RIGOROUS ERROR ANALYSIS OF NUMERICAL ALGORITHMS VIA SYMBOLIC COMPUTATIONS

MARIAN MROZEK

October 10, 1997

1. Introduction.

Like the goal of combinatorial algorithms is to construct certain combinatorial objects by means of some elementary operations, the goal of numerical algorithms is to construct some real numbers by means of the four elementary arithmetic operations: addition, subtraction, multiplication and division. The difference is that the set of real numbers is uncountable and as a consequence there is no finite coding of real numbers. Thus the phrase "to construct a real number" actually means to construct a rational approximation of that number. The difference between the exact number and the constructed number is the error of the algorithm. This error is present even in the ideal case when the four arithmetic operations are performed exactly. Exact computations on rational numbers are possible but impractical because of the fast growth of the size of representations of rational numbers in course of computations. As a consequence, most implementations of numerical algorithms additionally suffer from another source of errors, called rounding or truncation errors in contrast with the intrinsic error of the numerical algorithm.

The presence of errors in numerical algorithms gives them a different flavor when compared to combinatorial algorithms, because every numerical algorithm additionally requires a suitable error analysis, which usually is not a straightforward task, especially when rigorous error bounds are needed. A common approach is to perform only some approximate error analysis and this is satisfactory for many applications. The situation is different in case of computer assisted proofs based on numerical algorithms. Here exact error bounds are absolutely necessary. Since computer assisted proofs in analysis are more and more common (see [3, 4, 5, 6, 7, 8, 9]), a natural consequence is growing demand for rigorous error analysis of numerical algorithms. Such an analysis must take into account both kinds of errors: the rounding error and the intrinsic error. A method of handling the rounding error, known as the interval arithmetic (see [1]), was introduced already in sixties. The method is easily

Research supported by KBN, Grant 0449/P3/94/06.

implementable and automatically takes care of all rounding errors in course of computations. The method has one disadvantage: its implementations are rather slow. This is partly due to the nature of the method (numbers replaced by intervals) and partly because the optimization routines of the available compilers are not prepared to work with intervals.

The main problem with the intrinsic error is that even if explicit formulas for that error are available, they are difficult to evaluate in concrete problems. A good example is the Runge-Kutta method used to construct soultions of initial value problems in ODE's (see [2]). It is well known that the error bound of the p-th order Runge-Kutta method is a constant times the p-th power of the step of the method. What is difficult is the estimation of the constant.

The aim of this paper is to propose rigorous error analysis of both the intrinsic and rounding errors based on symbolic computations. Although our main object of interest are algorithms for ODE's, the presented method is widely applicable in analysis. The method was successfuly used in [6, 7] in a computer assisted proof of chaos in the Lorenz equaitons.

Unlike the interval analysis, the presented method is disjoint form the algorithm itself. This means that there is no need to rewrite the existing software and in particular the algorithms do not slow down. What is even more important, error estimates for a prescribed set of inputs may be obtained even before the algorithm itself is run. This is espacially convenient if the numerical algorithm is to be run several times for many similar inputs and the requested error bound must be in an a priori prescribed limit. In such a situation one can experiment with error bounds of various settings of the algorithm and the algorithm is run only after a setting with satisfactory error bounds is found.

One should mention that the rounding error bounds obtained this way are in general not as good as those produced by the interval arithmetic. However, this is irrelevant if rounding errors are small when compared to the intrinsic error, which is usually the case.

The organisation of the paper is as follows. In Section 2 we introduce representable numbers and in Section 3 we discuss binary orders of magnitude. The following section is devoted to representable arithmetic. An auxiliary function Γ is introduced in Section 5. Arithmetic expressions are dealt with in Section 6 and rounding error bounds for arithmetic expressions are discussed in Section 7. Section 8 presents error bounds for Runge-Kutta methods. A program in MATHEMATICA evaluating the error bound is introduced in Section 9. The last section contains a sample error evaluation for the Lorenz system.

Throughout the paper $\mathbb{R}$ and $\mathbb{Z}$ denote respectively the set of real numbers and the set of integers. These sets supplemented by positive and negative infinity are denoted by $\bar{\mathbb{R}}$ and $\bar{\mathbb{Z}}$. In the sequel it will be convenient to treat the minimum and maximum of two numbers $x, y \in \bar{\mathbb{R}}$ as a binary operation on $\bar{\mathbb{R}}$ denoted by $x \wedge y := \min(x, y)$

and $x \vee y := \max(x, y)$ respectively. The following proposition is an easy excercise.

Proposition 1.1. *Assume $a, b, a', b' \in \mathbb{R}$. Then*

$$ab \geq 0 \;\Rightarrow\; |a - b| \leq |a| \vee |b| \tag{1}$$

$$|(a \wedge b) - (a' \wedge b')| \leq |a - a'| \vee |b - b'| \tag{2}$$

$$|(a \vee b) - (a' \vee b')| \leq |a - a'| \vee |b - b'| \tag{3}$$

2. Representable numbers.

As we mentioned in the introduction, most implementations of numerical algorithms work on a certain finite subset of real numbers and all other real numbers are approximated (represented) by some numbers in this set. For the purposes of this paper it is convenient to formalize these ideas as follows.

Definition 2.1. $(\hat{\mathbb{R}}, \langle\cdot\rangle)$ *is a* representation *of $\bar{\mathbb{R}}$ if $\hat{\mathbb{R}} \subset \bar{\mathbb{R}}$ is a finite subset and $\langle\cdot\rangle : \bar{\mathbb{R}} \to \hat{\mathbb{R}}$ is a function such that for any $x, y \in \bar{\mathbb{R}}$ the following conditions are satisfied*

$$0, -\infty, +\infty \in \hat{\mathbb{R}} \tag{4}$$

$$x \in \hat{\mathbb{R}} \;\Rightarrow\; -x \in \hat{\mathbb{R}} \tag{5}$$

$$x \in \hat{\mathbb{R}} \;\Rightarrow\; x = \langle x \rangle \tag{6}$$

$$x \leq y \;\Rightarrow\; \langle x \rangle \leq \langle y \rangle \tag{7}$$

The number $\langle x \rangle$ is called *the representaton* of x. We define the *lower* and *upper representations* of x by

$$\downarrow\!(x) := \max\{\, y \in \hat{\mathbb{R}} \mid y \leq x \,\}$$
$$\uparrow\!(x) := \min\{\, y \in \hat{\mathbb{R}} \mid y \geq x \,\}$$

The following two properties follow immediately from the above definitions and property (7).

$$\downarrow\!(x) \leq x \leq \uparrow\!(x) \tag{8}$$

$$\langle x \rangle \in \{\, \downarrow\!(x), \uparrow\!(x) \,\} \tag{9}$$

A common way of implementing a representation of $\bar{\mathbb{R}}$ in hardware is as follows. Let $D \in \mathbb{N}, N_-, N_+ \in \mathbb{Z}$ and put $\delta := 2^{-D}$. Define

$$\mu_D := \{1 + k/2^D \mid k = 0, 1, 2, \ldots 2^D - 1\} \subset [1, 2)$$
$$N : \mathbb{R} \setminus \{0\} \ni x \to E(\log_2 |\, x \,|) \in \mathbb{Z}$$
$$m : \mathbb{R} \setminus \{0\} \ni x \to |\, x \,| \,/ 2^{N(x)} \in [1, 2)$$

Then for all $x \neq 0$

$$2^{N(x)} \leq \mid x \mid < 2^{N(x)+1}$$
$$N(-x) = N(x),\ m(-x) = m(x)$$
$$x = \operatorname{sgn}(x) m(x) 2^{N(x)}$$

Define in turn

$$\bar{N} : \hat{\mathbb{R}} \setminus \{0\} \ni x \to \begin{cases} -\infty & N(x) < N^- \\ +\infty & N(x) > N^+ \\ N(x) & \text{otherwise} \end{cases} \tag{10}$$

$$\bar{m} : \mathbb{R} \setminus \{0\} \ni x \to 2^{-D} E(2^D m(x) + 1/2) \in \mu_D \tag{11}$$

$$\langle x \rangle := \begin{cases} \operatorname{sgn}(x) \bar{m}(x) 2^{\bar{N}(x)} & \text{for } x \neq 0, \\ 0 & \text{for } x = 0. \end{cases} \tag{12}$$

$$\hat{\mathbb{R}} := \{ x \in \mathbb{R} \mid \langle x \rangle = x \} \tag{13}$$

It is an easy excercise to verify that the pair $(\hat{\mathbb{R}}, \langle \cdot \rangle)$ satisfies all properties required in Definition 2.1.

In the sequel $(\hat{\mathbb{R}}, \langle \cdot \rangle)$ is taken to be a fixed representation of $\bar{\mathbb{R}}$ and the elements of $\hat{\mathbb{R}}$ are referred to as *representable numbers.*

Proposition 2.2. *Assume* $a \in \mathbb{R}$ *and* $r_1, r_2 \in \hat{\mathbb{R}}$. *Then*

$$r_1 \leq |a| \leq r_2 \ \Rightarrow\ r_1 \leq |\langle a \rangle| \leq r_2.$$

Proof: If $a \geq 0$ then $r_1 \leq a \leq r_2$, $\langle a \rangle \geq 0$, and by (7) and (6) $r_1 \leq \langle a \rangle \leq r_2$, i.e. $r_1 \leq |\langle a \rangle| \leq r_2$. If $a \leq 0$ then $-r_2 \leq a \leq -r_1$, $|\langle a \rangle| = -\langle a \rangle$ and $-r_2 \leq \langle a \rangle \leq -r_1$, i.e. again $r_1 \leq |\langle a \rangle| \leq r_2$. QED

3. Binary orders of magnitude

A *binary order of magnitude* is a number of the form 2^k, where $k \in \bar{\mathbb{Z}}$. The set of binary orders of magnitude will be denoted by $\mathbb{R}_{\text{bom}}$.

The *(upper) binary order of magnitude* of x is given by

$$\lceil x \rceil := \min \{ y \in \mathbb{R}_{\text{bom}} \mid y \geq |x| \}$$

The following proposition summarizes the properties of the upper binary order of magnitude.

Proposition 3.1. *Assume* $a, b \in \bar{\mathbb{R}}$. *Then*

$$|a| \leq \lceil a \rceil \tag{14}$$

$$0 \leq |a| \leq |b| \;\Rightarrow\; \lceil a \rceil \leq \lceil b \rceil \tag{15}$$

$$\lceil a \rceil \in \hat{\mathbb{R}} \;\Rightarrow\; \lceil \langle a \rangle \rceil \leq \lceil a \rceil \tag{16}$$

$$\lceil a \pm b \rceil \leq 2(\lceil a \rceil \vee \lceil b \rceil) \tag{17}$$

$$\lceil a \times b \rceil \leq \lceil a \rceil \times \lceil b \rceil \tag{18}$$

$$\lceil a \vee b \rceil \leq \lceil a \rceil \vee \lceil b \rceil \tag{19}$$

$$\lceil a \wedge b \rceil \leq \lceil a \rceil \vee \lceil b \rceil \tag{20}$$

Proof: Properties (14) and (15) are obvious. To see (16) consider first the case $a \geq 0$. Then $0 \leq a = |a| \leq \lceil a \rceil$ and by (7) and (6) $0 \leq \langle a \rangle \leq \langle \lceil a \rceil \rangle = \lceil a \rceil$, i.e. $\lceil \langle a \rangle \rceil \leq \lceil a \rceil$. If $a \leq 0$, then $a \geq -\lceil a \rceil$, $0 \geq \langle a \rangle \geq \langle -\lceil a \rceil \rangle = -\lceil a \rceil$ and $|\langle a \rangle| = -\langle a \rangle \leq \lceil a \rceil$. Thus $\lceil \langle a \rangle \rceil \leq \lceil a \rceil$. This proves (16). To prove (17) observe that $|a \pm b| \leq |a| + |b| \leq \lceil a \rceil + \lceil b \rceil \leq 2(\lceil a \rceil \vee \lceil b \rceil) \in \mathbb{R}_{\mathrm{bom}}$. This means that also $\lceil a \pm b \rceil \leq 2(\lceil a \rceil \vee \lceil b \rceil)$. The proof of the remaining properties is similar. QED

4. Representable arithmetic

By a standard arithmetic operation we mean any of the operations $+, -, \times, /$, the minimum and maximum operations $\wedge, \vee$ and the five relations $=, <, >, \leq, \geq$ treated as functions $\mathbb{R} \times \mathbb{R} \to \{0, 1\} \subset \mathbb{R}$. Assume $\diamond : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is an arithmetic operation. $\langle \diamond \rangle : \hat{\mathbb{R}} \times \hat{\mathbb{R}} \to \hat{\mathbb{R}}$ is a *proper implementation* of $\diamond \in \{+, -, \times, /, \wedge, \vee\}$ if for all $x, y \in \hat{\mathbb{R}}$

$$x \langle \diamond \rangle y = \langle x \diamond y \rangle. \tag{21}$$

In particular, the above formula means that a proper implementation of the minimum and maximum operations on two representable numbers, as well as the relation operators are just exact.

A representation of $\bar{\mathbb{R}}$ equipped with proper implementations of standard arithmetic operations will be called a representable arithmetic. One can easily check that a 64-bit hardware arithmetic satisfying the IEEE standards (see [10]) is a representable arithmetic with accuracy 2^{-54} over the intervals $[2^{-1022}, 2^{1023}], [-2^{1023}, -2^{-1022}]$.

If $a, b \in \mathbb{R}$ are such that $a \leq b$ and $0 \notin [a, b]$, then the *accuracy* of $(\hat{\mathbb{R}}, \langle \cdot \rangle)$ over the interval $[a, b]$ is defined by

$$\delta(\hat{\mathbb{R}}, \langle \cdot \rangle, [a, b]) := \sup \left\{ \frac{|x - \langle x \rangle|}{\lceil x \rceil} \mid x \in [a, b] \right\}.$$

Observe that the standard hardware floating point representation of real numbers in the form $x = \pm m 2^N$ discussed above, where $m \in [1, 2)$ is a D-digit binary mantissa, leads to the accuracy not greater then $2^{-(D+2)}$ over any interval $[a, b]$ such that $a, b \in \hat{\mathbb{R}}$ and $[a, b] \cap \{0, -\infty, \infty\} = \emptyset$.

We will say that $(\hat{\mathbb{R}}, \langle\cdot\rangle, u^-, u^+, \delta)$ is a *strong representation* of $\bar{\mathbb{R}}$ if $(\hat{\mathbb{R}}, \langle\cdot\rangle)$ is a representation of $\bar{\mathbb{R}}$ and $\delta, u_-, u_+ \in \hat{\mathbb{R}}$ are positive numbers such that

$$x, z \in \hat{\mathbb{R}} \cap \mathbb{R}_{\mathrm{bom}},\ x \le y \le z,\ y \in \mathbb{R}_{\mathrm{bom}} \ \Rightarrow\ y \in \hat{\mathbb{R}} \tag{22}$$

$$u_-/u_+ \le \delta \le 1 \tag{23}$$

$$u_-/\delta \in \hat{\mathbb{R}} \tag{24}$$

$$\hat{\mathbb{R}} \text{ has accuracy } \delta \text{ over } [u^-, u^+] \text{ and } [-u^+, -u^-] \tag{25}$$

A strong representation equipped with a representable arithmetic will be called a *strong representable arithmetic*. Again, it is easy to check that a hardware arithmetic complying to the 64-bit IEEE standard becomes a strong representable arithmetic with $\delta := 2^{-54}, u^- := 2^{-1022}, u^+ := 2^{1023}$.

5. Function Gamma

Typically, a final error bound is given as a relative error with respect to the absolute value of the final result. However, since the machine accuracy drops rapidly close to zero and infinity, it is convenient to modify the absolute value close to zero and infinity by means of the following function. Given a strong representation $(\hat{\mathbb{R}}, \langle\cdot\rangle, u^-, u^+, \delta)$ define

$$\Gamma(x) := \begin{cases} \lceil x \rceil & \text{if } u_-/\delta \le |x| \le u_+ \\ u_-/\delta & \text{if } |x| < u_-/\delta \\ \infty & \text{if } |x| > u_+ \end{cases}$$

Proposition 5.1. *Assume $a, b \in \bar{\mathbb{R}}$. Then*

$$|a| \le |b| \ \Rightarrow\ \Gamma(a) \le \Gamma(b) \tag{26}$$

$$\Gamma(\langle a \rangle) \le \Gamma(a) \tag{27}$$

$$|a| \le \lceil a \rceil \le \Gamma(a) \tag{28}$$

$$|a - \langle a \rangle| \le \delta\Gamma(a) \tag{29}$$

$$\Gamma(a \pm b) \le \Gamma(\Gamma(a) + \Gamma(b)) \tag{30}$$

$$\Gamma(a \times b) \le \Gamma(\Gamma(a) \times \Gamma(b)) \tag{31}$$

$$\Gamma(a \wedge b) \le \Gamma(\Gamma(a) \vee \Gamma(b)) \tag{32}$$

$$\Gamma(a \vee b) \le \Gamma(\Gamma(a) \vee \Gamma(b)) \tag{33}$$

Proof: To prove (26) first observe that the property is obvious when $|b| > u^+$ or $|a| < u^-/\delta$. When $u^-/\delta \le |a| \le |b| \le u^+$ then the property follows from (15).

Property (27) is obvious when $|a| > u^+$. If $|a| < u^-/\delta$ then we get from (7) that also $|\langle a \rangle| < u^-/\delta$, hence $\Gamma(\langle a \rangle) = u^-/\delta = \Gamma(a)$. If $u^-/\delta \le |a| \le u^+$ then by Proposition 2.2 $u^-/\delta \le |\langle a \rangle| \le u^+$, i.e. $\Gamma(\langle a \rangle) = \lceil \langle a \rangle \rceil \le \lceil a \rceil = \Gamma(a)$.

Property (28) is straightforward. To prove (29) first observe that the property is obvious when $|a| > u^+$. If $u^- \leq |a| \leq u^+$ then it follows from the definition of the strong representation that $|a - \langle a \rangle| \leq \delta\lceil a \rceil \leq \delta\Gamma(a)$. It remains to consider the case $|a| < u^-$. Then $|\langle a \rangle| \leq u^-$ and $\Gamma(a) = u^-/\delta$. Since $a\langle a \rangle \geq 0$, it follows from property (1) that $|a - \langle a \rangle| \leq |a| \vee |\langle a \rangle| \leq u^- = \delta \frac{u^-}{\delta} = \delta\Gamma(u)$. This proves property (29).

To prove (30) first observe that the property is obvious if $|a \pm b| < u^-/\delta$. If $|a \pm b| > u^+$ then either $|a| > u^+/2$ or $|b| > u^+/2$. In every case we have

$$2(\Gamma(a) \vee \Gamma(b)) \geq 2(|a| \vee |b|) > u^+,$$

which implies that $\Gamma(2(\Gamma(a) \vee \Gamma(b))) = \infty$. Thus it remains to consider the case $u^-/\delta \leq |a + b| \leq u^+$. Then

$$\Gamma(a + b) = \lceil a + b \rceil \leq 2(\lceil a \rceil \vee \lceil b \rceil) \leq 2(\Gamma(a) \vee \Gamma(b)) \leq \Gamma(2(\Gamma(a) \vee \Gamma(b)))$$

The proof of the remaining properties is similar. QED

6. Arithmetic expressions

We want to study the evaluation of arithmetic expressions via considering a certain language. For this purpose we fix a countably infinite set of symbols $\mathcal{V} = \{X_1, X_2, \dots\}$, called *variables*, and define a *word* to be an arbitrary finite sequence of symbols in $\mathcal{V} \cup \hat{\mathbb{R}} \cup \{\ (\ ,\ )\ , +, -, \times, \wedge, \vee\}$. If w is a word then $|w|$ will stand for the length of w, i.e. the number of elements in w. The set of *(arithmetic) expressions* $\mathcal{W}$ is defined (for the purposes of this paper) as the smallest set of words satisfying the following two properties

$$x \in \mathcal{V} \cup \hat{\mathbb{R}} \ \Rightarrow\ x \in \mathcal{W}$$

$$w_1, w_2 \in \mathcal{W} \ \Rightarrow\ (w_1 + w_2), (w_1 - w_2), (w_1 \wedge w_2), (w_1 \vee w_2), (w_1 \times w_2) \in \mathcal{W}$$

The following proposition is an easy excercise.

Proposition 6.1. *If $w \in \mathcal{W}$ then exactly one of the following possibilities holds*

(i) $w = X_i$ for some $X_i \in \mathcal{V}$

(ii) $w = s$ for some $s \in \hat{\mathbb{R}}$

(iii) there exist $w_1, w_2 \in \mathcal{W}$ such that $w = (w_1 + w_2)$ or $w = (w_1 - w_2)$ or $w = (w_1 \times w_2)$ or $w = (w_1 \wedge w_2)$ or $w = (w_1 \vee w_2)$

Assume w is an expression. Let $n \in \mathbb{N}$ be large enough so that all variables appearing in w are among $X_1, \dots X_n$. Let $x_1, \dots, x_n$ be real numbers. We define recursively $w(x_1, \dots x_n)$, the evaluation of w at $x_1, \dots, x_n$ by

$$(34)\quad w(x_1, \dots x_n) := \begin{cases} s & \text{if } w = s, \text{ where } s \in \hat{\mathbb{R}} \\ x_i & \text{if } w = X_i, \text{ where } X_i \in \mathcal{V} \\ w_1(x_1, \dots x_n) \diamond w_2(x_1, \dots x_n) & \text{if } w = (w_1 \diamond w_2) \end{cases}$$

where $\diamond \in \{+, -, \times, \vee, \wedge\}$.

Similarly we define recursively $\langle w\rangle(x_1,\dots x_n)$, the machine evaluation of w at $x_1,\dots,x_n$ by

$$\langle w\rangle(x_1,\dots x_n) := \tag{35}$$
$$\begin{cases} s & \text{if } w = s,\ \text{where } s \in \hat{\mathbb{R}} \\ \langle x_i\rangle & \text{if } w = X_i,\ \text{where } X_i \in \mathcal{V} \\ \langle w_1\rangle(x_1,\dots x_n)\langle\diamond\rangle\langle w_2\rangle(x_1,\dots x_n) & \text{if } w = (w_1 \diamond w_2). \end{cases}$$

7. Rounding error bounds

For a given expression w we define recursively functions $H_w^*, H_w : \bar{\mathbb{R}}^n \to \bar{\mathbb{R}}$ representing bounds on absolute value by

$$H_w^*(x_1,\dots x_n) :=$$
$$\begin{cases} |s| & \text{if } w = s, \text{ where } s \in \hat{\mathbb{R}} \\ |x_i| & \text{if } w = X_i, \text{ where } X_i \in \mathcal{V} \\ H_{w_1}^*(x_1,\dots x_n) + H_{w_2}^*(x_1,\dots x_n) & \text{if } w = (w_1 \pm w_2) \\ H_{w_1}^*(x_1,\dots x_n) \vee H_{w_2}^*(x_1,\dots x_n) & \text{if } w = (w_1 \wedge w_2) \text{ or } w = (w_1 \vee w_2) \\ H_{w_1}^*(x_1,\dots x_n) \times H_{w_2}^*(x_1,\dots x_n) & \text{if } w = (w_1 \times w_2) \end{cases}$$

$$H_w(x_1,\dots x_n) :=$$
$$\begin{cases} \Gamma(s) & \text{if } w = s, \text{ where } s \in \hat{\mathbb{R}} \\ \Gamma(x_i) & \text{if } w = X_i, \text{ where } X_i \in \mathcal{V} \\ \Gamma(2(H_{w_1}(x_1,\dots x_n) \vee H_{w_2}(x_1,\dots x_n))) & \text{if } w = (w_1 \pm w_2) \\ H_{w_1}(x_1,\dots x_n) \vee H_{w_2}(x_1,\dots x_n) & \text{if } w = (w_1 \wedge w_2) \text{ or } w = (w_1 \vee w_2) \\ \Gamma(H_{w_1}(x_1,\dots x_n) \times H_{w_2}(x_1,\dots x_n)) & \text{if } w = (w_1 \times w_2) \end{cases}$$

Similarly we define a function $\epsilon_w : \bar{\mathbb{R}}^n \to \bar{\mathbb{R}}$ representing the relative evaluation error by

$$\epsilon_w(\eta_1,\dots \eta_n) :=$$
$$\begin{cases} 0 & \text{if } w = s, \text{ where } s \in \hat{\mathbb{R}} \\ \eta_i & \text{if } w = X_i, \text{ where } X_i \in \mathcal{V} \\ \frac{\epsilon_{w_1}(\eta_1,\dots\eta_n)}{2} + \frac{\epsilon_{w_2}(\eta_1,\dots\eta_n)}{2} + \delta & \text{if } w = (w_1 \pm w_2) \\ \epsilon_{w_1}(\eta_1,\dots \eta_n) \vee \epsilon_{w_2}(\eta_1,\dots \eta_n) & \text{if } w = (w_1 \wedge w_2) \text{ or } w = (w_1 \vee w_2) \\ \epsilon_{w_1}(\eta_1,\dots \eta_n) + \epsilon_{w_2}(\eta_1,\dots \eta_n) + \delta & \text{if } w = (w_1 \times w_2) \end{cases}$$

We will say that a function $\psi(x_1,\dots x_n)$ is *monotone* if $|x_i| \le y_i$ for $i = 1, 2, \dots n$ implies $\psi(x_1,\dots x_n) \le \psi(y_1,\dots y_n)$. The following theorem shows, how functions H_w and ϵ_w may be used to estimate the rounding error on machine evaluation of w.

Theorem 7.1. *The functions H^*_w and H_w are monotone and if $x_1, \dots, x_n \in \bar{\mathbb{R}}$ are such that*

$$|x_i| \le N_i \quad \textit{for } i = 1, \dots n \tag{36}$$

$$|x_i - \langle x_i \rangle| \le \eta_i \times N_i, \quad \textit{for } i = 1, \dots n \tag{37}$$

for some $N_1, \dots N_n, \eta_1, \dots, \eta_n \in \mathbb{R}^+$, then

$$|w(x_1, \dots x_n)| \le H^*_w(N_1, \dots, N_n), \tag{38}$$

$$|w(x_1, \dots x_n)| \le \Gamma(w(x_1, \dots x_n)) \le H_w(N_1, \dots, N_n), \tag{39}$$

$$|\langle w \rangle(x_1, \dots x_n)| \le \Gamma(\langle w \rangle(x_1, \dots x_n)) \le H_w(N_1, \dots, N_n), \tag{40}$$

$$|w(x_1, \dots x_n) - \langle w \rangle(x_1, \dots x_n)| \le \epsilon_w(\eta_1, \dots, \eta_n) \times H_w(N_1, \dots, N_n), \tag{41}$$

Proof: The monotonicity of H_w and H^*_w is straightforward by an induction argument on the length of w. The first inequality both in (39) and in (40) follows immediately from (28). We will prove the second inequality in (39) and in (40) by induction on the length of w. Assume first that $|w| = 1$. Then either $w = X_i$ for some $X_i \in \mathcal{V}$ or $w = s$ for some $s \in \hat{\mathbb{R}}$. In the first case we have by (27)

$$\Gamma(\langle w \rangle(x_1, \dots x_n)) = \Gamma(\langle x_i \rangle) \le \Gamma(x_i) = H_w(x_1, \dots x_n) \le H_w(N_1, \dots, N_n)$$

and

$$\Gamma(w(x_1, \dots x_n)) = \Gamma(x_i) = H_w(x_1, \dots x_n) \le H_w(N_1, \dots, N_n).$$

In the second case we have

$$\Gamma(\langle w \rangle(x_1, \dots x_n)) = \Gamma(w(x_1, \dots x_n)) = \Gamma(s) = H_w(N_1, \dots, N_n)$$

This completes the proof of (39) and (40) in case $|w| = 1$. Thus assume $|w| > 1$. By Proposition 6.1 w has one of the forms given in (iii) of that proposition. Consider these cases turn by turn.

Assume first that $w = (w_1 \pm w_2)$ for some expressions w_1, w_2. Since $|w_1|, |w_2| < |w|$, we have by (21), (27), (30), (26) and the induction assumption

$$\begin{aligned}
&\Gamma(\langle w \rangle(x_1, \dots x_n)) = \\
&\quad \Gamma(\langle w_1 \rangle(x_1, \dots x_n) \langle \pm \rangle \langle w_2 \rangle(x_1, \dots x_n)) \le \Gamma(\langle w_1 \rangle(x_1, \dots x_n) \pm \langle w_2 \rangle(x_1, \dots x_n)) \le \\
&\qquad \Gamma(2(\Gamma(\langle w_1 \rangle(x_1, \dots x_n)) \vee \Gamma(\langle w_2 \rangle(x_1, \dots x_n)))) \le \\
&\qquad\quad \Gamma(2(H_{w_1}(N_1, \dots, N_n) \vee H_{w_2}(N_1, \dots, N_n))) = H_w(N_1, \dots, N_n)
\end{aligned}$$

and

$$\begin{aligned}
&\Gamma(w(x_1, \dots x_n)) = \\
&\quad \Gamma(w_1(x_1, \dots x_n) \pm w_2(x_1, \dots x_n)) \le \Gamma(2(\Gamma(w_1(x_1, \dots x_n)) \vee \Gamma(w_2(x_1, \dots x_n)))) \le \\
&\qquad \Gamma(2(H_{w_1}(N_1, \dots, N_n) \vee H_{w_2}(N_1, \dots, N_n))) = H_w(N_1, \dots, N_n),
\end{aligned}$$

which shows that (39) and (40) are satisfied in case $w = (w_1 \pm w_2)$.

Assume in turn that $w = (w_1 \times w_2)$ for some expressions w_1, w_2. Again we have by (21), (27), (31), (26) and the induction assumption

$$\begin{aligned}
&\Gamma(\langle w\rangle(x_1,\dots x_n)) = \\
&\quad\Gamma(\langle w_1\rangle(x_1,\dots x_n)\langle\times\rangle\langle w_2\rangle(x_1,\dots x_n)) \le \Gamma(\langle w_1\rangle(x_1,\dots x_n)\times\langle w_2\rangle(x_1,\dots x_n)) \le \\
&\qquad\Gamma(\Gamma(\langle w_1\rangle(x_1,\dots x_n))\times\Gamma(\langle w_2\rangle(x_1,\dots x_n))) \le \\
&\qquad\quad\Gamma(H_{w_1}(N_1,\dots,N_n)\times H_{w_2}(N_1,\dots,N_n)) = H_w(N_1,\dots,N_n)
\end{aligned}$$

and

$$\begin{aligned}
&\Gamma(w(x_1,\dots x_n)) = \\
&\quad\Gamma(w_1(x_1,\dots x_n)\times w_2(x_1,\dots x_n)) \le \Gamma(\Gamma(w_1(x_1,\dots x_n))\times\Gamma(w_2(x_1,\dots x_n))) \le \\
&\qquad\Gamma(H_{w_1}(N_1,\dots,N_n)\times H_{w_2}(N_1,\dots,N_n)) = H_w(N_1,\dots,N_n),
\end{aligned}$$

which shows that (39) and (40) are satisfied in case $w = (w_1 \times w_2)$.

It remains to consider the case $w = (w_1 \diamond w_2)$ with $\diamond \in \{\wedge, \vee\}$. We have then by (27), (32), (33), (26) and the induction assumption that

$$\begin{aligned}
&\Gamma(\langle w\rangle(x_1,\dots x_n)) = \\
&\Gamma(\langle w_1\rangle(x_1,\dots x_n)\diamond\langle w_2\rangle(x_1,\dots x_n)) \le \Gamma(\langle w_1\rangle(x_1,\dots x_n))\vee\Gamma(\langle w_2\rangle(x_1,\dots x_n)) \le \\
&\qquad H_{w_1}(N_1,\dots,N_n)\vee H_{w_2}(N_1,\dots,N_n) = H_w(N_1,\dots,N_n)
\end{aligned}$$

and

$$\begin{aligned}
&\Gamma(w(x_1,\dots x_n)) = \\
&\quad\Gamma(w_1(x_1,\dots x_n)\diamond w_2(x_1,\dots x_n)) \le \Gamma(w_1(x_1,\dots x_n))\vee\Gamma(w_2(x_1,\dots x_n)) \le \\
&\qquad H_{w_1}(N_1,\dots,N_n)\vee H_{w_2}(N_1,\dots,N_n) = H_w(N_1,\dots,N_n).
\end{aligned}$$

This completes the proof of (39) and (40).

The proof of (38) goes along the same lines as the proof of (39), hence we leave it as an excercise.

The proof of (41) is again by induction on the length of w. Assume that $|w| = 1$. If $w = X_i$ for some $X_i \in \mathcal{V}$ then

$$|w(x_1,\dots x_n) - \langle w\rangle(x_1,\dots x_n)| = |x_i - \langle x_i\rangle| \le \eta_i N_i = \epsilon_w(\eta_1,\dots\eta_n)H_w(N_1,\dots N_n),$$

whereas if $w = s$ for some $s \in \hat{\mathbb{R}}$ then

$$|w(x_1,\dots x_n) - \langle w\rangle(x_1,\dots x_n)| = |s - \langle s\rangle| = 0 = \epsilon_w(\eta_1,\dots\eta_n)H_w(N_1,\dots N_n),$$

which proves (41) in case $|w| = 1$. Assume $|w| > 1$. We again consider various forms of w. Consider first the case $w = (w_1 \pm w_2)$. It follows from (28), (29), (30), (26) and the induction assumption that

$$
\begin{aligned}
&|w(x_1,\dots x_n)-\langle w\rangle(x_1,\dots x_n)|\le\\
&\quad|(w_1(x_1,\dots x_n)\pm w_2(x_1,\dots x_n))-(\langle w_1\rangle(x_1,\dots x_n)\pm\langle w_2\rangle(x_1,\dots x_n))+\\
&\quad(\langle w_1\rangle(x_1,\dots x_n)\pm\langle w_2\rangle(x_1,\dots x_n))-\langle w_1\rangle(x_1,\dots x_n)\langle\pm\rangle\langle w_2\rangle(x_1,\dots x_n)|\le\\
&\quad|w_1(x_1,\dots x_n)-\langle w_1\rangle(x_1,\dots x_n)|+|w_2(x_1,\dots x_n)-\langle w_2\rangle(x_1,\dots x_n)|+\\
&\quad|(\langle w_1\rangle(x_1,\dots x_n)\pm\langle w_2\rangle(x_1,\dots x_n))-\langle\langle w_1\rangle(x_1,\dots x_n)\pm\langle w_2\rangle(x_1,\dots x_n)\rangle|\le\\
&\quad\epsilon_{w_1}(\eta_1,\dots\eta_n)H_{w_1}(N_1,\dots N_n)+\epsilon_{w_2}(\eta_1,\dots\eta_n)H_{w_2}(N_1,\dots N_n)+\\
&\quad\delta\Gamma(\langle w_1\rangle(x_1,\dots x_n)\pm\langle w_2\rangle(x_1,\dots x_n))\le\\
&\quad(\epsilon_{w_1}(\eta_1,\dots\eta_n)+\epsilon_{w_2}(\eta_1,\dots\eta_n))\times(H_{w_1}(N_1,\dots N_n)\vee H_{w_2}(N_1,\dots N_n))+\\
&\quad\delta\Gamma(2(\Gamma(\langle w_1\rangle(x_1,\dots x_n))\vee\Gamma(\langle w_2\rangle(x_1,\dots x_n))))\le\\
&\quad(\frac{\epsilon_{w_1}(\eta_1,\dots\eta_n)}{2}+\frac{\epsilon_{w_2}(\eta_1,\dots\eta_n)}{2}+\delta)\Gamma(2(H_{w_1}(N_1,\dots N_n)\vee H_{w_2}(N_1,\dots N_n)))=\\
&\quad\epsilon_w(\eta_1,\dots\eta_n)H_w(N_1,\dots N_n).
\end{aligned}
$$

This proves (41) in case $w=(w_1\pm w_2)$. Assume in turn that $w=(w_1\times w_2)$. Then by (28), (29), (31), (26), properties (39), (40) and the induction assumption we get

$$
\begin{aligned}
&|w(x_1,\dots x_n)-\langle w\rangle(x_1,\dots x_n)|\le\\
&\quad|w_1(x_1,\dots x_n)\times w_2(x_1,\dots x_n)-\langle w_1\rangle(x_1,\dots x_n)\times w_2(x_1,\dots x_n)|+\\
&\quad|\langle w_1\rangle(x_1,\dots x_n)\times w_2(x_1,\dots x_n)-\langle w_1\rangle(x_1,\dots x_n)\times\langle w_2\rangle(x_1,\dots x_n)|+\\
&\quad|\langle w_1\rangle(x_1,\dots x_n)\times w_2|(x_1,\dots x_n)+\langle\langle w_1\rangle(x_1,\dots x_n)\langle\times\rangle\langle w_2\rangle(x_1,\dots x_n)\rangle\le\\
&\quad|w_1(x_1,\dots x_n)-\langle w_1\rangle(x_1,\dots x_n)|\times|w_2(x_1,\dots x_n)|+\\
&\quad|\langle w_1\rangle(x_1,\dots x_n)|\times|w_2(x_1,\dots x_n)-\langle w_2\rangle(x_1,\dots x_n)|+\\
&\quad\delta\times\Gamma(\langle w_1\rangle(x_1,\dots x_n)\times\langle w_2\rangle(x_1,\dots x_n))\le\\
&\quad\epsilon_{w_1}(\eta_1,\dots\eta_n)\times H_{w_1}(N_1,\dots N_n)\times H_{w_2}(N_1,\dots N_n)+\\
&\quad H_{w_1}(N_1,\dots N_n)\times\epsilon_{w_2}(\eta_1,\dots\eta_n)\times H_{w_2}(N_1,\dots N_n)+\\
&\quad\delta\times\Gamma(\Gamma(\langle w_1\rangle(x_1,\dots x_n))\times\Gamma(\langle w_2\rangle(x_1,\dots x_n)))\le\\
&\quad(\epsilon_{w_1}(\eta_1,\dots\eta_n)+\epsilon_{w_2}(\eta_1,\dots\eta_n))\times H_{w_1}(N_1,\dots N_n)\times H_{w_2}(N_1,\dots N_n)+\\
&\quad\delta\times\Gamma(\Gamma(H_{w_1}(N_1,\dots N_n)\times H_{w_2}(N_1,\dots N_n))\le\\
&\quad(\epsilon_{w_1}(\eta_1,\dots\eta_n)+\epsilon_{w_2}(\eta_1,\dots\eta_n)+\delta)\times\Gamma(H_{w_1}(N_1,\dots N_n)\times H_{w_2}(N_1,\dots N_n))=\\
&\quad\epsilon_w(\eta_1,\dots\eta_n)\times H_w(N_1,\dots N_n).
\end{aligned}
$$

It remains to consider the case $w=(w_1\diamond w_2)$, where $\diamond\in\{\wedge,\vee\}$. We have by (2),

(3) and the induction assumption

$$
\begin{aligned}
&|w(x_1,\dots x_n) - \langle w\rangle(x_1,\dots x_n)| = \\
&\qquad |w_1(x_1,\dots x_n) \diamond w_2(x_1,\dots x_n) - \langle w_1\rangle(x_1,\dots x_n) \diamond \langle w_2\rangle(x_1,\dots x_n)| \le \\
&\qquad |w_1(x_1,\dots x_n) - \langle w_1\rangle(x_1,\dots x_n)| \vee |w_2(x_1,\dots x_n) - \langle w_2\rangle(x_1,\dots x_n)| \le \\
&\qquad \epsilon_{w_1}(\eta_1,\dots \eta_n) \times H_{w_1}(N_1,\dots N_n) \vee \epsilon_{w_2}(\eta_1,\dots \eta_n) \times H_{w_2}(N_1,\dots N_n) \le \\
&\qquad (\epsilon_{w_1} \vee \epsilon_{w_2}) \times (H_{w_2}(N_1,\dots N_n) \vee H_{w_2}(N_1,\dots N_n)) = \\
&\qquad\qquad \epsilon_w(\eta_1,\dots \eta_n) \times H_w(N_1,\dots N_n).
\end{aligned}
$$

This completes the proof of (41). QED

8. Error bounds for Runge-Kutta methods

We turn now our attention to the estimation of intrinsic errors of a numerical algorithm. We illustrate our approach with the fourth order Runge-Kutta method for ordinary differential equations. Consider the initial value problem of the form

$$\frac{\partial}{\partial t}\varphi(t,x_0) = V(\varphi(t,x_0)) \tag{42}$$

$$\varphi(0,x_0) = x_0, \tag{43}$$

where $x_0 \in \mathbb{R}^d$, $V : \mathbb{R}^d \to \mathbb{R}^d$ is a vector field and $\varphi : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ is the unknown solution. Given a small $h > 0$, the goal of the algorithm is to construct a sequence of points $\{ x_1, x_2, \dots \}$ such that x_n approximates $\varphi(t_n, x_0)$ at $t_n := nh$. In Runge-Kutta methods the goal is achieved by means of a recursive formula

$$x_{n+1} = \Phi(h, x_n).$$

In the most commonly used fourth order Runge-Kutta one takes

$$\Phi(h,x) := x + \frac{h}{6}\sum_{i=0}^{3} \kappa_i k_i(h,x), \tag{44}$$

where $\kappa_0 = \kappa_3 = 1$, $\kappa_1 = \kappa_2 = 2$ and

$$
\begin{aligned}
k_0(h,x) &:= V(x) \\
k_1(h,x) &:= V(x + \frac{h}{2}k_1(h,x)) \\
k_2(h,x) &:= V(x + \frac{h}{2}k_2(h,x)) \\
k_3(h,x) &:= V(x + hk_3(h,x)).
\end{aligned}
$$

From know on we assume that the vector field V is a polynomial. Note that then also Φ is a polynomial.

It is well known that the one step error of this method is bounded by Ch^5. In particular, it follows from [2], Theorem 3.1 that for any $x \in \mathbb{R}^d$ and $h > 0$

$$||\varphi(h,x) - \Phi(h,x)||_{\sup} \le C(h,x)h^5,$$

where

$$C(h,x) := \frac{1}{120} \max_{t\in[0,1]} \bar{G}(th,x) + \frac{1}{144} \sum_{i=0}^{3} \kappa_i \max_{t\in[0,1]} K_i(th,x)$$

and

$$\bar{G}(h,x) := ||\frac{\partial^5 \varphi}{\partial t^5}(h,x)||_{\sup},$$
$$K_i(h,x) := ||\frac{\partial^4 k_i}{\partial t^4}(h,x)||_{\sup}.$$

Define recursively functions $V_n : \mathbb{R}^d \to \mathbb{R}^d$ by

$$V_1(x) := V(x)$$
$$V_{n+1} := DV_n(x)V(x)$$

and put

$$G(x) := ||V_5(x)||_{\sup},$$

where $||\cdot||_{\sup}$ stands for the supremum norm in $\mathbb{R}^d$. An easy induction argument shows that

$$\bar{G}(h,x) = G(\varphi(h,x)).$$

Let $Z \subset \mathbb{R}^d$ be a compact subset. For $s > 0$ denote

$$Z_s := \{\, x \in Z \mid \varphi([0,s],x) \subset Z \,\}.$$

and let $x^Z := (x_i^Z)_{i=1,\dots d}$, where $x_i^Z := \max\{|x_i| \mid x \in Z\}$. Define

$$E_Z(h) := \Big[\frac{1}{120} H_G^*(x^Z) + \frac{1}{144}\sum_{i=0}^{3} \kappa_i H_{K_i}^*(h,x^Z)\Big] h^5 + \epsilon_\Phi(0) H_\Phi(h,x^Z). \tag{45}$$

Theorem 8.1. *Assume the flow φ is generated by (42), where V is a polynomial vector field. Then for every $h > 0$ and $x \in Z_h$*

$$||\varphi(h,x) - \langle\Phi\rangle(h,x)||_{\sup} \;\le\; E_Z(h).$$

Proof: First observe that by the definition of Z_h, property (39) and the monotonicity of H we have for $x \in Z_h$

$$\max_{t\in[0,1]} \bar{G}(th,x) = \max_{t\in[0,1]} G(\varphi(th,x)) \le \max_{x\in Z} G(x) \le \max_{x\in Z} H_G^*(x) \le H_G^*(x^Z).$$

Similarly

$$\max_{t\in[0,1]} K_i(th,x) \le \max_{t\in[0,1]} H_{K_i}^*(th,x) \le H_{K_i}^*(h,x^Z).$$

Hence

$$C(h,x) \le \frac{1}{120} H_G^*(x^Z) + \frac{1}{144} \sum_{i-0}^{3} \kappa_i H_{K_i}^*(h, x^Z). \tag{46}$$

Now from property (41) we get

$$\begin{aligned} ||\varphi(h,x) - \langle\Phi\rangle(h,x)||_{\sup} &\le ||\varphi(h,x) - \Phi(h,x)||_{\sup} + ||\Phi(h,x) - \langle\Phi\rangle(h,x)||_{\sup} \\ &\le C(h,x)h^5 + \epsilon_\Phi(0) H_\Phi(h,x) \\ &\le \left[\frac{1}{120} H_G^*(h,x^Z) + \frac{1}{144}\sum_{i-0}^{3} \kappa_i H_{K_i}^*(h,x^Z)\right] h^5 + \epsilon_\Phi(0) H_\Phi(h,x^Z) \\ &= E_Z(h) \end{aligned} \tag{47}$$

QED

Finding the error bound E_Z is an elementary task but, typically extremly arduous or even practically unrealisable by hand even for very simple vector fileds like the vector field in the Lorenz system

$$\begin{cases} \dot{x} &= s(y-x) \\ \dot{y} &= rx - y - xz \\ \dot{z} &= xy - qz. \end{cases} \tag{48}$$

In the following sections we show how the error bound E_Z may be found by means of symbolic computations.

9. Symbolic computation of E_Z

The contents of this section set with the typewriter font in the display style constitutes a program in MATHEMATICA which finds the formula for the error bound E_Z introduced in the previous section. Of course the program may be easily implemented in any other package for symbolic computations. We begin with the definition of the machine representation of x as given by formula (12), a function testing if a given number is a representable number and the definition of the upper binary order of magnitude.

```
bilog[x_ /; NumberQ[x] && x!=0]:=Module[{n,y},n=0;y=Abs[x];
   If[x>=1,While[y>=2,y=y/2;n=n+1;],While[y<1,y=2y;n=n-1]];
   n]
dmant[x_/; NumberQ[x] && x!=0]:=
       2^(-d) Floor[2^d Abs[x]/(2^bilog[x])+1/2]
dmant[0]:=0
repr[x_ /; NumberQ[x]]:=Sign[x] dmant[x] 2^bilog[x]
exact[x_ /; NumberQ[x]]:=x==repr[x];
ubom[x_ /; NumberQ[x]]:= 2^(1+bilog[x])
```

Since the function Γ involves maximum and minimum operations, we would like to have them evaluated whenever possible. For this reason we assume the following definitions of the maximum and minimum functions, denoted respectively by Mx and Mn. Note that if a lower bound for an expression a is known to be greater or equal than an upper bound of an expression b then $a \vee b = a$. Similarly one can evaluate the minimum function.

```
SetAttributes[Mx,Orderless]
Mx[a_,b_]:=If[a>b,a,b] /; NumberQ[a] && NumberQ[b]
Mx[a_,b_]:=a /; MinEst[a]>=MaxEst[b]
Mx[x_ /; Head[x]=!=List] := x
Mx[{a_}]:=a
Mx[x_ /; Head[x]===List && Length[x]>1] := Mx[First[x],Mx[Rest[x]]]
Mx[a_,a_]:=a
Mx[x_ a_, y_ a_]:=a Mx[x,y]
Mx[a_, y_ a_]:=a Mx[1,y]
Mx[a_^k_ x_,a_^l_ y_]:=a^Min[k,l] Mx[a^(k-Min[k,l])x,a^(l-Min[k,l])y]
Mx[a_^k_,a_^l_ y_]:=a^Min[k,l] Mx[a^(k-Min[k,l]),a^(l-Min[k,l])y]
Mx[a_^k_,a_^l_]:=a^Min[k,l] Mx[a^(k-Min[k,l]),a^(l-Min[k,l])]
Mx[a_ x_,a_^l_ y_]:=a Mx[x,a^(l-1)y]
Mx[a_,a_^l_ y_]:=a Mx[1,a^(l-1)y]
Mx[a_ x_,a_^l_]:=a Mx[x,a^(l-1)]
Mx[a_,a_^l_]:=a Mx[1,a^(l-1)]
Mx[a_,Mx[a_,b_]]:=Mx[a,b]
Mx[Mx[a_,b_],Mx[a_,c_]]:=Mx[a,Mx[b,c]]

SetAttributes[Mn,Orderless]
Mn[a_,b_]:=If[a<b,a,b] /; NumberQ[a] && NumberQ[b]
Mn[a_,b_]:=a /; MinEst[b]>=MaxEst[a]
Mn[x_ /; Head[x]!=List] := x
Mn[{a_}]:=a
Mn[x_ /; Head[x]===List && Length[x]>1] := Mn[First[x],Mn[Rest[x]]]
Mn[a_,a_]:=a
Mn[x_ a_, y_ a_]:=a Mn[x,y]
Mn[a_, y_ a_]:=a Mn[1,y]
Mn[a_^k_ x_,a_^l_ y_]:=a^Min[k,l] Mn[a^(k-Min[k,l])x,a^(l-Min[k,l])y]
Mn[a_^k_,a_^l_ y_]:=a^Min[k,l] Mn[a^(k-Min[k,l]),a^(l-Min[k,l])y]
Mn[a_^k_,a_^l_]:=a^Min[k,l] Mn[a^(k-Min[k,l]),a^(l-Min[k,l])]
Mn[a_ x_,a_^l_ y_]:=a Mn[x,a^(l-1)y]
Mn[a_,a_^l_ y_]:=a Mn[1,a^(l-1)y]
Mn[a_ x_,a_^l_]:=a Mn[x,a^(l-1)]
Mn[a_,a_^l_]:=a Mn[1,a^(l-1)]
Mn[a_,Mn[a_,b_]]:=Mn[a,b]
Mn[Mn[a_,b_],Mn[a_,c_]]:=Mn[a,Mn[b,c]]
```

To make full use of the above definitions we need some rules to find upper and lower bounds of an expression in terms of upper and lower bounds of subexpressions. Here are the relevant rules.

```
MinEst[a_ b_]:=MinEst[a]MinEst[b]
MinEst[a_^k_]:=MinEst[a]^k
MinEst[a_]:=a /; NumberQ[a]
MaxEst[a_ b_]:=MaxEst[a]MaxEst[b]
MaxEst[a_^k_]:=MaxEst[a]^k
MaxEst[a_]:=a /; NumberQ[a]
MaxEst[Mx[a_,b_]]:=Max[MaxEst[a],MaxEst[b]]
MinEst[Mx[a_,b_]]:=Max[MinEst[a],MinEst[b]]
```

Note that $H_w(x_1, \ldots x_n)$ is always a binary order of magnitude. Thus a simplified definition of the function Γ may be assumed

```
Gam[x_]:=Mx[u,Mn[umax,x]]
```

We may know define the functions H^*_w, H_w, ϵ_w

```
Hstar[n_ /; NumberQ[n]]:=Abs[n]
Hstar[x_ /; Head[x]==Symbol]:=x
Hstar[x_+y_]:=Hstar[x]+Hstar[y]
```

```
Hstar[x_ y_]:=Hstar[x] Hstar[y]
Hstar[Power[a_,n_]]:=Power[Hstar[a],n]
Hstar[Mx[x_,y_]]:=Mx[Hstar[x],Hstar[y]]
Hstar[Mn[x_,y_]]:=Mx[Hstar[x],Hstar[y]]

H[n_ /; NumberQ[n]]:=Gam[ubom[n]]
H[x_ /; Head[x]==Symbol]:=x
H[x_+y_]:= Gam[2*Mx[H[x],H[y]]]
H[x_ y_]:=Gam[ H[x]H[y] ]
H[Power[a_,n_]]:= Gam[ H[a]H[Power[a,n-1]] ]
H[Mx[x_,y_]]:=Mx[H[x],H[y]]
H[Mn[x_,y_]]:=Mx[H[x],H[y]]

eps[n_ /; NumberQ[n]]:=If[exact[n],0,delta]
eps[x_ /; Head[x]==Symbol]:=0
eps[x_ + y_]:= eps[x]+eps[y]+delta
eps[x_ y_]:=eps[x]+eps[y]+delta/2
eps[Power[a_,n_]]:=n eps[x] + (n-1) delta/2
eps[Mx[x_,y_]]:=Mx[eps[x],eps[y]]
eps[Mn[x_,y_]]:=Mx[eps[x],eps[y]]
```

The rounding error bound given by (41) is

```
fpdeviation[a_]:=eps[Mx[a]]H[Mx[a]]
```

We turn know our attention to the intrinsic error of the fourth order Runge-Kutta method for a vector field $V : \mathbb{R}^3 \to \mathbb{R}^3$. The Runge-Kutta method itself is implemented as follows.

```
k1[h_]:=f[{x,y,z}]
k2[h_]:=f[{x,y,z} + 1/2 h k1[h]]
k3[h_]:=f[{x,y,z} + 1/2 h k2[h]]
k4[h_]:=f[{x,y,z} +     h k3[h]]
rk[h_]:={x,y,z} + (1/6)h(k1[h] + 2 k2[h] + 2 k3[h] + k4[h])
```

The coefficient C may be found as follows

```
Di[{f_,g_,h_}]:={Di[f],Di[g],Di[h]}
Di[e_]:={D[e,x],D[e,y],D[e,z]}
Der[1]:=f[{x,y,z}]
Der[n_]:=Di[Der[n-1]].f[{x,y,z}]
c1=1/120 Der[5];
c21=1/6 D[k1[h],{h,4}];
c22=1/3 D[k2[h],{h,4}];
c23=1/3 D[k3[h],{h,4}];
c24=1/6 D[k4[h],{h,4}];
c2=1/24(c21+c22+c23+c24);
c=c1+c2;
```

and its upper estimate as follows

```
$RecursionLimit=1024;
flatc=Expand[c];
flatCoef=Hstar[Mx[flatc]];
```

Thus the total error estimate is

```
TotalErrorEstimate=fpdeviation[rk[h]]+flatCoef h^5;
```

10. An example

As an example of application we consider again the Lorenz equations (48). Here is the implementation of the Lorenz vector field in MATHEMATICA.

```
 f[{x_,y_,z_}]:={s(y-x),r x - y - x z, x y - q z}
```

We assume the following bounds for the step size

```
MinEst[h]:=1/10000
MaxEst[h]:=1/1000
```

The formula for the error estimate of one step of the fourth order Runge-Kutta method in case of the Lorenz system with arbitrary coefficients takes a lot of space, so we do not present it here. Actually, the following rules

```
PolDeg[n_ /; NumberQ[n]]:=0
PolDeg[x_ /; Head[x]==Symbol]:=1
PolDeg[x_+y_]:=Mx[PolDeg[x],PolDeg[y]]
PolDeg[x_ y_]:=PolDeg[x]+PolDeg[y]
PolDeg[Power[a_,n_]]:=n PolDeg[a]
PolDeg[Mx[x_,y_]]:=Mx[PolDeg[x],PolDeg[y]]
PolDeg[Mn[x_,y_]]:=Mx[PolDeg[x],PolDeg[y]]
PolDeg[TotalErrorEstimate]
```

may be used to find that it is a polynomial of degree 23. Fixing the coefficients and assuming upper bounds for x, y, z to be 50 by means of a substitution

```
Error=TotalErrorEstimate /. {x->50,y->50,z->50,r->54,s->45,q->10}
```

we obtain the following formula for $E_Z(h)$

```
                                                                         2
     418125 h      5    31132488375   3125302734375 h   145764404296875 h
   ------------- + h Mx[----------- + --------------- + ------------------,
   1099511627776             2              4                   4

                                                                                    3
      372703910775   21516416015625 h                      2   744773630712890625 h
   Mx[------------ + ---------------- + 1332268857421875 h  + --------------------- +
            4                2                                             8

                          4                          5                          6
     4051790720947265625 h  + 79749667803955078125 h  + 491954864501953125000 h  ,

                                                         2                       3
    409190660625                           2619730128515625 h    347243452294921875 h
    ------------ + 10967653125000 h + ------------------- + --------------------- +
          4                                     2                       4

                          4
     8538374234619140625 h                           5                          6
     ---------------------- + 114496818310546875000 h  + 860921012878417968750 h ]]
                2
```

Using the following directive

```
Table[Error, {h,0.0001,0.0005,0.0001}]
```

we obtain a list of sample error bounds for steps $h = 0.0001i$, where $i = 1, 2, 3, 4, 5$.

```
         -9           -8            -7            -6            -6
{1.0721 10  , 3.35302 10  , 2.56985 10  , 1.09481 10  , 3.37894 10  }
```

Finally let us mention that although we do not present the whole formula for the error estimate depending on all three parameters of the Lorenz system and with arbitrary upper bounds for variables x, y, z, because of its length, there is no problem in using it in computations and the whole formula actually was used in the computer assisted proof of chaos in the Lorenz system presented in [6, 7]. This may raise one question: is the error estimate still rigorous when its numerical value for concrete parameters is obtained via evaluating the formula in machine arithmetic? The answer is yes, under the assumption that the machine arithmetic with roundig up is used, i.e. every representation equals the upper representation. Most present day hardware offers such machine arithmetic as one of many possibilities.

References

1. O. Aberth, *Precise Numerical Analysis*, William C. Brown Publishers, Dubuque, Iowa, 1988.
2. E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I, Nonstiff Problems*, Springer-Verlag, Berlin Heidelberg 1987.
3. B. Hassard, S. Hastings, W. Troy, and J. Zhang, A computer proof that the Lorenz equations have "chaotic" solutions, *Appl. Math. Letter*, to appear.
4. S.P. Hastings and W.C. Troy, A shooting approach to the Lorenz equations, *Bulletin (New Series) of the American Mathematical Society* 27(1992) 298-303.
5. H. Koch, A. Schenkel, P. Wittwer, Computer assisted proofs in analysis and programming in logic: a case study, Universite de Geneve, preprint.
6. K. Mischaikow and M. Mrozek, Chaos in Lorenz equations: a computer assisted proof, *Bull. AMS*, **33**(1995), 66-72.
7. K. Mischaikow and M. Mrozek, Chaos in Lorenz equations: a computer assisted proof, Part II: Details, submitted.
8. K. Mischaikow, M. Mrozek and A. Szymczak, Chaos in Lorenz equations: a computer assisted proof. Part III: The classical case, in preparation.
9. T. Rage, A. Neumaier, and C. Schlier, Rigorous verification of chaos in a molecular model, *Physical Rev. E*, **50** (1994), 2682-2688.
10. *The IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Std 754,1985.

Instytut Informatyki, Uniwersytet Jagielloński, Kraków, Poland
E-mail address: mrozek@ii.uj.edu.pl