \\ &\frac{Q(R+5)}{4}h^4 + R(R+1)w\left(Y_0\right) + Sw\left(T_0\right) \\ &\qquad + S(R+1)w\left(T_{1/2}\right) \leq \\ &\frac{Q(R+5)}{4}h^4 + R(R+1)w(Y_0) + S(R+2)w\left(T_{1/2}\right), \end{aligned}$$

from which it follows that

$$h^4 \geq 4\frac{w\left(Y(h)\right) + w\left(Y\left(2 \times \frac{h}{2}\right)\right) - R(R+1)w\left(Y_0\right) - S(R+2)w\left(T_{1/2}\right)}{Q(R+5)}. \tag{19}$$

Since in a number of examples we have $w\left(Y_0\right) \approx 0$ and $w\left(T_{1/2}\right) \approx 0$, assuming $w\left(Y_0\right) = 0$ and $w\left(T_{1/2}\right) = 0$, from (19) we have

$$h \geq \sqrt{2} \cdot \sqrt[4]{\frac{w\left(Y(h)\right) + w\left(Y\left(2 \times \frac{h}{2}\right)\right)}{Q(R+5)}}. \tag{20}$$

Applying the presented procedure, after a number of steps we observe that halving and doubling the step size h do not decrease the width of interval solution at some T_1, or $h > h_0$, or the inequalities (20) does not hold. In all these case we accept the interval obtained with the last correct h. This procedure (presented in part A of block diagram in Fig. 1) gives us only the information that the correct step size h is within the interval $\left[\frac{h}{2}, 2h\right]$, but there can exist a step size $\overline{h}$ such that $\frac{h}{2} \leq \overline{h} \leq h$ or $h \leq \overline{h} \leq 2h$, for which the width of Y is smaller. With a machine accuracy eps we are able to find such an $\overline{h}$ in the interval $[h, 2h]$ for which the width of Y is w_1 (part B in Fig. 1) and such an $\overline{h'}$ in the interval $\left[\frac{h}{2}, h\right]$ with the width of Y equals w_1' (part C in Fig. 1). If $w_1 \neq w_1'$, we substitute $h := \left(\overline{h} + \overline{h'}\right)/2$ and repeat part A with this new step size. Otherwise ($w_1 = w_1'$), the obtained $\overline{h}$ is the optimal step size (according to the machine accuracy and because of $\overline{h} \geq \overline{h'}$). It means that the width of interval $Y\left(\overline{h}\right)$ is the smallest.

The above procedure may be applied for the next integration steps, i.e., for $k > 0$. Since $w\left(Y_{k+1}\right) \geq w\left(Y_k\right)$, which follows immediately from (10), and taking into account the expense of the described algorithm, for $k > 0$ we propose to use the constant step size obtained in the first integration step, i.e., for $k = 0$. Although we have considered only the fourth order interval Runge-Kutta method, the same algorithm can be applied to other interval methods of Runge-Kutta type (explicit and implicit – see, e.g., [6, 7, 15–17, 23, 24, 29]).

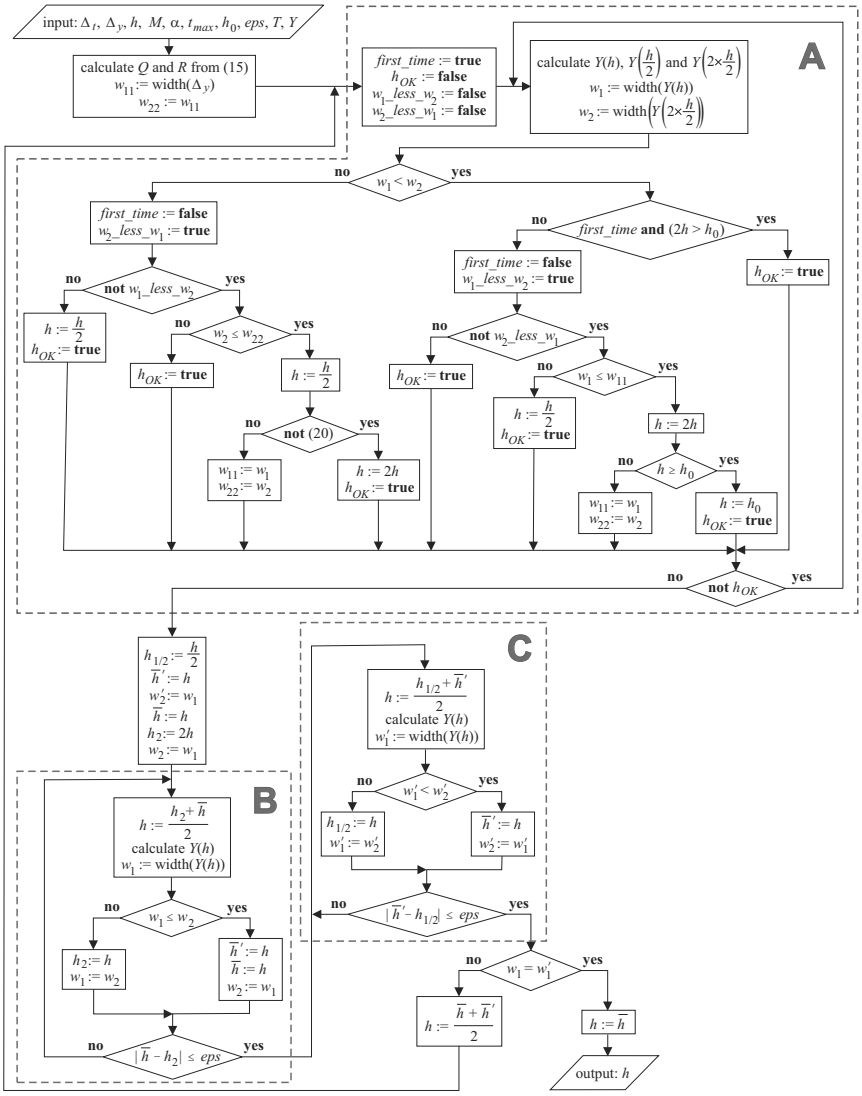

Fig. 1. A block diagram for step size changing

In fact, we have presented our algorithm only for one-dimensional initial value problems, but it can be easily extended to systems of ordinary differential equations (interval methods of Runge-Kutta type for a system of equations have been presented, among others, in [7], [17] and [23]). In this case the inequalities (17) and (18) should be replaced by

$$\sum_{i=1}^{N} w_i \left(Y_{(i)}(h)\right) \le \sum_{i=1}^{N} w_i \left(Y_{(i)}\left(2 \times \frac{h}{2}\right)\right)$$

and

$$\sum_{i=1}^{N} w_i \left(Y_{(i)}\left(2 \times \frac{h}{2}\right)\right) < \sum_{i=1}^{N} w_i \left(Y_{(i)}(h)\right),$$

respectively, where N denotes the number of equations in the system.

VI. NUMERICAL EXAMPLES

In the examples presented below 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 [18]). 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 [19]. The first two examples concern a commonly used test problem with a known exact solution. For the third example the exact solution is unknown, but we compare our results with those obtained by a method based on variable order Taylor series.

VI. 1. Example 1

At first, let us consider the following simple initial value problem:

$$y' = 0.5y, \quad y(0) = 1, \tag{21}$$

with the exact solution $y = \exp(0.5t)$. Let us assume

$$\Delta_t = \{t \in \mathbb{R} : 0 \leq t \leq 10\},$$
$$\Delta_y = \{y \in \mathbb{R} : \underline{0.9} \leq y \leq 149\},$$

where $\underline{x}$ denotes the largest machine number less or equal to x (similarly, by $\overline{x}$ we will denote the smallest machine number greater or equal to x). We can find that $M = 0.003$, and on the basis of (13) for $h_0 = 0.05$ we have $t_{\max} = \eta \approx 1.9866$. It appears that for $h < h_0$ and for the first integration step, the algorithm presented in Fig. 1 gives the optimal step size $\overline{h} = 7.66261590758908911\text{E-}4$. This step size is obtained regardless of whether the starting step size h is slightly or much smaller/greater than the final step size $\overline{h}$ (see Tab. 1–4).

Using optimal step size we obtain interval solutions presented in Tab. 5. Note that the integration step 2592 is the last one for which $t \in [0, t_{\max}]$. Taking $h = 0.0005$ and $h = 0.001$ at the last integration for which $t \in [0, t_{\max}]$ we get

$$\begin{aligned} Y\,([&1.9864999999999997\text{E+0000},\\ &1.9865000000000001\text{E+0000}])\\ =[&2.6999952128764781\text{E+0000},\\ &2.6999952128764800\text{E+0000}] \end{aligned}$$

and

$$\begin{aligned} Y\,([&1.9859999999999998\text{E+0000},\\ &1.9860000000000001\text{E+0000}])\\ =[&2.6993202984410782\text{E+0000},\\ &2.6993202984410801\text{E+0000}] \end{aligned}$$

with the widths approximately equal to 1.79E-15 and 1.88E-15, respectively. Both these widths are greater than the widths of intervals obtained with the optimal step size. Obviously, in each case the exact solution is within the interval enclosure obtained.

All calculations have been carried out on a computer with Intel® Core™ i7–5500U CPU@ 2.40 GHz processor. To find the optimal step size this computer needed about 1:28 min (1 minute and 28 seconds), while all other calculations (to achieve $t_{\max}$ with this optimal step size, i.e., to execute 2592 integration steps) lasted 27:57 min, approximately. ■

VI. 2. Example 2

In Example 1 we have found the optimal step size only for the first integration step, and then we have used this step size as a constant one for further steps. Although the procedure for step size changing is very expensive, it may be interesting to see how the step size changes in further steps. Thus, let us consider the same initial value problem as in Example 1 (with the same additional data), but now let us apply the presented algorithm in each integration step. The changes of step size in the interval $[0, t_{\max}]$ are showed in Fig. 2, and the interval enclosures obtained are presented in Tab. 6.

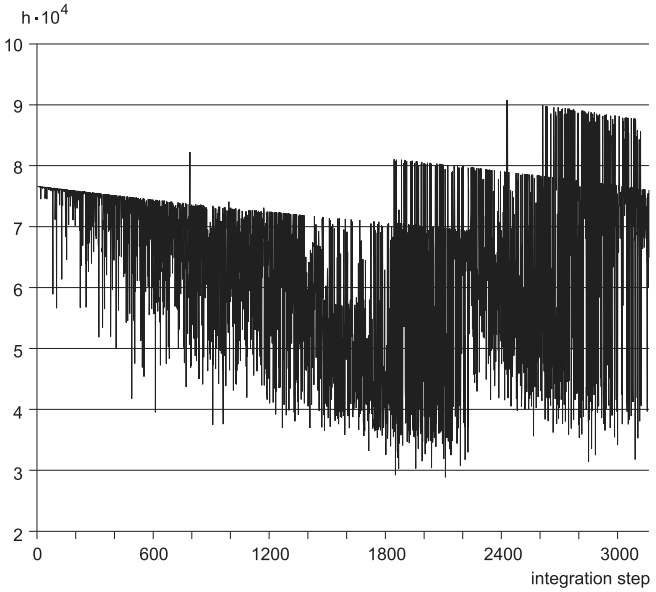

Fig. 2. Changes of step size during calculations

From Fig. 2 it follows that there is no regularity in step size changes. We can only observe that the values of step size fluctuate more for greater t, i.e., for further integration steps. It may be interesting that the mean value of step size on the whole integration interval $[0, t_{\max}]$ equals 6.28E-4, approximately, which differs insignificantly from the optimal step size for the first integration step ($\approx$ 7.66E-4). On the other hand, comparing the results in the last lines in Tab. 5 and 6, we see that applying the algorithm for step size changing in each integration step brings in small profit (taking into account the widths of intervals obtained). Obviously, the calculation time is much larger in this case and to achieve $t_{\max}$ (in 3162 integration steps) we need 19021:00 min, approximately, i.e., more than 13 days (compare with 1:28 + 27:57 min in Example 1). ■

In Examples 1 and 2 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 0.5). In Example 3 we consider a problem in which all these derivatives are different from zero.

VI. 3. Example 3

For the initial value problem (the problem A5 from [5], p. 23)

$$y' = \frac{y - t}{y + t}, \quad y(0) = 4, \tag{22}$$

Tab. 1. Finding the optimal step size for initial $h = 0.0006$

Step	Step size considered (part of algorithm)	In interval
1	$h(A)$	[3.00000000000000000E-0004, 1.20000000000000000E-0003]
2	$h(A)$	[6.00000000000000000E-0004, 2.40000000000000000E-0003]
		initial $\overline{h} = 6.0000000000000000$E-0004 in $[h/2, 2h] = [3.00000000000000000$E-0004, 1.20000000000000000E-0003], checking $\overline{h}$ in $[\geq h, \leq 2h] = [6.00000000000000000$E-0004, 1.20000000000000000E-0003]
3	$\overline{h}(B)$	[9.00000000000000000E-0004, 1.20000000000000000E-0003]
4	$\overline{h}(B)$	[9.00000000000000000E-0004, 1.05000000000000000E-0003]
...	...	...
62	$\overline{h}(B)$	[9.54555771447376606E-0004, 9.54555771447376607E-0004]
		checking $\overline{h'}$ in $[\geq h/2, \leq \overline{h}] = [3.00000000000000000$E-0004, 9.54555771447376606E-0004]
63	$\overline{h'}(C)$	[3.00000000000000000E-0004, 6.27277885723688303E-0004]
64	$\overline{h'}(C)$	[4.63638942861844152E-0004, 6.27277885723688303E-0004]
...	...	...
122	$\overline{h'}(C)$	[6.27277885723688303E-0004, 6.27277885723688303E-0004]
		new $h = \left(\overline{h} + \overline{h'}\right)/2 = 7.90916828585532455$E-0004
123	$h(A)$	[3.95458414292766227E-0004, 1.58183365717106491E-0003]
		initial $\overline{h} = 3.95458414292766227$E-0004 in $[h/2, 2h] = [1.97729207146383114$E-0004, 7.90916828585532455E-0004] checking $\overline{h}$ in $[\geq h, \leq 2h] = [3.95458414292766227$E-0004, 7.90916828585532455E-0004]
124	$\overline{h}(B)$	[5.93187621439149341E-0004, 7.90916828585532455E-0004]
125	$\overline{h}(B)$	[6.92052225012340898E-0004, 7.90916828585532455E-0004]
...	...	...
182	$\overline{h}(B)$	[7.66261590758908911E-0004, 7.66261590758908912E-0004]
		checking $\overline{h'}$ in $[\geq h/2, \leq \overline{h}] = [1.97729207146383114$E-0004, 7.66261590758908911E-0004]
183	$\overline{h'}(C)$	[1.97729207146383114E-0004, 4.81995398952646012E-0004]
184	$\overline{h'}(C)$	[3.39862303049514563E-0004, 4.81995398952646012E-0004]
...	...	...
242	$\overline{h'}(C)$	[4.81995398952646012E-0004, 4.81995398952646012E-0004]
		$w_1 = w_1' \Rightarrow h := \overline{h} = 7.66261590758908911$E-0004

let us take

$$\Delta_t = \{t \in \mathbb{R} : 0 \leq t \leq 4\},$$
$$\Delta_y = \left\{y \in \mathbb{R} : 4 \leq y \leq \overline{6.3}\right\},$$
$$h_0 = 0.01, \quad M = 0.0537.$$

For these data and the method (10)–(11) we have found $t_{\max} \approx 1.46$. Using our algorithm we get $\overline{h} = 8.17462272838888630$E-0004 as the optimal step size for the first integration step. Using this value as a constant step size for next integration steps we obtain interval enclosures presented in Tab. 7. Note that the intervals obtained are very tiny.

Taking $h = 1.46 - t_{1786}$, from the last result we can find an interval enclosure at $t = 1.46$. We have

$$[5.0849553259401612\text{E}{+}0000,$$
$$5.0849553259401642\text{E}{+}0000]$$

with the width equals 2.87E-15, approximately. On the other hand, the well-known VNODE-LP package [27] (based on variable order Taylor series) produces

$$5.0849553259401[565, 699].^{2}$$

[2] Original output from VNODE-LP.

Tab. 2. Finding the optimal step size for initial $h = 0.0008$

Step	Step size considered (part of algorithm)	In interval
1	$h(A)$	[4.00000000000000000E-0004, 1.60000000000000000E-0003]
2	$h(A)$	[8.00000000000000000E-0004, 3.20000000000000000E-0003]
		initial $\overline{h} = 8.0000000000000000$E-0004 in $[h/2, 2h] = [4.00000000000000000$E-0004, 1.60000000000000000E-0003], checking $\overline{h}$ in $[\geq h, \leq 2h] = [8.00000000000000000$E-0004, 1.60000000000000000E-0003]
3	$\overline{h}(B)$	[1.20000000000000000E-0003, 1.60000000000000000E-0003]
4	$\overline{h}(B)$	[1.20000000000000000E-0003, 1.40000000000000000E-0003]
...	...	...
61	$\overline{h}(B)$	[1.21444277883821840E-0003, 1.21444277883821841E-0003]
		checking $\overline{h'}$ in $[\geq h/2, \leq \overline{h}] = [4.00000000000000000$E-0004, 1.21444277883821840E-0003]
62	$\overline{h'}(C)$	[4.00000000000000000E-0004, 8.07221389419109202E-0004]
63	$\overline{h'}(C)$	[4.00000000000000000E-0004, 6.03610694709554601E-0004]
...	...	...
121	$\overline{h'}(C)$	[6.03610694709554600E-0004, 6.03610694709554601E-0004]
		new $h = \left(\overline{h} + \overline{h'}\right)/2 = 9.09026736773886502$E-0004
122	$h(A)$	[4.54513368386943251E-0004, 1.81805347354777300E-0003]
123	$h(A)$	[2.27256684193471626E-0004, 9.09026736773886502E-0004]
		initial $\overline{h} = 4.54513368386943251$E-0004 in $[h/2, 2h] = [2.27256684193471626$E-0004, 9.09026736773886502E-0004] checking $\overline{h}$ in $[\geq h, \leq 2h] = [4.54513368386943251$E-0004, 9.09026736773886502E-0004]
124	$\overline{h}(B)$	[6.81770052580414877E-0004, 9.09026736773886502E-0004]
125	$\overline{h}(B)$	[6.81770052580414877E-0004, 7.95398394677150689E-0004]
...	...	...
182	$\overline{h}(B)$	[7.66261590758908911E-0004, 7.66261590758908912E-0004]
		checking $\overline{h'}$ in $[\geq h/2, \leq \overline{h}] = [2.27256684193471626$E-0004, 7.66261590758908911E-0004]
183	$\overline{h'}(C)$	[4.96759137476190268E-0004, 7.66261590758908911E-0004]
184	$\overline{h'}(C)$	[6.31510364117549590E-0004, 7.66261590758908911E-0004]
...	...	...
241	$\overline{h'}(C)$	[7.66261590758908910E-0004, 7.66261590758908911E-0004]
		$w_1 = w_1' \Rightarrow h := \overline{h} = 7.66261590758908911$E-0004

The width of this interval is equal to 1.34E-14. Thus, our method gives better enclosure, but – on the other hand – the VNODE-LP package gives a result very quickly. ■

VII. CONCLUSIONS

Until now, interval Runge-Kutta methods (explicit and implicit) have been considered only with a constant time step. This step size (h) has been chosen in such a way that $0 < h \leq h_0$, where h_0 denotes some given step size for which the error of interval method is determined (see (10)–(11) for details). Although from the theory of interval Runge-Kutta methods it follows that one can use any h which fulfils the inequalities $0 < h \leq h_0$, we usually search for such an h which guarantees that the intervals obtained have the smallest widths. So far, the only way to find such an h has been based on a number of trials to apply the considered interval methods for different values of h.

In this paper we have presented an algorithm giving the optimal step size for the next (for the first at the beginning of calculations) integration step in interval Runge-Kutta methods. The optimal step size means the step size which gives a resulting interval with the smallest width. Although our algorithm can be applied in each integration step,

Tab. 3. Finding the optimal step size for initial $h = 0.00001$

Step	Step size considered (part of algorithm)	In interval
1	$h(A)$	[5.0000000000000000E-0006, 2.0000000000000000E-0005]
2	$h(A)$	[1.0000000000000000E-0005, 4.0000000000000000E-0005]
...	...	...
8	$h(A)$	[6.4000000000000000E-0004, 2.5600000000000000E-0003]
		initial $\overline{h} = 6.4000000000000000$E-0004
		in $[h/2, 2h] = [3.2000000000000000$E-0004, 1.2800000000000000E-0003],
		checking $\overline{h}$
		in $[\geq h, \leq 2h] = [6.4000000000000000$E-0004, 1.2800000000000000E-0003]
9	$\overline{h}(B)$	[9.6000000000000000E-0004, 1.2850000000000000E-0003]
10	$\overline{h}(B)$	[9.6000000000000000E-0004, 1.1200000000000000E-0003]
...	...	...
67	$\overline{h}(B)$	[1.0572338452085213E-0004, 1.0572338452085214E-0004]
		checking $\overline{h'}$
		in $[\geq h/2, \leq \overline{h}] = [4.0000000000000000$E-0004, 1.2144427788382184E-0003]
68	$\overline{h'}(C)$	[3.2000000000000000E-0004, 6.8861692260426067E-0004]
69	$\overline{h'}(C)$	[5.0430846130213034E-0004, 6.8861692260426067E-0004]
...	...	...
127	$\overline{h'}(C)$	[6.8861692260426067E-0004, 6.8861692260426067E-0004]
		new $h = \left(\overline{h} + \overline{h'}\right)/2 = 8.7292538390639101$E-0004
128	$h(A)$	[4.3646269195319550E-0004, 1.7458507678127820E-0003]
		initial $\overline{h} = 4.3646269195319550$E-0004
		in $[h/2, 2h] = [2.1823134597659775$E-0004, 8.7292538390639101E-0004]
		checking $\overline{h}$
		in $[\geq h, \leq 2h] = [4.3646269195319550$E-0004, 8.7292538390639101E-0004]
129	$\overline{h}(B)$	[6.5469403792979325E-0004, 8.7292538390639101E-0004]
130	$\overline{h}(B)$	[7.6380971091809213E-0004, 8.7292538390639101E-0004]
...	...	...
187	$\overline{h}(B)$	[7.6626159075890911E-0004, 7.6626159075890912E-0004]
		checking $\overline{h'}$
		in $[\geq h/2, \leq \overline{h}] = [2.1823134597659775$E-0004, 7.6626159075890911E-0004]
188	$\overline{h'}(C)$	[2.1823134597659775E-0004, 4.9224646836775333E-0004]
189	$\overline{h'}(C)$	[3.5523890717217554E-0004, 4.9224646836775333E-0004]
...	...	...
246	$\overline{h'}(C)$	[4.9224646836775330E-0004, 4.9224646836775331E-0004]
		$w_1 = w_1' \Rightarrow h := \overline{h} = 7.6626159075890911$E-0004

due to its complexity (see Example 2) we recommend using the algorithm only for the first integration step and then taking the obtained h as a constant step size for each succeeding steps.

In the examples presented in this paper we have applied our algorithm to an interval version of classical Runge-Kutta method of fourth order. However, there is no restriction to apply the presented procedure to other kinds of interval methods of Runge-Kutta type (explicit and implicit – see, e.g., [16, 17, 23, 24, 29]). It should also be noted that although interval methods based on high-order Taylor series are commonly considered as most universal, sometimes other interval methods (not only of Runge-Kutta type) give better enclosures of the exact solutions (see examples presented in [20–22, 24] and Example 3). This is the main reason to consider also possibilities of applying such methods for solving various initial value problems.

Acknowledgment

The paper was supported by the Poznan University of Technology (Poland) as part of Grant No. 09/91/DSPB/0628.

Tab. 4. Finding the optimal step size for initial $h = 0.01$

Step	Step size considered (part of algorithm)	In interval
1	$h(A)$	$[5.0000000000000000\text{E-}0003, 2.0000000000000000\text{E-}0002]$
2	$h(A)$	$[2.5000000000000000\text{E-}0003, 1.0000000000000000\text{E-}0002]$
...	...	...
5	$h(A)$	$[3.1250000000000000\text{E-}0004, 1.2500000000000000\text{E-}0003]$
		initial $\overline{h} = 6.2500000000000000\text{E-}0004$ in $[h/2, 2h] = [3.1250000000000000\text{E-}0004, 1.2500000000000000\text{E-}0003]$, checking $\overline{h}$ in $[\geq h, \leq 2h] = [6.2500000000000000\text{E-}0004, 1.2500000000000000\text{E-}0003]$
6	$\overline{h}(B)$	$[6.2500000000000000\text{E-}0004, 9.3750000000000000\text{E-}0004]$
7	$\overline{h}(B)$	$[6.2500000000000000\text{E-}0004, 7.8125000000000000\text{E-}0004]$
...	...	...
65	$\overline{h}(B)$	$[7.6626159075890891\text{1E-}0004, 7.6626159075890891\text{2E-}0004]$
		checking $\overline{h'}$ in $[\geq h/2, \leq \overline{h}] = [3.1250000000000000\text{E-}0004, 7.6626159075890891\text{1E-}0004]$
66	$\overline{h'}(C)$	$[5.3938079537945445\text{6E-}0004, 7.6626159075890891\text{1E-}0004]$
67	$\overline{h'}(C)$	$[6.5282119306918168\text{3E-}0004, 7.6626159075890891\text{1E-}0004]$
...	...	...
124	$\overline{h'}(C)$	$[7.6626159075890891\text{0E-}0004, 7.6626159075890891\text{1E-}0004]$
		$w_1 = w_1' \Rightarrow h := \overline{h} = 7.6626159075890891\text{1E-}0004$

Tab. 5. The interval enclosures to the solution of (21) obtained with the optimal step size for the first integration step

Step	Y	Width	Exact solution
500	$[1.2111440365720224\text{E+}0000,$ $1.2111440365720227\text{E+}0000]$	$\approx 1.78\text{E-}16$	$1.2111440365720225\text{E+}0000$
1000	$[1.4668698773229724\text{E+}0000,$ $1.4668698773239729\text{E+}0000]$	$\approx 3.96\text{E-}16$	$1.4668698773239726\text{E+}0000$
1500	$[1.7765907043480633\text{E+}0000,$ $1.7765907043480640\text{E+}0000]$	$\approx 6.59\text{E-}16$	$1.7765907043480636\text{E+}0000$
2000	$[2.1517072370004459\text{E+}0000,$ $2.1517072370004470\text{E+}0000]$	$\approx 9.98\text{E-}16$	$2.1517072370004464\text{E+}0000$
2500	$[2.6060273885419534\text{E+}0000,$ $2.6060273885419549\text{E+}0000]$	$\approx 1.45\text{E-}15$	$2.6060273885419541\text{E+}0000$
2592	$[2.6995228134287445\text{E+}0000,$ $2.6995228134287462\text{E+}0000]$	$\approx 1.57\text{E-}15$	$2.6995228134287453\text{E+}0000$

References

[1] M. Berz, G. Hoffstätter, *Computation and Application of Taylor Polynomials with Interval Remainder Bounds*, Reliable Computing **4**(1), 83–97 (1998).

[2] M. Berz, K. Makino, *Performance of Taylor Model Methods for Validated Integration of ODEs*, [In:] J. Dongarra, K. Madsen, J. Wasniewski (eds.) *Applied Parallel Computing. State of the Art in Scientific Computing*, Lecture Notes in Computer Science **3732**, 65–73 (2005).

[3] J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*, John Wiley & Sons, Chichester (1987).

[4] G.F. Corliss, R. Rihm, *Validating an A Priori Enclosure Using High-Order Taylor Series*, [In:] Scientific Computing, Computer Arithmetic, and Validated Numerics, 228–238, Akademie Verlag (1996).

[5] W.H. Enright, J.D. Pryce, *Two Fortran Packages for Assessing Initial Value Methods*, ACM Transactions on Mathematical Software **13**(1), 1–27 (1987).

Tab. 6. The interval enclosures to the solution of (21) obtained with a step size changing in each integration step

Step	Y	Width	Exact solution
500	[1.2018073081170456E+0000, 1.2018073081170458E+0000]	≈ 1.08E-16	1.2018073081170458E+0000
1000	[1.4179877484171219E+0000, 1.4179877484171222E+0000]	≈ 2.17E-16	1.4179877484171221E+0000
1500	[1.6487894962346351E+0000, 1.6487894962346355E+0000]	≈ 3.26E-16	1.6487894962346353E+0000
2000	[1.8723210819363658E+0000, 1.8723210819363663E+0000]	≈ 4.49E-16	1.8723210819363661E+0000
2500	[2.1773586900207728E+0000, 2.1773586900207735E+0000]	≈ 6.39E-15	2.1773586900207732E+0000
3000	[2.5485293132332228E+0000, 2.5485293132332239E+0000]	≈ 9.12E-15	2.5485293132332234E+0000
3162	[2.6990947890180960E+0000, 2.6990947890180971E+0000]	≈ 1.01E-15	2.6990947890180967E+0000

Tab. 7. The interval enclosures to the solution of the problem (22) obtained with the optimal step size for the first integration step

Step	T	Y	Width
500	[4.0873113641944430E-0001, 4.0873113641944432E-0001]	[4.3717586653031176E+0000, 4.3717586653031183E+0000]	≈ 6.70E-16
1000	[8.1746227283888861E+0000, 8.1746227283888865E+0000]	[4.6836807485176570E+0000, 4.6836807485176585E+0000]	≈ 1.46E-15
1500	[1.2261934092583329E+0000, 1.2261934092583330E+0000]	[4.9498209108613228E+0000, 4.9498209108613252E+0000]	≈ 2.34E-15
1786	[1.4599876192902550E+0000, 1.4599876192902552E+0000]	[5.0849484688085775E+0000, 5.0849484688085804E+0000]	≈ 2.87E-15

[6] K. Gajda, M. Jankowska, A. Marciniak, B. Szyszka, *A Survey of Interval Runge-Kutta and Multistep Methods for Solving the Initial Value Problem*, [In:] R. Wyrzykowski, J. Dongarra, K. Karczewski, J. Wasniewski (eds.) *Parallel Processing and Applied Mathematics*, Lecture Notes in Computer Science **4967**, 1361–1371. Springer-Verlag, Berlin (2007).

[7] K. Gajda, A. Marciniak, B. Szyszka, *Three-and Four-Stage Implicit Interval Methods of Runge-Kutta Type*, Computational Methods in Science and Technology **6**, 41–59 (2000).

[8] E. Hairer, S.P. Norsett, G. Wanner, *Solving Ordinary Differential Equations I – Nonstiff Problems*, Springer–Verlag, Berlin (1987).

[9] R. Hammer, M. Hocks, U. Kulisch, D. Ratz, *Numerical Toolbox for Verified Computing I. Basic Numerical Problems, Theory, Algorithms, and Pascal-XSC Programs*, Springer-Verlag, Berlin (1993).

[10] E.R. Hansen, *Topics in Interval Analysis*, Oxford University Press, London (1969).

[11] K.R. Jackson, N.S. Nedialkov, *Some Recent Advances in Validated Methods for IVPs for ODEs*, Applied Numerical Mathematics **42**, 269–284 (2002).

[12] M. Jankowska, A. Marciniak, *Implicit Interval Methods for Solving the Initial Value Problem*, Computational Methods in Science and Technology **8**(1), 17–30 (2002).

[13] M. Jankowska, A. Marciniak, *On Explicit Interval Methods of Adams-Bashforth Type*, Computational Methods in Science and Technology **8**(2), 46–57 (2002).

[14] M. Jankowska, A. Marciniak, *On Two Families of Implicit Interval Methods of Adams-Moulton Type*, Computational Methods in Science and Technology **12**(2), 109–113 (2006).

[15] S.A. Kalmykov, J.I. Shokin, Z.H. Juldashev, *Solving Ordinary Differential Equations by Interval Methods [in Russian]*, Doklady AN SSSR **230**(6) (1976).

[16] A. Marciniak, *Implicit Interval Methods for Solving the Initial Value Problem*, Numerical Algorithms **37**, 241–251 (2004).

[17] A. Marciniak, *Selected Interval Methods for Solving the Initial Value Problem*, Publishing House of Poznan University of Technology, Poznan (2009). http://www.cs.put.poznan.pl/amarciniak/IMforIVP-book/IMforIVP.pdf

[18] A. Marciniak, *Interval Arithmetic Unit* (2016). http://www.cs.put.poznan.pl/amarciniak/IAUnits/IntervalArithmetic32and64.pas

[19] A. Marciniak, *Delphi Pascal Programs for Step Size Control in Interval Runge-Kutta Methods* (2017). http://www.cs.put.poznan.pl/amarciniak/VSSIRKM-Examples

[20] A. Marciniak, M.A. Jankowska, *Interval Versions for Special Kinds of Explicit Linear Multistep Methods* (in review, available from the authors).

[21] A. Marciniak, M.A. Jankowska, *Interval Versions of Milne's Multistep Methods*, Numerical Algorithms **79**(1), 87–105 (2018).

[22] A. Marciniak, M.A. Jankowska, T. Hoffmann, *On Interval Predictor-Corrector Methods*, Numerical Algorithms **77**(3), 777–808 (2017).

[23] A. Marciniak, B. Szyszka, *One-and Two-Stage Implicit Interval Methods of Runge-Kutta Type*, Computational Methods in Science and Technology **5**, 53–65 (1999).

[24] A. Marciniak, B. Szyszka, T. Hoffmann, *An Interval Version of Kuntzmann-Butcher Method for Solving the Initial Value Problem* (in review, available from the authors).

[25] R.E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs (1966).

[26] R.E. Moore, *Methods and Applications of Interval Analysis*, SIAM Society for Industrial & Applied Mathematics, Philadelphia (1979).

[27] N.S. Nedialkov, *VNODE-LP – a Validated Solver for Initial Value Problems in Ordinary Differential Equations*, Tech. Rep. CAS 06-06-NN, Department of Computing and Software, McMaster University, Hamilton (2006).

[28] N.S. Nedialkov, K.R. Jackson, G.F. Corliss, *Validated Solutions of Initial Value Problems for Ordinary Differential Equations*, Applied Mathematics and Computation **105**(1), 21–68 (1999).

[29] Y.I. Shokin, *Interval Analysis [in Russian]*, Nauka, Novosibirsk (1981).

Andrzej Marciniak was born in Poznań (Poland) in 1953. He received the M.Sc. degree in mathematics in 1977, the M.Sc. degree in astronomy in 1979 and Ph.D. degree in mathematics in 1981, all from the Adam Mickiewicz University in Poznań. In 1993 he received the Dr.Habil. degree in physics from the Nicolaus Copernicus Univeristy in Toruń (Poland) and in 2010 he received the Professor Title from the President of Poland. From 1977 to 1987 and from 2000 to 2011 he held a research position at the Faculty of Mathematics and Computer Science of the Adam Mickiewicz Univeristy, and since 1987 he has been an an assistant professor in Institute of Mathematics and then a professor of computer science at the Faculty of Computing Science of the Poznań University of Technology. From 2005 to 2008 he held the office of the President of Polish Information Processing Society. His research interests include computer programming and numerical methods, especially for solving ordinary and partial defferential equations with applications to dynamical problems. In these fields he wrote three monographs, more than 20 textbooks and a number of scientific articles.

Barbara Szyszka received the M.Sc. degree in Applied Mathematics in 1994 and Ph.D. degree in Computer Science in 2003, both from the Poznan University of Technology. From 1994 she works in the Poznan University of Technology in the Institute of Mathematics. Her fields of research interest include programming and numerical methods, in particular difference methods for solving ordinary and partial differential equations.