Round-off error and exceptional behavior analysis of explicit Runge-Kutta methods

Sylvie Boldo, Florian Faissole, Alexandre Chapoutot

Round-off error and exceptional behavior analysis of explicit Runge-Kutta methods

Sylvie Boldo[1], Florian Faissole[1], and Alexandre Chapoutot[2]

[1]Inria, Université Paris-Saclay, F-91120 Palaiseau,
LRI, CNRS & Univ. Paris-Sud, F-91405 Orsay,
Email: {sylvie.boldo,florian.faissole}@inria.fr

[2]U2IS, ENSTA ParisTech, Université Paris-Saclay,
828 bd des Maréchaux, 91762 Palaiseau cedex France,
Email: alexandre.chapoutot@ensta-paristech.fr

Abstract

Numerical integration schemes are mandatory to understand complex behaviors of dynamical systems described by ordinary differential equations. Implementation of these numerical methods involve floating-point computations and propagation of round-off errors. This paper presents a new fine-grained analysis of round-off errors in explicit Runge-Kutta integration methods, taking into account exceptional behaviors, such as underflow and overflow. Linear stability properties play a central role in the proposed approach. For a large class of Runge-Kutta methods applied on linear problems, a tight bound of the round-off errors is provided. A simple test is defined and ensures the absence of underflow and a tighter round-off error bound. The absence of overflow is guaranteed as linear stability properties imply that (computed) solutions are non-increasing.

Keywords: Round-off error, Numerical integration, Runge-Kutta method, Underflow, Overflow, Linear stability.

1 Introduction

Ordinary differential equations (ODEs) are essential to model physical or biological phenomena. Such equations are the most common way to mathematically describe the temporal evolution and the variations of physical quantities. They spread their use in physics, biology, medicine, meteorology, economics and various other scientific fields [14]. As an example, the study of population dynamics is useful to model the spread of diseases in a population [33]. The Cauchy-Lipschitz Theorem ensures the existence of a unique solution for ordinary differential equations when the initial state of the studied quantity is known [29]. An ordinary differential equation together with an initial condition is called an Initial Value Problem (IVP).

In general, finding a quadrature method to solve an ODE could be difficult, *i.e.*, there is no way to exhibit an exact solution of the equation. Numerical integration methods have been designed in order to provide an approximation of the solution. They consist in partitioning a time interval by a set of discrete time instants and then to approximate the image of the solution at each point of the subdivision. Such methods are iterative: from the initial condition provided by the IVP, the approximation of the solution is built step by step, discrete point by discrete point.

There exists a wide variety of numerical integration methods. They can be *fixed step-size* or *variable step-size*, *i.e.*, time instants are equidistant from each other or not; they can be *single step* or *multi-step*, *i.e.*, they use one or several initial conditions; and they can be *explicit* or *implicit*, *i.e.*, the solution at a given time instant is only computed from initial conditions or it is computed as the solution of a fixed-point equation. Depending on its properties, a method is suitable for different classes of Initial Value Problems. For instance, explicit methods are generally not suitable to numerically solve stiff equations. That is why numerical tools provide several numerical methods: for example, Simulink is a tool to model and simulate dynamical systems widely used in the industry, it provides 13 numerical integration methods whose 11 belong to the Runge-Kutta family[1]. In this article, we focus on explicit single-fixed-step-size methods belonging to the class of Runge-Kutta methods.

[1]`https://fr.mathworks.com/help/simulink/index.html`

The main active research on numerical integration methods consists in developing new methods which are able to deal efficiently with the largest class of problems [30]. That is why most of the research is focused on increasing the order of the method while keeping the computation complexity as low as possible and on defining more stable numerical methods. There are several formal definitions of the notion of numerical stability. Intuitively, it means that the error made on the numerical solution of an ODE does not increase too much. If one considers an Initial Value Problem whose exact solution converges to zero, a method is stable if there exists a step-size such that the numerical solutions also converge to zero (see Section 2 for a more precise definition). We refer to [28] for a more comprehensive presentation of stability notions of numerical integration methods.

Numerical integration methods are built in order to provide a good approximation of the exact solution. As numerical schemes are inextricably tied to the method errors they induce, these errors are well-known and have been studied for a long time. When implemented on a machine with finite memory, numerical algorithms use computer arithmetic (floating-point arithmetic for instance) [31]. Floating-point arithmetic (FP) is known to be inaccurate and each step of computation may lead to round-off errors. As numerical integration methods are iterative methods, these round-off errors may accumulate and bias the final result. Surprisingly, numerical accuracy problems in computations have been studied even before computers when considering hand calculations [11]. Even if mathematicians are aware of the round-off errors, they are usually dismissing this problem. For instance, [36] claims the rounding errors "merely accumulate roughly in proportion to the square root of the number of steps in the calculation", but without any proof. We have provided in [7] an analysis of round-off errors of Runge-Kutta methods. However, we assumed there were neither underflow nor overflow. The contribution of this paper is to adapt the methodology of [7] taking exceptional behavior into account and to provide better error bounds:

- In [7], we postulate that overflow is easy to check *a posteriori* as the result will be a NaN or an infinity. We take here another way: we assume the initial condition of the ODE is bounded by a reasonable value and use the linear stability to prove that no overflow occurs in all the computations.
- On the contrary, underflow cannot be prevented. It has to be taken into account in the bounds on round-off errors we provide. Still, as long as no underflow occurs, we prove bounds as tight as in [7].
- We provide slightly tighter bounds using optimal bounds on relative errors studied in details by Jeannerod and Rump [26]. See Theorem 3 in Section 5.

The paper is organized as follows. Section 2 presents the Runge-Kutta methods we are interested in, together with their mathematical properties and their FP implementations. Section 3 describes a systematic methodology to bound round-off errors induced by Runge-Kutta methods taking exceptional behaviors into account. Section 4 proves that iterations of stable Runge-Kutta methods are non-increasing as long as no underflow occurs. Section 5 is devoted to overflow's handling. Then, Section 6 focuses on generic results to bound local round-off errors of Runge-Kutta methods taking underflow into account and applications on Euler, Runge-Kutta 2 and Runge-Kutta 4 methods. Section 7 provides a way to bound global round-off errors from local errors and then gives bounds on global error for the same methods. Section 8 presents related work. Finally, Section 9 concludes and gives some perspectives.

2 Runge-Kutta methods

In this section, a brief introduction on Runge-Kutta methods is recalled. A mathematical description of the integration process is in Section 2.1. Their mathematical properties and the link with linear stability are in Section 2.2. Finally, their FP counterpart is in Section 2.3.

2.1 Mathematical formulation

Runge-Kutta methods are able to solve *initial value problem* (IVP) of non-autonomous *Ordinary Differential Equations* (ODEs) defined by

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}) \quad \text{with} \quad \mathbf{y}(0) = \mathbf{y}_0 \quad \text{and} \quad t \in [0, t_{\text{end}}]. \tag{1}$$

The function $\mathbf{f} : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ is the dynamic, $\mathbf{y} \in \mathbb{R}^m$ is the vector of state variables and $\dot{\mathbf{y}}$ is the derivative of $\mathbf{y}$ with respect to time t. We shall always assume at least that $\mathbf{f}$ is globally Lipschitz in $\mathbf{y}$, so Equation (1) admits a unique solution for a given initial condition $\mathbf{y}_0$ [21]. The exact solution of Equation (1) is denoted by $\mathbf{y}(t; \mathbf{y}_0)$. Usually, a closed form of $\mathbf{y}(t; \mathbf{y}_0)$ is not computable and numerical methods to approximate it are used.

A Runge-Kutta method, starting from an initial value $\mathbf{y}_\ell$ at time t_ℓ and a finite time horizon h, the *step size*, produces an approximated solution $\mathbf{y}_{\ell+1}$ at time $t_{\ell+1}$, with $t_{\ell+1} - t_\ell = h$, of the solution $\mathbf{y}(t_{\ell+1}; \mathbf{y}_\ell)$. Furthermore, to compute $\mathbf{y}_{\ell+1}$, a Runge-Kutta method computes s evaluations of $\mathbf{f}$ at predetermined time instants. The number s is known as the number of *stages* of a Runge-Kutta method. More precisely, a Runge-Kutta method is defined by

$$\mathbf{y}_{\ell+1} = \mathbf{y}_\ell + h \sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{2}$$

with $\mathbf{k}_i$ defined by

$$\mathbf{k}_i = \mathbf{f}\left(t_\ell + c_i h, \mathbf{y}_\ell + h \sum_{j=1}^{s} a_{ij} \mathbf{k}_j \right). \tag{3}$$

The coefficients c_i, a_{ij} and b_i, for $i, j = 1, 2, \cdots, s$, fully characterize a Runge-Kutta method and they are usually summarized into a *Butcher tableau* [21]. In this paper, only *explicit* Runge-Kutta methods are considered that is the computation of the intermediate $\mathbf{k}_i$ only depends on the previous steps $\mathbf{k}_j$ for $j < i$.

The *order* of a Runge-Kutta method is p if and only if the *local truncation error* is such that

$$\mathbf{y}(t_\ell; \mathbf{y}_{\ell-1}) - \mathbf{y}_\ell = \mathcal{O}(h^{p+1}).$$

Basically, the order of a Runge-Kutta method gives an indication on the magnitude of the *method error* for one integration step.

2.2 Mathematical properties

An important mathematical property of Runge-Kutta methods is their stability properties which make them suitable to solve a wide class of problems. Starting from a stable IVP-ODE, a Runge-Kutta method is stable if there is at least an integration step which makes the numerical solution preserving the stability of the solution of IVP-ODE. The stability property depends on the class of IVP-ODEs. The most basic class of IVP-ODEs is associated to linear ODEs of the form (with $\lambda \in \mathbb{C}$):

$$\dot{y} = \lambda y. \tag{4}$$

This equation produces a stable solution, *i.e.*, convergent to zero, if and only if $\Re(\lambda) < 0$, where $\Re$ stands for the real part of λ. Note that, a very general class of linear systems is considered by using a complex value for the constant but we will focus in the following on real numbers only. While general class of linear systems is high dimensional, it is sufficient to consider scalar problems, as defined in Equation (4), to study the *linear stability* of numerical integration methods.

The application of explicit Runge-Kutta methods for the linear problem defined in Equation (4) defines the relation

$$y_{n+1} = R(h, \lambda) y_n. \tag{5}$$

Example 1. *Application of forward Euler's method produces*

$$y_{n+1} = y_n + h\lambda y_n = (1 + h\lambda) y_n = R_{Euler}(h, \lambda) y_n.$$

Example 2. *Application of RK2 (a.k.a. explicit midpoint) method on Equation* (4) *produces*

$$\begin{aligned} k_1 &= \lambda y_n \\ k_2 &= \lambda(y_n + 0.5hk_1) = (\lambda + 0.5h\lambda^2) y_n \end{aligned}$$

$$\begin{aligned} y_{n+1} &= y_n + hk_2 \\ &= (1 + h\lambda + 0.5h^2\lambda^2) y_n = R_{RK2}(h, \lambda) y_n. \end{aligned} \tag{6}$$

Example 3. *Application of RK4 method on Equation* (4) *produces*

$$\begin{aligned} k_1 &= \lambda y_n \\ k_2 &= \lambda(y_n + \frac{1}{2}hk_1) = (\lambda + \frac{1}{2}h\lambda^2) y_n \\ k_3 &= \lambda(y_n + \frac{1}{2}hk_2) = (\lambda + \frac{1}{2}h\lambda^2 + \frac{1}{4}h^2\lambda^3) y_n \\ k_4 &= \lambda(y_n + hk_3) = (\lambda + h\lambda^2 + \frac{1}{2}h^2\lambda^3 + \frac{1}{4}h^3\lambda^4) y_n \end{aligned}$$

$$
\begin{aligned}
y_{n+1} &= y_n + h(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4) \\
&= \left(1 + \lambda h + \frac{1}{2}(\lambda h)^2 + \frac{1}{6}(\lambda h)^3 + \frac{1}{24}(\lambda h)^4\right) y_n \\
&= R_{RK4}(h, \lambda) y_n. \qquad (7)
\end{aligned}
$$

Equation (5) defines a geometric sequence, which only converges if $|R(h, \lambda)| < 1$. The function R is known as the *linear stability function* of Runge-Kutta methods. It is a polynomial function for explicit Runge-Kutta methods. The region of absolute convergence of Equation (5) determines the different values of λh for which the method is stable. These regions can be determined, see [28] for more details. From now on, we will only consider linear-stable methods.

2.3 Algorithmic formulation

As in Section 2.2, we focus on differential equations for functions valued in $\mathbb{R}$. The function R, defined by Equation (5), only depends on the employed method and is purely mathematical. It is important to distinguish R and its FP counterpart $\widetilde{R}$, which is an algorithm depending on the implementation of the method.

In the rest of this article, variables mark with a tilde, *e.g.*, $\widetilde{y_n}$ or $\widetilde{\lambda}$, will represent FP numbers. Operations $\oplus$, $\ominus$, $\otimes$, $\oslash$ are the FP counterpart of the mathematical operations, correctly rounded to nearest. To denote the FP evaluation of a mathematical expression e, the notation $\circ[e]$ will be used, meaning that each operation of e inside the brackets is the FP one. For example, $\circ[a + b + c \times d]$ stands for $a \oplus (b \oplus (c \otimes d))$.

Unlike in [7], we consider here that h is not a parameter of the problem. It can indeed be chosen by the user or the program within a certain range, therefore we assume it can be chosen to be a correct FP number. This both increases the accuracy of the final result and simplifies the reasoning. An explicit integration method corresponding to Equation (5) is implemented by the algorithm

$$
\begin{cases}
\widetilde{y_0} \approx y_0 \in \mathbb{R} \\
\forall n, \widetilde{y_{n+1}} = \widetilde{R}(h, \widetilde{\lambda}, \widetilde{y_n}) \in \mathbb{R}.
\end{cases} \qquad (8)
$$

Example 4. *Let us consider the RK2 method on the linear IVP-ODE problem defined in Equation* (4)*, an associated algorithm is defined by*

$$
\widetilde{y_{n+1}} = \circ\left[\widetilde{y_n} + h\widetilde{\lambda}\widetilde{y_n} + hh\frac{1}{2}\widetilde{\lambda}\widetilde{\lambda}\widetilde{y_n}\right].
$$

Example 5. *Let us consider the RK4 method. The exact iterative relation $y_{n+1} = R_{RK4}(h, \lambda)y_n$ is given in Equation* (7)*. However, the FP implementation automatically obtained is more intricate:*

$$
\begin{aligned}
\widetilde{y_{n+1}} = \circ\Bigg[\widetilde{y_n} + \frac{h}{6}\widetilde{\lambda}\widetilde{y_n} + \frac{h}{3}\widetilde{\lambda}\widetilde{y_n} + \frac{h^2}{6}\widetilde{\lambda}^2\widetilde{y_n} + \frac{h}{3}\widetilde{\lambda}\widetilde{y_n} + \\
\frac{h^2}{6}\widetilde{\lambda}^2\widetilde{y_n} + \frac{h^3}{12}\widetilde{\lambda}^3\widetilde{y_n} + \frac{h}{6}\widetilde{\lambda}\widetilde{y_n} + \frac{h^2}{6}\widetilde{\lambda}^2\widetilde{y_n} + \\
\frac{h^3}{12}\widetilde{\lambda}^3\widetilde{y_n} + \frac{h^4}{24}\widetilde{\lambda}^4\widetilde{y_n}\Bigg]. \qquad (9)
\end{aligned}
$$

We could have manually written a simpler algorithm, but this is the one automatically generated from the basic definitions by tools such as Simulink.

The implementation is correct when the algorithm corresponds to the numerical method. It means that, if $\widetilde{R}(h, \lambda, y_n)$ was computed without rounding error, it would be mathematically equal to $R(h, \lambda)y_n$ and we aim at providing bounds on the distance between y_n and $\widetilde{y_n}$.

3 Overview of the method

In this work, we study the FP behavior of these numerical schemes. Seen as FP programs, we want to ensure their behavior regarding both underflow and overflow, and bound the round-off errors with reasonable bounds. For that, we assume an FP format with radix 2 and p bits of significand. An FP number is therefore a value equal to

$$
d_0.d_1 \cdots d_{p-1} \times 2^e
$$

with $e_{\min} \leq e \leq e_{\max}$ and d_0 is 1 except for subnormal values where $e = e_{\min}$.

We then denote the unit round-off as $u = 2^{-p}$. We denote by Ω the largest positive FP number, that is $2^{e_{\max}}\left(2 - 2^{1-p}\right)$. We denote by ξ the smallest positive normal number, that is $\xi = 2^{e_{min}}$. We denote by η the smallest positive number, that is the subnormal $\eta = 2^{e_{\min}-p+1}$. We only consider rounding to nearest, with any tie-breaking rule. For instance in *binary64* [23], we have $u_{b64} = 2^{-53}$, $\Omega_{b64} \approx 2^{1024}$, $\xi_{b64} = 2^{-1022}$ and $\eta_{b64} = 2^{-1074}$. Given a generic FP format, we want to bound the global round-off error of a Runge-Kutta method as defined above denoted by E_n:

$$E_n = \widetilde{y_n} - y_n. \tag{10}$$

For that, we will rely on a *local* round-off error ε_n, that roughly corresponds to the round-off error made at step n assuming the inputs are correct:

$$\begin{aligned} \varepsilon_0 &= |\widetilde{y_0} - y_0| \\ \forall n \in \mathbb{N}^*, \quad \varepsilon_n &= |\widetilde{R}(h, \widetilde{\lambda}, \widetilde{y_{n-1}}) - R(h,\lambda)\widetilde{y_{n-1}}|. \end{aligned} \tag{11}$$

With similar notations, our previous work proved the following result [7]: Let $C > 0$. Suppose that we have $\forall n \in \mathbb{N}^*, \varepsilon_n \leqslant Cu|\widetilde{y_{n-1}}|$ then $\forall n$,

$$|E_n| \leqslant (Cu + |R(h,\lambda)|)^n \left(\varepsilon_0 + n\frac{Cu|y_0|}{Cu + |R(h,\lambda)|}\right).$$

But it is *assumed an infinite exponent range*. Our goal is to take into account the finite range of FP. The previous bound does not hold anymore, but we try to have a similar result. The idea is to have the same result as long as underflow does not occur and have a correct but coarser result in case of underflow. One difficulty lies in a possible oscillation around the underflow threshold: one computation underflows while the next does not and the analysis must remain correct.

In order to handle all exceptional behaviors, we look into the possible ones. Given the considered computations that are only additions and multiplications with finite constant coefficients, the possible exceptions are overflow and underflow. We assume the input value $\widetilde{y_0}$ is neither an infinity, nor a NaN. It is rather easy to prevent overflow (see Section 5), as we will prove that no infinity can be produced provided a mild hypothesis on the input.

Underflow is more complicated to handle. We indeed have to modify all the error bounds from [7]. Let us recall the results and let us see how we will modify them:

- We assumed the local error was bounded by $\varepsilon_n \leqslant Cu|\widetilde{y_{n-1}}|$ with C being a small constant (such as 11 or 28). We now assume the local error is bounded by $\varepsilon_n \leqslant Cu|\widetilde{y_{n-1}}| + D\eta$, with both C and D being small constants and η the smallest subnormal number.
- We prove that this kind of formula for the local error actually holds for the common Runge-Kutta methods. This is done in Section 6 and, in particular, in Section 6.2 for Euler's, RK2 and RK4 methods.
- At last, we need to combine these local errors into a global error taking underflow into account and this is done in Theorem 12 of Section 7.

This underflow handling gives a slightly larger error bound (with an additional term $nD\eta$, which is small unless n is massive), but we want to keep the preceding result without η when no underflow occurred at all. For that, we provide a value M such that, when $|\widetilde{y_n}| \geqslant M$, the local error is bounded only by the relative error term $\varepsilon_n \leqslant Cu|\widetilde{y_{n-1}}|$ (without $D\eta$). For instance, if we were looking at a single multiplication, we have that if $|\widetilde{y_n}| \geqslant 16\xi$, then $\circ(\widetilde{y_n}/12)$ does not underflow, therefore its relative round-off error is bounded by u. The value of M is non-trivial and depends upon the scheme. Values will be given for our example methods in Section 6. More precisely, in this section, we will build the values of both C, D and M for each scheme. These values may be computed in a systematic way for any Runge-Kutta scheme, from the lemmas given in Section 6.1.

4 Non-increasing computations

As explained in Section 2.2, we only consider stable methods. From the mathematical point of view, it means that the solutions computed by these methods are non-increasing. We can prove that the FP counterparts are also non-increasing as long as no underflow occurs.

Lemma 1. *Let $C \geqslant 0, M > 0$. Suppose that:*

- $Cu + |R(h,\lambda)| \leqslant 1$;
- $\forall n \in \mathbb{N}^*, |\widetilde{y_n}| \geqslant M \Rightarrow \varepsilon_n \leqslant Cu|\widetilde{y_{n-1}}|$;

Then: $\forall n \in \mathbb{N}^*, |\widetilde{y_n}| \geqslant M \Rightarrow |\widetilde{y_{n-1}}| \geqslant |\widetilde{y_n}| \geqslant M$.

Proof. Let $n \in \mathbb{N}^*$. Suppose that $|\widetilde{y_n}| \geqslant M$. We have:
$|\widetilde{y_n}| = |\widetilde{R}(h,\widetilde{\lambda},\widetilde{y_{n-1}})| = |\widetilde{R}(h,\widetilde{\lambda},\widetilde{y_{n-1}}) - R(h,\lambda)\widetilde{y_{n-1}} + R(h,\lambda)\widetilde{y_{n-1}}|$.

By triangular inequality and by definition of local errors:

$$|\widetilde{y_n}| \leqslant |\widetilde{R}(h,\widetilde{\lambda},\widetilde{y_{n-1}}) - R(h,\lambda)\widetilde{y_{n-1}}| + |R(h,\lambda)||\widetilde{y_{n-1}}| = \varepsilon_n + |R(h,\lambda)||\widetilde{y_{n-1}}|.$$

As $|\widetilde{y_n}| \geqslant M$, $\varepsilon_n \leqslant Cu|\widetilde{y_{n-1}}|$.
Moreover, $Cu + |R(h,\lambda)| \leqslant 1$, thus:
$M \leqslant |\widetilde{y_n}| \leqslant (Cu + |R(h,\lambda)|)|\widetilde{y_{n-1}}| \leqslant |\widetilde{y_{n-1}}|$. □

Then, a simple induction provide the following result:

Lemma 2. *Let $C \geqslant 0, M > 0$. Suppose that:*

- $Cu + |R(h,\lambda)| \leqslant 1$;
- $\forall n \in \mathbb{N}^*, |\widetilde{y_n}| \geqslant M \Rightarrow \varepsilon_n \leqslant Cu|\widetilde{y_{n-1}}|$;

Then:
$\forall n \in \mathbb{N}, |\widetilde{y_n}| \geqslant M \Rightarrow |\widetilde{y_n}| \leqslant |\widetilde{y_{n-1}}| \leqslant \cdots \leqslant |\widetilde{y_1}| \leqslant |\widetilde{y_0}|$.

Thus, as long as no underflow occurs, the computed $\widetilde{y_i}$ are non-increasing. However, as soon as an underflow occurs, the computations are not necessarily non-increasing, but still stay away from overflow.

5 Overflow's handling

It is easy to check *a posteriori* if an overflow occurs. However, as we only consider stable methods, Lemma 2 shows that the computations are non-increasing as long as no underflow occurs. Thus, we prove that there is no risk of overflow if the initial value $\widetilde{y_0}$ is bounded by a reasonable value.

Let us first state an adaptation of a result proved by Jeannerod and Rump that provides an optimal bound for one FP operation [26]. Note that we also take gradual underflow into account:

Theorem 3. (Jeannerod, Rump) [26] *Let $t \in \mathbb{R}$. Then:*

$$|\circ(t) - t| \leqslant |t|\frac{u}{1+u} + \frac{\eta}{2}.$$

Our main concern is the occurrence of "local overflow": local overflow means that there is an overflow in an intermediate computation at a given step of the method. For a Runge-Kutta method described by the iterative process $\widetilde{y_{n+1}} = \bigoplus_{i=0}^{s} \widetilde{\alpha_i} \otimes \widetilde{y_n}$, *i.e.*, $\widetilde{y_{n+1}} = (\widetilde{\alpha_0} \otimes \widetilde{y_n} \oplus (\widetilde{\alpha_1} \otimes \widetilde{y_n} \oplus (\cdots \oplus (\widetilde{\alpha_s} \otimes \widetilde{y_n}))\ldots)$, local overflow means that at least one sub-sum $\bigoplus_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes \widetilde{y_n}$ (with $k \leqslant s$) overflows.

Lemma 4. Local overflow
Let $s \in \mathbb{N}$. Let $(\widetilde{\alpha_i})_{n\in\mathbb{N}}$ a sequence of FP numbers. Let y an FP number. Let $0 < V$. Suppose that:

- $\sum_{i=0}^{s} |\widetilde{\alpha_i}| \leqslant V$;
- $|y| \leqslant \frac{\Omega}{(1+(s+1)u)V}$;
- $(s+1)u \leqslant 1$;
- $\xi \leqslant 2^{e_{max}-p}$.

Then: $\forall k \in \mathbb{N}, k \leqslant s$,

$$\left| \bigoplus_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes y \right| \leq \Omega.$$

Proof. By Jeannerod-Rump [26, Th 4.1], even if underflow occurs, we have:

$$\left|\bigoplus_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes y - \sum_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes y\right| \leqslant \frac{ku}{1+u} \sum_{i=0}^{k} |\widetilde{\alpha_{i+s-k}} \otimes y|.$$

Thus:

$$\left|\bigoplus_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes y\right| \leqslant \frac{ku}{1+u} \sum_{i=0}^{k} |\widetilde{\alpha_{i+s-k}} \otimes y| + \left|\sum_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes y\right|.$$

Hence, by triangular inequalities inside the sum:

$$\left|\bigoplus_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes y\right| \leqslant \left(\frac{ku}{1+u} + 1\right) \sum_{i=0}^{k} |\widetilde{\alpha_{i+s-k}} \otimes y|.$$

Let us bound the above-mentioned sum. We note that underflow may happen in the multiplication by y. Hence, η appears for each term of the sum.

$$\sum_{i=0}^{k} |\widetilde{\alpha_{i+s-k}} \otimes y| \leqslant \sum_{i=0}^{k} \left(|y||\widetilde{\alpha_{i+s-k}}|(1+u) + \frac{\eta}{2}\right)$$

$$\leqslant \frac{\Omega}{V(1+(s+1)u)} V(1+u) + (k+1)\frac{\eta}{2} = \frac{\Omega}{1+(s+1)u}(1+u) + (k+1)\frac{\eta}{2}.$$

So, we have:

$$\left|\bigoplus_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes y\right| \leqslant \left(\frac{1+(k+1)u}{1+u}\right)\left(\frac{\Omega}{1+(s+1)u}(1+u) + (k+1)\frac{\eta}{2}\right)$$

$$< \frac{1+(k+1)u}{1+(s+1)u}\Omega + (1+(k+1)u)(k+1)\frac{\eta}{2}.$$

So, as $k \leqslant s$ and $(k+1)u \leqslant (s+1)u \leqslant 1$:

$$\left|\bigoplus_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes y\right| < \Omega + (1+(k+1)u)(k+1)\frac{\eta}{2}$$

$$\leqslant \Omega + \frac{\eta}{u} = \Omega + 2\xi \leqslant \Omega + 2 \times 2^{e_{max}-p} = \Omega + \mathrm{ulp}(\Omega).$$

Thus, as $\bigoplus_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes y$ is an FP number, $\left|\bigoplus_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes y\right| \leqslant \Omega$. □

As the sequence $(\widetilde{y_i})_{i\in\mathbb{N}}$ is non-increasing as long as no underflow occurs (see Lemma 2 in Section 4), we prevent overflow along the iterations of the method:

Theorem 5. Global overflow
Let $C \geqslant 0$, $M > 0$. Let a Runge-Kutta method described by the relation $\forall n \in \mathbb{N}, \widetilde{y_{n+1}} = \bigoplus_{i=0}^{s} \widetilde{\alpha_i} \otimes \widetilde{y_n}$. Suppose that:

- $\sum_{i=0}^{s} |\widetilde{\alpha_i}| \leqslant V$;
- $(s+1)u \leqslant 1$;
- $\forall n \in \mathbb{N}^*, |\widetilde{y_n}| \geqslant M \Rightarrow \varepsilon_n \leqslant Cu|\widetilde{y_{n-1}}|$;
- $M \leqslant \frac{\Omega}{(1+(s+1)u)V}$;
- $\xi \leqslant 2^{e_{max}-p}$.
- $|\widetilde{y_0}| \leqslant \frac{\Omega}{(1+(s+1)u)V}$.

Then, for all $n \in \mathbb{N}$, no overflow occurs in the computation of $\widetilde{y_n}$ by the Runge-Kutta method.

Proof. Let us prove the result by induction.

If $n = 0$, as $\widetilde{y_0}$ is finite, there is no overflow.

Let n a given natural number. Suppose there is no overflow up to iteration n.

If $|\widetilde{y_n}| \geqslant M$, then by Lemma 2, $|\widetilde{y_n}| \leqslant |\widetilde{y_0}| \leqslant \frac{\Omega}{(1+(s+1)u)V}$. If $|\widetilde{y_n}| < M$ then $|\widetilde{y_n}| < \frac{\Omega}{(1+(s+1)u)V}$ too. Thus, in both cases, by Lemma 4, for all $k \leqslant s$, we have:

$$\left| \bigoplus_{i=0}^{k} \widetilde{\alpha_{i+s-k}} \otimes \widetilde{y_n} \right| \leqslant \Omega.$$

Thus, no overflow occurs in the computation of $\widetilde{y_{n+1}}$. Using the induction hypothesis, it means that no overflow occurs up to iteration $n+1$ and so the result. □

Let us discuss the hypotheses of Theorem 5:

- To prevent overflow, it is sufficient to check that $|\widetilde{y_0}|$ is not too huge. Table 1 gives the values of s and the order of magnitude of V and $\frac{\Omega}{(1+(s+1)u)V}$ for classic methods in radix 2 *binary64* format. Actually, classic problems solved by Runge-Kutta methods commonly involve reasonable values. For instance, even in astrophysics, initial conditions never exceed 10^{24} [6], which is much smaller than the values exhibited in Table 1.
- Furthermore, Theorem 5 assumes $\xi \leqslant 2^{e_{max}-p}$. Actually, all useful FP formats verify this assumption. For instance, in *binary64*, $\xi = 2^{-1022}$ and $2^{e_{max}-p} = 2^{970}$.

Table 1: Values of V, s, and $\frac{\Omega}{(1+(s+1)u)V}$ for classic methods in *binary*64

Method	V	s	$\frac{\Omega}{(1+(s+1)u)V}$
Euler	3	2	$\frac{\Omega}{3+9u} \simeq 5.99 \times 10^{307}$
RK2	5	3	$\frac{\Omega}{5+20u} \simeq 3.6 \times 10^{307}$
RK4	16.5	11	$\frac{\Omega}{16.5+198u} \simeq 1.09 \times 10^{307}$

In the next sections, the contribution of gradual underflow is taken into account in round-off errors.

6 Local errors

Following our methodology (see Section 3), a first step is to bound local errors of Runge-Kutta methods. This section is devoted to local errors and is organized as follows. In Section 6.1, we give preliminary results which apply to a wide range of algorithms, taking underflow into account. Then, Section 6.2 provides bounds on local errors for both Euler's, Runge-Kutta 2 and Runge-Kutta 4 methods.

6.1 Floating-point preliminaries

Before studying particular Runge-Kutta methods, we focus on generic and quite technical results used to bound local round-off errors. First, we focus on the addition of two FP values X_1 and X_2 that both correspond to a multiple of a given value y. Second, we focus on the computation of $X_1 \oplus (X_2 \otimes y)$ with X_1 corresponding to a multiple of y. Underflow may then occur, but we retain the error bound corresponding to an unbounded exponent range as in [7], provided a few hypotheses, including the fact that y is sufficiently large.

Lemma 6. *Let $y \in \mathbb{R}$. Let $C_1, C_2, D_1, D_2 \in \mathbb{R}_+$. Let $\alpha_1, \alpha_2 \in \mathbb{R}$. Let X_1 and X_2 be FP numbers such that:*

- $|X_1 - \alpha_1 y| \leqslant C_1 u|y| + D_1\eta$;
- $|X_2 - \alpha_2 y| \leqslant C_2 u|y| + D_2\eta$.

Then:

$$|X_1 \oplus X_2 - (\alpha_1 + \alpha_2)y| \leqslant \left(C_1 u + C_2 u + \frac{u}{1+u}(|\alpha_1| + |\alpha_2| + C_1 u + C_2 u) \right) |y| + (1+u)(D_1 + D_2)\eta.$$

Proof. The proof combines FP facts, *e.g.* underflowing additions are correct and Theorem 3, and the hypotheses.

$$
\begin{aligned}
&| X_1 \oplus X_2 - (\alpha_1+\alpha_2)y | \\
&\quad \leqslant |X_1 \oplus X_2 - (X_1+X_2)| + |X_1 + X_2 - (\alpha_1+\alpha_2)y| \\
&\quad \leqslant \frac{u}{1+u} \times |X_1+X_2| + (C_1u + C_2u)|y| + (D_1+D_2)\eta \\
&\quad \leqslant \frac{u}{1+u} \times ((C_1u + |\alpha_1|)|y| + (C_2u+|\alpha_2|)|y|) \\
&\qquad + (C_1u+C_2u)|y| + (D_1+D_2)\left(1+\frac{u}{1+u}\right)\eta \\
&\quad = \left((C_1+C_2)u + \frac{u}{1+u}\times(|\alpha_1|+|\alpha_2|+(C_1+C_2)u)\right)|y| + \eta\frac{1+2u}{1+u}(D_1+D_2) \\
&\quad \leqslant \left((C_1+C_2)u + \frac{u}{1+u}\times(|\alpha_1|+|\alpha_2|+(C_1+C_2)u)\right)|y| + \eta(1+u)(D_1+D_2).
\end{aligned}
$$

□

We now focus on computing $X_1 \oplus (X_2 \otimes y)$, with a possible underflow in the multiplication.

Lemma 7. *Let y an FP number. Let $C_1, C_2, D_1, M, P_2 \in \mathbb{R}_+$. Let $\alpha_1 \in \mathbb{R}, \alpha_2 \in \mathbb{R}^*$. Let X_1 and X_2 be FP numbers such that:*

- $|X_1 - \alpha_1 y| \leqslant C_1 u|y| + D_1\eta$;
- $|y] \geqslant M \Rightarrow |X_1 - \alpha_1 y| \leqslant C_1 u|y|$;
- $|X_2 - \alpha_2| \leqslant C_2 u$;
- $\left|\frac{X_2-\alpha_2}{\alpha_2}\right| \leqslant P_2 < 1$;
- $M \geqslant \frac{\xi}{|\alpha_2|(1-P_2)}$.

Then, $|y| \geqslant M \Rightarrow$

$$
\begin{aligned}
&|X_1 \oplus (X_2 \otimes y) - (\alpha_1+\alpha_2)y| \\
&\qquad \leqslant |y|\left(C_1u + C_2u + \frac{u}{1+u}(|\alpha_1| + 2|\alpha_2| + C_1u + 2C_2u) + \left(\frac{u}{1+u}\right)^2(C_2u+|\alpha_2|)\right)
\end{aligned}
$$

and

$$
\begin{aligned}
&|X_1 \oplus (X_2 \otimes y) - (\alpha_1+\alpha_2)y| \\
&\qquad \leqslant |y|\left(C_1u + C_2u + \frac{u}{1+u}(|\alpha_1| + 2|\alpha_2| + C_1u + 2C_2u) + \left(\frac{u}{1+u}\right)^2(C_2u+|\alpha_2|)\right) + \eta(1+u)\left(\frac{1}{2}+D_1\right).
\end{aligned}
$$

Proof. • Assume $|y| < M$.

The proof relies on an application of Lemma 6 with X_1, α_1 and α_2 being the same. But the X_2 of Lemma 6 is then $X_2 \otimes y$. We have the expected assumptions on X_1, α_1, C_1 and D_1. As for C_2 and D_2, we need to bound the error on $X_2 \otimes y$:

$$
\begin{aligned}
|X_2 \otimes y - \alpha_2 y| &\leqslant |X_2 \otimes y - X_2 \times y| + |y|\,|X_2 - \alpha_2| \\
&\leqslant \frac{u}{1+u}\,|X_2|\,|y| + \frac{\eta}{2} + |y|\,C_2u \\
&\leqslant \frac{u}{1+u}\,(C_2u + |\alpha_2|)\,|y| + |y|\,C_2u + \frac{\eta}{2} \\
&= |y|\left(C_2u + \frac{u}{1+u}(C_2u+|\alpha_2|)\right) + \frac{\eta}{2}.
\end{aligned}
$$

So we can take for C_2 the value $\left(C_2 + \frac{1}{1+u}(C_2u+|\alpha_2|)\right)$, for D_2 the value $\frac{1}{2}$ and have all the required hypotheses. Applying Lemma 6, we get

$$|X_1 \oplus X_2 \otimes y - (\alpha_1+\alpha_2)y| \leqslant \big(C_1u + C_2u + \frac{u}{1+u}(C_2u+|\alpha_2|)+$$
$$\frac{u}{1+u}(|\alpha_1|+|\alpha_2|+C_1u+C_2u+\frac{u}{1+u}(C_2u+|\alpha_2|))\big)|y| + \eta(1+u)\left(\frac{1}{2}+D_1\right)$$
$$= \left(C_1u + C_2u + \frac{u}{1+u}(|\alpha_1|+2|\alpha_2|+C_1u+2C_2u) + \left(\frac{u}{1+u}\right)^2(C_2u+|\alpha_2|)\right)|y|$$
$$+ \eta(1+u)\left(\frac{1}{2}+D_1\right).$$

- Assume $|y| \geqslant M$.

  We know that $|y| \geqslant \frac{\xi}{|\alpha_2|(1-P_2)}$. Moreover, $\left|\frac{X_2-\alpha_2}{\alpha_2}\right| \leqslant P_2$ *i.e.* $|X_2-\alpha_2| \leqslant P_2|\alpha_2|$ and thus:

  $$|\alpha_2| - |X_2| \leqslant P_2|\alpha_2| \text{ } i.e \text{ } |X_2| \geqslant (1-P_2)|\alpha_2|$$

  and as $P_2 < 1$: $|X_2y| \geqslant (1-P_2)|\alpha_2|\frac{\xi}{|\alpha_2|(1-P_2)} = \xi$

  So $|X_2 \otimes y - X_2y| \leqslant \frac{u}{1+u}|X_2||y|$ because the multiplication of X_2 and y does not underflow.

  Then, as in the other case, we apply Lemma 6 with X_1, α_1 and α_2 being the same, X_2 being $X_2 \otimes y$. We have the right assumption on X_1, α_1 and C_1. As for C_2, we need to bound the error on $X_2 \otimes y$:

  $$\begin{aligned}|X_2 \otimes y - \alpha_2 y| &\leqslant |X_2 \otimes y - X_2 \times y| + |y|\,|X_2-\alpha_2| \\ &\leqslant \frac{u}{1+u}|X_2|\,|y| + |y|\,C_2u \\ &\leqslant \frac{u}{1+u}(C_2u+|\alpha_2|)\,|y| + |y|\,C_2u \\ &= |y|\left(C_2u + \frac{u}{1+u}(C_2u+|\alpha_2|)\right).\end{aligned}$$

  So we can take for C_2 the same value as before $C_2 + \frac{1}{1+u}(C_2u+|\alpha_2|)$ and take $D_1 = D_2 = 0$, so that we have all the required hypotheses. Applying Lemma 6, we get

  $$\begin{aligned}&|X_1 \oplus X_2 \otimes y - (\alpha_1+\alpha_2)y| \\ &\leqslant \big(C_1u + C_2u + \frac{u}{1+u}(C_2u+|\alpha_2|) + u(|\alpha_1|+|\alpha_2|+C_1u+C_2u+\frac{u}{1+u}(C_2u+|\alpha_2|))\big)|y| \\ &= (C_1+C_2)u + \frac{u}{1+u}(|\alpha_1|+2|\alpha_2|+(C_1+2C_2)u) + \left(\frac{u}{1+u}\right)^2(C_2u+|\alpha_2|))|y|.\end{aligned}$$

□

The hypotheses of this lemma may seem redundant. There is indeed a bound P_2 on the relative round-off error of α_2 and a bound C_2 on its absolute error. Of course, one could deduce one from the other. But in our application, α_2 will not be a known constant but will live in an interval, and we need an error bound whatever the value in this interval. The P_2 and C_2 will be obtained automatically depending on the stability interval for $h\lambda$. For instance, α_2 may be $\frac{h^2\lambda^2}{6}$ and X_2 may be $\circ\left[h^2\frac{1}{6}\widetilde{\lambda}^2\right]$. As $-3 \leqslant h\lambda \leqslant 0$, we only know that $0 \leqslant \alpha_2 \leqslant 1.5$ and we bound the round-off errors whatever the effective values of h and λ. If we had the value of α_2, we would have better error bounds and no need for both errors. But as we only have an interval, in order to prevent a correlation on α_2 and to have better bounds, we have to consider both bounds.

These results are precisely tailored for their use in Section 6.2 for bounding the local round-off errors of the numerical methods taking possible underflow into account.

We note that if no underflow occurs (which corresponds here to $|y|$ greater than or equal to the threshold M), we get a result similar to the one of [7]. Nonetheless, the bound provided by Lemma 7 is a bit tighter, as we use the bound $\frac{u}{1+u}$ provided by Jeannerod and Rump [26]. Unfortunately, this tighter bound is not enough to get rid of u^2 (and higher) terms, contrary to the seminal work [26] where error bounds were simplified. This is slightly disappointing but this improves the error bounds anyway. If an underflow may have occurred, a term depending on η is added.

6.2 Bound on local round-off errors of classic methods

In this section, we use the general results of Section 6.1 to bound relative local errors of classic Runge-Kutta methods. To perform it, we have to bound each term of the $R(h,\lambda)$ polynomial (such as $\frac{h\lambda}{6}$, $\frac{h^2\lambda^2}{2}$, $\frac{h^3\lambda^3}{12}$ and their FP errors). For this purpose, we use Gappa, a tool to formally verify properties on numerical

programs using FP arithmetic [12, 13].[2] It bounds FP errors and values, and produces a proof term which could be checked by proof assistants such as Coq. In this section, we assume that computations are done in the *binary64* IEEE-754 format. To refine our results, we focus on stable methods and use their region of stability for $h\lambda$. First, we study the Runge-Kutta method of order 1, well-known as Euler's method.

Lemma 8. Bound on Euler local round-off errors. *Let us assume* $-2 \leqslant h\lambda \leqslant -2^{-100}$ *and* $2^{-60} \leqslant h \leqslant 1$. *Let* $n \in \mathbb{N}$. *Then:* $|\widetilde{y_n}| \geqslant \frac{\xi}{2(1-2.01u)} \Rightarrow \varepsilon_{n+1} \leqslant 9.01u\,|\widetilde{y_n}|$ *and* $\varepsilon_{n+1} \leqslant 9.01u\,|\widetilde{y_n}| + (0.5+u)\eta$.

Proof. This proof relies on an application of Lemma 7 with $y = \widetilde{y_n}$. Then $X_1 = \widetilde{y_n}$, $\alpha_1 = 1$, $C_1 = 0$ and $D_1 = 0$. And $X_2 = h \otimes \widetilde{\lambda}$, $\alpha_2 = h\lambda$. We then need a correct C_2. Using Gappa, we obtain $|X_2 - h\lambda| \leqslant 4 \times 2^{-53}$ from intervals on $h\lambda$ and on h, so we choose $C_2 = 4$. Moreover, Gappa provides a bound on the relative error $\left|\frac{X_2 - h\lambda}{h\lambda}\right| \leqslant 2.01 \times 2^{-53}$. So we choose $P_2 = 2.01u$. The application of Lemma 7 then gives that $|\widetilde{y_n}| \geqslant \frac{\xi}{2(1-2.01u)}$ implies

$$\varepsilon_{n+1} \leqslant |\widetilde{y_n}| \left(4u + u\,(1 + 2|h\lambda| + 8u) + u^2(4u + |h\lambda|)\right).$$

Note that $\frac{\xi}{2(1-2.01u)}$ has been chosen in order to have the conditions of Lemma 7 on M. Moreover, Lemma 7 gives:

$$\varepsilon_{n+1} \leqslant |\widetilde{y_n}|\,(4u + u\,(1 + 2|h\lambda| + 8u) + u^2(4u + |h\lambda|)) + (0.5+u)\eta.$$

As $|h\lambda| \leq 2$, we have

$$\begin{aligned}\varepsilon_{n+1} &\leq |\widetilde{y_n}| \left(9u + 14u^2 + 6u^3\right) + (0.5+u)\eta \\ &\leqslant 9.01u\,|\widetilde{y_n}| + (0.5+u)\eta.\end{aligned}$$

and $|\widetilde{y_n}| \geqslant \frac{\xi}{2(1-2.01u)} \Rightarrow$

$$\varepsilon_{n+1} \leq |\widetilde{y_n}| \left(9u + 14u^2 + 6u^3\right) \leqslant 9.01u\,|\widetilde{y_n}|.$$

□

Lemma 9. Bound on Runge-Kutta 2 local round-off errors. *Let us assume stability, that* $-2 \leqslant h\lambda \leqslant -2^{-100}$ *and* $2^{-60} \leqslant h \leqslant 1$. *Then for all* n, $|\widetilde{y_n}| \geqslant \frac{\xi}{2(1-8u)} \Rightarrow \varepsilon_{n+1} \leqslant 27.01u\,|\widetilde{y_n}|$ *and* $\varepsilon_{n+1} \leqslant 27.01u\,|\widetilde{y_n}| + 1.01\eta$.

Proof. As explained, we have for the RK2 method

$$\widetilde{y_{n+1}} = \circ\left[\widetilde{y_n} + h\widetilde{\lambda}\widetilde{y_n} + hh\tfrac{1}{2}\widetilde{\lambda}\widetilde{\lambda}\widetilde{y_n}\right].$$

Stability means that $|1 + h\lambda + \frac{h^2\lambda^2}{2}| \leqslant 1$. As in Euler's method, we rely on Lemma 7. We put in Table 2 the corresponding α, C, P_2 and D. To determine the threshold M at which there is no underflow in a step of computation, we determine an underflow's threshold M_{loc} for each term of the corresponding computation. Then, we just need to choose M as the maximum of all the M_{loc} values.

The first 3 lines are similar to the local error of Euler's method above (as the hypotheses on $h\lambda$ are the same).

Then Gappa gives us the bound of Line 4

$$\left|\circ\left[hh\frac{1}{2}\widetilde{\lambda}\widetilde{\lambda}\right] - \frac{h^2\lambda^2}{2}\right| \leqslant 13u$$

under the given hypotheses on $h\lambda$ and h. It also gives the bound on the relative error:

$$\left|\frac{\circ\left[hh\frac{1}{2}\widetilde{\lambda}\widetilde{\lambda}\right] - \frac{h^2\lambda^2}{2}}{\frac{h^2\lambda^2}{2}}\right| \leqslant 2^{-50} = 8u$$

Many other lines have been removed for the sake of brevity. The last line corresponds to using Lemma 7 with the C and α of Lines 3 and 4, while bounding the u^2 and u^3 terms.
Here, M is the maximum of all the M_{loc} values, *i.e.*, $\frac{\xi}{2(1-8u)}$. □

Lemma 10. Bound on Runge-Kutta 4 local round-off errors *Let us assume stability, that* $-3 \leqslant h\lambda \leqslant -2^{-100}$ *and* $2^{-60} \leqslant h \leqslant 1$. *Then for all* n, $|\widetilde{y_n}| \geqslant \frac{\xi}{0.5(1-4u)} \Rightarrow \varepsilon_{n+1} \leqslant 164u\,|\widetilde{y_n}|$ *and* $\varepsilon_{n+1} \leqslant 164u\,|\widetilde{y_n}| + 5.6\eta$.

[2] The corresponding Gappa scripts are available online at the following address: `https://www.lri.fr/~faissole/Gappa_RK/`

Table 2: Steps for the local round-off error bound of RK2.

FPterm	C	P_2	M_{loc}	D
$\widetilde{y_n}$	0	–	0	0
$h \otimes \widetilde{\lambda}$	4	$2.01u$	–	–
$\circ \left[\widetilde{y_n} + h\widetilde{\lambda}\widetilde{y_n}\right]$	$9 + 15u$	–	$\frac{\xi}{2(1-2.01u)}$	$0.5 + u$
$\circ \left[hh\frac{1}{2}\widetilde{\lambda}\widetilde{\lambda}\right]$	13	$8u$	–	–
$\circ [\widetilde{y_n} + \dots]$	27.01	–	$\frac{\xi}{2(1-8u)}$	1.01

Proof. The RK4 method is defined by Equation (9). In this case, stability means:

$$\left|1 + h\lambda + \frac{h^2\lambda^2}{2} + \frac{h^3\lambda^3}{12} + \frac{h^4\lambda^4}{24}\right| \leqslant 1$$

which leads to $-3 \leqslant h\lambda \leqslant 0$. As previously, we rely on Lemma 7 and on Gappa under the given hypotheses on $h\lambda$ and h. We put in Table 3 the corresponding α, C, P_2, D and M_{loc} of some FP terms. Here, we can take $M = \frac{\xi}{0.5(1-4u)}$, which is the maximum of the M_{loc} values. Here, contrarily to Lemmas 8 and 9, M is not the M_{loc} value corresponding to the last term. □

Table 3: Steps for the local round-off error bound of RK4.

FPterm	C	P_2	M_{loc}	D
$\widetilde{y_n}$	0	–	0	0
$h \otimes \frac{1}{6} \otimes \widetilde{\lambda}$	2	$4u$	–	–
$\circ \left[\widetilde{y_n} + h\frac{1}{6}\widetilde{\lambda}\right]$	4	–	$\frac{\xi}{0.5(1-4u)}$	$0.5 + u$
$h \otimes \frac{1}{3} \otimes \widetilde{\lambda}$	4	$4u$	–	–
$\circ \left[h^2\frac{1}{6}\widetilde{\lambda}^2\right]$	9	$8u$	–	–
$\circ \left[h^3\frac{1}{12}\widetilde{\lambda}^3\right]$	21	$16u$	–	–
$\circ \left[h^4\frac{1}{24}\widetilde{\lambda}^4\right]$	40	$16u$	–	–
$\circ [\widetilde{y_n} + \dots]$	164	–	$\frac{\xi}{3.375(1-16u)}$	5.6

7 Global errors

In Section 6, we have exhibited bounds on local errors for classic methods. Following our methodology (see Section 3), we now want to bound the global round-off error of these methods, *i.e.*, the total round-off error after a given number of iterations. We first prove, in Section 7.1, that we can find bounds on global errors from bounds on all the previous local errors. Then, we provide bounds on global round-off errors of classic Runge-Kutta methods in Section 7.2.

7.1 From local to global round-off errors

This section is devoted to the dependency between local and global round-off errors. More particularly, if we take underflow into account to bound local errors, we have to take it into account when bounding the global error. We first prove a technical lemma to relate local and global errors.

Lemma 11. *Let $C \geqslant 0$. Suppose that $0 < Cu + |R(h,\lambda)|$. Then, for all $n \in \mathbb{N}$:*
$(Cu + |R(h,\lambda)|)^{n+1} \left(\varepsilon_0 + n\frac{Cu|y_0|}{Cu+|R(h,\lambda)|}\right) + Cu|R(h,\lambda)|^n|y_0|$

$$\leq (Cu + |R(h,\lambda)|)^{n+1} \left(\varepsilon_0 + (n+1)\frac{Cu|y_0|}{Cu+|R(h,\lambda)|}\right)$$

Proof. By simplifying the $(Cu + |R(h,\lambda)|)^{n+1}\varepsilon_0$ value, it remains to prove that

$$(Cu + |R(h,\lambda)|)^{n+1} n\frac{Cu|y_0|}{Cu + |R(h,\lambda)|} + Cu\,|R(h,\lambda)|^n\,|y_0| \leqslant (Cu + |R(h,\lambda)|)^{n+1}(n+1)\frac{Cu|y_0|}{Cu + |R(h,\lambda)|}.$$

By simplifying again by $(Cu + |R(h,\lambda)|)^{n+1} n\frac{Cu|y_0|}{Cu+|R(h,\lambda)|}$, it remains to prove that

$$Cu\,|R(h,\lambda)|^n\,|y_0| \leqslant (Cu + |R(h,\lambda)|)^{n+1}\frac{Cu|y_0|}{Cu + |R(h,\lambda)|}.$$

This is true if $y_0 = 0$ or $C = 0$. Otherwise, we have to prove

$$|R(h,\lambda)|^n \leqslant (Cu + |R(h,\lambda)|)^{n+1}\frac{1}{Cu + |R(h,\lambda)|} = (Cu + |R(h,\lambda)|)^n.$$

As $C \geqslant 0$, this inequality holds and so the result. □

The following theorem provides the bound on the global round-off error of a method from all the previous local errors. In Section 6, the local error of the method at iteration n depends on whether an underflow occurs. Actually, stability of our methods implies that there is no underflow contribution in the global error until there is an underflow contribution in the last local error.

Theorem 12. From local to global round-off error. *Let $C \geqslant 0, D \geqslant 0, M > 0$. Suppose that:*

- $\forall n \in \mathbb{N}^*, |\widetilde{y_n}| \geqslant M \Rightarrow \varepsilon_n \leqslant Cu|\widetilde{y_{n-1}}|$;
- $\forall n \in \mathbb{N}^*, \varepsilon_n \leqslant Cu|\widetilde{y_{n-1}}| + D\eta$;
- $0 < Cu + |R(h,\lambda)| < 1$.

Then:

- $\forall n,\ |\widetilde{y_n}| \geqslant M \Rightarrow |E_n| \leqslant (Cu + |R(h,\lambda)|)^n \left(\varepsilon_0 + n\frac{Cu|y_0|}{Cu+|R(h,\lambda)|}\right)$.
- $\forall n, |E_n| \leqslant (Cu + |R(h,\lambda)|)^n \left(\varepsilon_0 + n\frac{Cu|y_0|}{Cu+|R(h,\lambda)|}\right) + nD\eta$.

Proof. We do an induction on n. For $n = 0$, we have:

$$|E_0| = \varepsilon_0 = (Cu + |R(h,\lambda)|)^0 \left(\varepsilon_0 + 0\frac{Cu|y_0|}{Cu + |R(h,\lambda)|}\right) + 0D\eta$$

and both results hold.

Suppose the result is true for a given $n \in \mathbb{N}$.

- Assume $|\widetilde{y_{n+1}}| \geqslant M$.

  By Lemma 1, we have $|\widetilde{y_n}| \geqslant M$. Let us try to bound $|E_{n+1}|$ by a simple unfolding of the definitions given in Equations (11) and (10) and a triangular inequality:

  $$|E_{n+1}| \leqslant |\widetilde{y_{n+1}} - R(h,\lambda)\widetilde{y_n}| + |R(h,\lambda)\widetilde{y_n} - R(h,\lambda)y_n| = \varepsilon_{n+1} + |R(h,\lambda)|\,|E_n|.$$

  Let us use the hypothesis on ε_{n+1} to bound it:

  $$\varepsilon_{n+1} \leqslant Cu|\widetilde{y_n}| \leqslant Cu\,(|\widetilde{y_n} - y_n| + |y_n|) = Cu|E_n| + Cu\,|R(h,\lambda)|^n\,|y_0|. \tag{12}$$

  Then, using Equation (12):

  $|E_{n+1}| \leqslant Cu|E_n| + Cu|R(h,\lambda)|^n|y_0| + |R(h,\lambda)||E_n| = (Cu + |R(h,\lambda)|)|E_n| + Cu|R(h,\lambda)|^n|y_0|.$

  Using the induction hypothesis, we now have

  $$\begin{aligned}|E_{n+1}| &\leqslant (Cu + |R(h,\lambda)|)(Cu + |R(h,\lambda)|)^n \times \left(\varepsilon_0 + n\,\frac{Cu|y_0|}{Cu + |R(h,\lambda)|}\right) + Cu\,|R(h,\lambda)|^n\,|y_0| \\ &= (Cu + |R(h,\lambda)|)^{n+1}\left(\varepsilon_0 + n\frac{Cu|y_0|}{Cu + |R(h,\lambda)|}\right) + Cu\,|R(h,\lambda)|^n\,|y_0|.\end{aligned}$$

  Applying Lemma 11, we get:

  $|E_{n+1}| \leqslant$

  $$(Cu + |R(h,\lambda)|)^{n+1}\left(\varepsilon_0 + (n+1)\frac{Cu|y_0|}{Cu+|R(h,\lambda)|}\right).$$

- Assume $|\widetilde{y_{n+1}}| < M$.

  Bounding ε_{n+1} from the assumptions, we have:

  $$\begin{aligned}|E_{n+1}| &\leqslant \varepsilon_{n+1} + |R(h,\lambda)||E_n| \leqslant Cu|\widetilde{y_n}| + D\eta + |R(h,\lambda)||E_n| \\ &\leqslant Cu\left(|\widetilde{y_n} - y_n| + |y_n|\right) + D\eta + |R(h,\lambda)||E_n| \\ &= Cu|E_n| + Cu\,|R(h,\lambda)|^n|y_0| + D\eta + |R(h,\lambda)||E_n| \\ &= (Cu + |R(h,\lambda)|)|E_n| + Cu|R(h,\lambda)|^n|y_0| + D\eta.\end{aligned}$$

  Using the induction hypothesis, we now have

  $$\begin{aligned}|E_{n+1}| &\leqslant (Cu + |R(h,\lambda)|) \\ &\quad \times \left((Cu + |R(h,\lambda)|)^n \left(\varepsilon_0 + n\tfrac{Cu|y_0|}{Cu+|R(h,\lambda)|}\right) + nD\eta\right) + Cu|R(h,\lambda)|^n|y_0| + D\eta \\ &= (Cu + |R(h,\lambda)|)^{n+1} \left(\varepsilon_0 + n\tfrac{Cu|y_0|}{Cu+|R(h,\lambda)|}\right) + Cu|R(h,\lambda)|^n\,|y_0| + (Cu + |R(h,\lambda)|)nD\eta + D\eta. \\ &= (Cu + |R(h,\lambda)|)^{n+1} \left(\varepsilon_0 + n\tfrac{Cu|y_0|}{Cu+|R(h,\lambda)|}\right) + Cu|R(h,\lambda)|^n\,|y_0| + (n+1)D\eta.\end{aligned}$$

  Applying Lemma 11, we get:

  $$|E_{n+1}| \leqslant (Cu + |R(h,\lambda)|)^{n+1} \left(\varepsilon_0 + (n+1)\tfrac{Cu|y_0|}{Cu+|R(h,\lambda)|}\right) + (n+1)D\eta.$$

□

This theorem bounds the global round-off error in a general way, even if underflow occurs. Applying Theorem 12 on stable methods gives an interesting property: even if local round-off errors can accumulate through the iterations, the global error does not diverge too fast. The first errors are attenuated by the computations as they are multiplied by a power of $|R(h,\lambda)| < 1$. It ensures a reasonable final error bound and gives a link between stability in the sense of numerical analysis and stability in the sense of FP arithmetic.

As in Section 6, we get a slightly better result than in [7] (due to the bounds from [26]) as long as no underflow occurs. In case underflow may have occurred, we get the same result to which is added a term proportional to η.

Note also that from Theorem 12 we can bound the relative round-off error when no underflow occurs:

$$\left|\tfrac{\widetilde{y_n}-y_n}{y_n}\right| \leqslant \left(\tfrac{Cu+|R(h,\lambda)|}{|R(h,\lambda)|}\right)^n \left(\left|\tfrac{\widetilde{y_0}-y_0}{y_0}\right| + n\tfrac{Cu}{Cu+|R(h,\lambda)|}\right).$$

If we assume that $Cu \ll |R(h,\lambda)|$, this reduces to

$$\left|\frac{\widetilde{y_n} - y_n}{y_n}\right| \lesssim \left|\frac{\widetilde{y_0} - y_0}{y_0}\right| + n\frac{Cu}{|R(h,\lambda)|}. \tag{13}$$

As seen in Section 6, C is actually never above a small constant (at most 200 for the Runge-Kutta methods we consider). Thus, assuming that $Cu \ll |R(h,\lambda)|$, this relative bound is proportional to n, as is the number of FP operations involved in the computation of $\widetilde{y_n}$.

7.2 Bound on global error of Runge-Kutta methods

In Section 6, we have exhibited bounds on the local errors of classic Runge-Kutta methods. In Section 7.1, we have shown how to bound the global error of a scheme from all its local round-off errors. Hence, computing bounds on global round-off errors of classic methods is a generic process. In this section, we give bounds on global errors for Euler, Runge-Kutta 2 and Runge-Kutta 4 methods. As in Section 6, we assume that computations are done in *binary64*.

First, we can give a bound on the global round-off error of the Euler's method instantiating Theorem 12 with $C = 9.01$, $D = 0.5 + u$ and $M = \frac{\xi}{2(1-2.01u)}$.

Theorem 13. Bound on absolute error of Euler scheme. *Let us assume* $-2 \leqslant h\lambda \leqslant -2^{-100}$ *and* $2^{-60} \leqslant h \leqslant 1$. *Then* $\forall n$,

$|E_n| \leqslant (9.01u + |R(h,\lambda)|)^n \left(\varepsilon_0 + n\tfrac{9.01u|y_0|}{9.01u+|R(h,\lambda)|}\right) + n(0.5+u)\eta$

and

$|\widetilde{y_n}| \geqslant \tfrac{\xi}{2(1-2.01u)} \Rightarrow |E_n| \leqslant (9.01u + |R(h,\lambda)|)^n \left(\varepsilon_0 + n\tfrac{9.01u|y_0|}{9.01u+|R(h,\lambda)|}\right).$

The reasoning is the same for higher-order methods as for Euler's method. For Runge-Kutta 2 methods, we instantiate Theorem 12 with the constants provided by Lemma 9.

Theorem 14. Bound on absolute error of RK2 method. *Let us assume* $-2 \leqslant h\lambda \leqslant -2^{-100}$ *and* $2^{-60} \leqslant h \leqslant 1$. *Then* $\forall n$,

$|E_n| \leqslant (27.01u + |R(h,\lambda)|)^n \left(\varepsilon_0 + n\frac{27.01u|y_0|}{27.01u+|R(h,\lambda)|}\right) + 1.01n\eta$

and

$|\widetilde{y_n}| \geqslant \frac{\xi}{2(1-8u)} \Rightarrow |E_n| \leqslant (27.01u + |R(h,\lambda)|)^n \left(\varepsilon_0 + n\frac{27.01u|y_0|}{27.01u+|R(h,\lambda)|}\right)$.

At last, we provide a bound on the global round-off error of the Runge-Kutta 4 method using the bounds on local errors of Lemma 10.

Theorem 15. Bound on absolute error of RK4 method. *Let us assume* $-3 \leqslant h\lambda \leqslant -2^{-100}$ *and* $2^{-60} \leqslant h \leqslant 1$. *Then* $\forall n$,

$|E_n| \leqslant (164u + |R(h,\lambda)|)^n \left(\varepsilon_0 + n\frac{164u|y_0|}{164u+|R(h,\lambda)|}\right) + 5.6n\eta$

and

$|\widetilde{y_n}| \geqslant \frac{\xi}{0.5(1-4u)} \Rightarrow |E_n| \leqslant (164u + |R(h,\lambda)|)^n \left(\varepsilon_0 + n\frac{164u|y_0|}{164u+|R(h,\lambda)|}\right)$.

8 Related work

We review here part of the previous works on the numerical accuracy of one-step numerical integration methods.

Defining new numerical integration methods which produce small method errors while preserving efficiency has been an important research direction. But in particular cases, for long-time integration as for Hamiltonian systems [34, 15, 37, 19, 20, 4] (see below) which preserved total energy properties, round-off errors shall not be neglected. Mostly, compensated summation techniques [31, Chap. 5] are used to increase the accuracy of FP computation while preserving the overall efficiency of integration methods. In particular, it is applied in [34, 19, 20, 4]. Note that for implicit Runge-Kutta methods, another source of inaccuracy is the FP representation of the coefficients of the Butcher tableau which usually breaks mathematical properties such as energy preservation. A study with an interval analysis approach of this phenomenon is provided in [1]. Note that only [4] provides an estimation of the propagation of round-off errors. FP errors are estimated using a second implementation of the integration scheme with a lower precision FP arithmetic. The difference between the full precision integration scheme and the lower precision one is then used as a rough estimation of the round-off error propagation. In [15, 37], FP errors are avoided by recasting the integration methods into integer arithmetic by scaling all values. [37] extends [15] by considering that the increment part of integration, *i.e.*, $h \otimes f$, is not exactly computed and proposes a problem-dependent rounding function to ensure the energy preservation properties of the numerical scheme. No analysis of the propagation of FP errors is given but a method to measure the ratio between round-off error and truncation error is given to understand what kind of error is preponderant during the numerical integration.

[22, 39, 16, 38, 27] are among the few existing work on the propagation of round-off errors in numerical integration schemes. In [22], a particular two-dimensional linear IVP-ODE is considered and it is solved with Heun's method (a variant of RK2 method). Note that IEEE754 standard is not considered in this limited study on one example. In [16], the propagation of round-off errors on the initial values are considered for two-dimensional non-linear IVP-ODE. An explicit formula of the propagation of all errors (method and round-off errors) is presented involving partial derivative of f, the local truncation error, and an estimation of the round-off errors induce by the evaluation of f. But some assumptions are made for simplification, *i.e.*, only linear terms of the Taylor expansion of the error expression are considered. In [39] a general method is presented to study the propagation of round-off errors and method errors for multi-dimensional linear IVP-ODE. This method is based on sensitivity analysis of the problems with respect to the initial conditions to estimate the propagation of the errors. In [38] a framework to analyze the propagation of round-off errors based on Lipschitz constant of non-linear of IVP-ODE is considered. A special case of FP errors analysis is in [27] which proposes a variable step-size integration method based on Euler's formula with a goal to reach optimal accuracy for non-linear IVP-ODE. A theorem in [27] states that "The value of the total error is minimum when the global truncation error coincides with the rounding error" so a step-size h is looked for such as the round-off error has the same magnitude than the truncation error. All these approaches produce a rough bound on the FP errors in regards to our fine-grained analysis mainly because they are based on a more general setting and exceptional behaviors are not considered.

In *validated numerical integration* methods, interval arithmetic is used in order to bound round-off errors and method errors. Despite these methods produce rigorous results, in case of numerical integration methods based on Taylor series [5, 32] or based on Runge-Kutta methods [10, 2], the distinction between round-off errors and method errors is not considered, and so no information on the propagation of round-off errors is provided nor considered. Note that an implementation of the approach of [10] has been formally

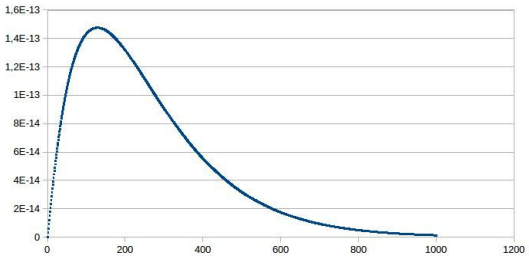

Figure 1: RK2 example: round-off error bound.

proved in [25] but without giving much attention on the propagation on round-off errors as they rely on multi-precision. A special case is the work [35] in which a more precise study on the propagation of round-off errors in the numerical integration method is considered in order to reduce the pessimism of interval computations. Mainly, in [35] the implementation mixes FP computations and interval computations. Error free transformations [31] are used to compute round-off errors of the FP part which are then accumulated and added in a high order interval term associated to the method error. A probabilistic propagation of round-off errors is considered in [3] which uses stochastic arithmetic to assert the accuracy of the results.

In *computer arithmetic community*, sharper round-off analysis are usually performed as in [18, 17] which study the implementation of quadrature formula with multi-precision arithmetic in order to guarantee the accuracy of the computed result. In [18], Newton-Cotes quadrature is considered and in [17], Gauss-Legendre quadrature method is studied. In both articles a comprehensive analysis to bound the FP errors is given but without considering exceptional behaviors. They consider approximation on the evaluation of the function to integrate and the coefficients of the quadrature methods in order to produce a p-bit correct approximation of the integral. Quadrature formula share important properties with numerical methods to solve IVP-ODE. The main difference is that quadrature formula usually assume the expression of functions to integrate is known while in numerical methods for IVP-ODE, some approximations of the function to integrate have to be introduced. Our work can be considered as an extension of these previous works by considering more elaborate algorithms.

9 Conclusion and future work

We have studied a large family of numerical integration methods that are commonly used to solve ordinary differential equations. We have done a fine-grained FP analysis, providing a new and tight error bound on the computed values, that includes exceptional behavior. More precisely, we have given a sufficient condition to prevent overflow during the whole computation and we have given round-off error bounds even in presence of underflow. Moreover, under an easy-to-check assumption, we provide an error bound that is as with an unbounded exponent range.

Let us now describe in details an example. The ODE is $\dot{y} = -\frac{y}{2}$, that is to say $\lambda = -0.5$ with $y_0 = 1$. We choose the RK2 method, $h = \frac{1}{64}$ and we compute $n = 1,000$ iterations. To compute the wanted errors, we have compared them with a multi-precision (1,000 bits) computations of the RK2 iterates using MPFR. Figure 1 shows our round-off error bound on this example. It exhibits a bump for small n before a fast decrease. Indeed, the bound is here $(Cu + |R(h,\lambda)|)^n \, n \frac{Cu|y_0|}{Cu+|R(h,\lambda)|}$ with $C \approx 28$. For small n, the dominating term is about $nCu|y_0|$. When n increases, the $(Cu + |R(h,\lambda)|)^n$ term is dominating and the error (near-) exponentially decreases. To compare with reality, Figure 2 shows the real round-off error of this numerical scheme. Both figures exhibit the same behavior with a bump and a fast decrease. Our error bounds are of course larger than the real errors, but it is reassuring that the curves look alike. As far as the method error comes into play, things are very different. Figure 3 shows on a logarithmic scale both the real method error (compared with the 1000-bit MPFR exp function), the round-off error bound and the real error bound. We have plotted the errors for other values of λ, h and other schemes. When λ gets near zero, the bump is more flat, but the behavior is very similar in all cases. In particular, even for the RK2 method, which is assumed to be reasonably correct, our example shows that the method error is overriding. It also demonstrates that, even if larger than reality, our error bound is quite sufficient to ensure that the errors due to FP computations are negligible in this case.

A natural perspective is to enlarge the class of methods we consider. We tackled constant-step methods, but variable-step methods exist, where the h is a function of n. This is difficult due to the choice of the value h_n, that depends upon the dynamics of the system. Another choice was to only consider unidimensional problems, but we are also interested in multi-dimensional problems: the scalar λ is then a matrix and

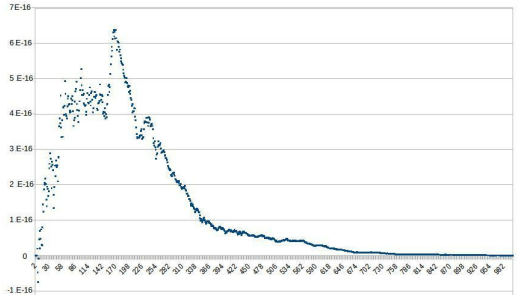

Figure 2: RK2 example: signed round-off error.

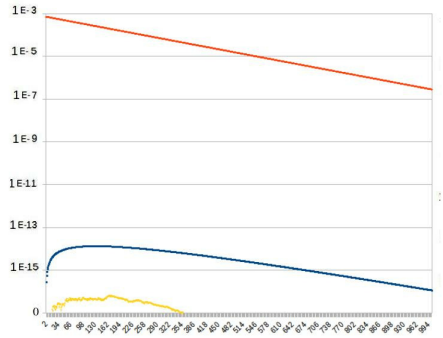

Figure 3: RK2 example: method error (red line), round-off error bound (blue line) and signed round-off error (yellow line) on a logarithmic scale.

matrix-vector multiplications are done at each step. The corresponding stability conditions would expectedly depend on the eigenvalues of the matrix. As we tackled only one-step methods, we may also want to extend our round-off error analysis to other numerical integration methods, *e.g.*, the multi-step methods belonging into Adams-Bashworth family. A last family to be studied is the implicit numerical integration methods family; they usually require the solution of fixed-point equations and so they involve more complex numerical algorithms which may produce more round-off errors.

A related perspective is to enlarge the class of the systems we consider by allowing a more complex dynamics: we want to consider non-linear ODEs. As for linear stability, some classes of non-linear problems have been studied from a stability perspectives such that contracting non-linear systems. The presented round-off error analysis in this article could be extended to deal with such non-linear problems.

The pen-and-paper proofs displayed here are quite complex and may be hard to trust. A solution would be to formally prove the theorems presented here in order to increase the trust in them. It would also probably help in finding the values M, C, P_2 and D as the methodology is quite systematic. To choose the proof assistant to be used, we will look into the available libraries, to decrease the proof burden. For instance, Coq has a large library for FP arithmetic [8, 9] and mixes well with Gappa while Isabelle/HOL has a large library about ODEs [24].

Acknowledgments

This research was partially supported by Labex DigiCosme (project ANR-11-LABEX-0045-DIGICOSME) operated by ANR as part of the program “Investissement d’Avenir” Idex Paris-Saclay (ANR-11-IDEX-0003-02) and by FastRelax ANR-14-CE25-0018-01.

References

[1] Julien Alexandre dit Sandretto. Runge-Kutta theory and constraint programming. *Reliable Computing*, 25:178–201, 2017.

[2] Julien Alexandre dit Sandretto and Alexandre Chapoutot. Validated explicit and implicit Runge-Kutta methods. *Reliable Computing*, 22, 2016.

[3] René Alt and Jean Vignes. Validation of results of collocation methods for ODEs with the CADNA library. *Applied Numerical Mathematics*, 21(2):119–139, 1996.

[4] Mikel Antoñana, Joseba Makazaga, and Ander Murua. Reducing and monitoring round-off error propagation for symplectic implicit Runge-Kutta schemes. *Numerical Algorithms*, 76(4):861–880, 2017.

[5] Martin Berz and Kyoko Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4(4):361–369, 1998.

[6] P. Bodenheimer, G.P. Laughlin, M. Rozyczka, T. Plewa, H.W. Yorke, and H.W. Yorke. *Numerical Methods in Astrophysics: An Introduction*. Series in Astronomy and Astrophysics. CRC Press, 2006.

[7] Sylvie Boldo, Florian Faissole, and Alexandre Chapoutot. Round-off Error Analysis of Explicit One-Step Numerical Integration Methods. In *24th IEEE Symposium on Computer Arithmetic*, pages 82–89, London, United Kingdom, July 2017.

[8] Sylvie Boldo and Guillaume Melquiond. Flocq: A unified library for proving floating-point algorithms in Coq. In *Proceedings of the 20th IEEE Symposium on Computer Arithmetic*, pages 243–252, July 2011.

[9] Sylvie Boldo and Guillaume Melquiond. *Computer Arithmetic and Formal Proofs*. ISTE Press - Elsevier, December 2017.

[10] Olivier Bouissou, Alexandre Chapoutot, and Adel Djoudi. Enclosing temporal evolution of dynamical systems using numerical methods. In *NASA Formal Methods*, number 7871 in LNCS, pages 108–123. Springer, 2013.

[11] Dirk Brouwer. On the accumulation of errors in numerical integration. *The Astronomical Journal*, 46:149–153, 1937.

[12] Marc Daumas and Guillaume Melquiond. Certification of bounds on expressions involving rounded operators. *Transactions on Mathematical Software*, 37(1):1–20, 2010.

[13] Florent de Dinechin, Christoph Lauter, and Guillaume Melquiond. Certifying the floating-point implementation of an elementary function using Gappa. *Transactions on Computers*, 60(2):242–253, 2011.

[14] R.C. DiPrima. *Elementary Differential Equations and Boundary Value Problems*. Wiley, 2008.

[15] David J. D. Earn. Symplectic integration without roundoff error. In *Ergodic Concepts in Stellar Dynamics*, volume 430 of *LNP*, pages 122–130. Springer Berlin Heidelberg, 1994.

[16] Erwin Fehlberg. Error propagation in Runge-Kutta type integration formulas. Technical Report NASA-TR-R-352, M-530, NASA Marshall Space Flight Center, 1970.

[17] Laurent Fousse. Accurate multiple-precision Gauss-Legendre quadrature. In *18th IEEE Symposium on Computer Arithmetic*, pages 150–160. IEEE Computer Society, 2007.

[18] Laurent Fousse. Multiple-precision correctly rounded Newton-Cotes quadrature. *ITA*, 41(1):103–121, 2007.

[19] Toshio Fukushima. Reduction of round-off errors in the extrapolation methods and its application to the integration of orbital motion. *Astronomical Journal*, 112(3), 1996.

[20] E. Hairer, R. I. McLachlan, and A. Razakarivony. Achieving Brouwer's law with implicit Runge–Kutta methods. *BIT Numerical Mathematics*, 48(2):231–243, Jun 2008.

[21] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8. Springer-Verlag, 1993.

[22] Harry B. Huskey. On the precision of a certain procedure of numerical integration. *Journale of Research of the National Bureau of Standards*, 42:57–62, 1949.

[23] IEEE standard for floating-point arithmetic. *IEEE Std 754-2008*, Aug 2008.

[24] Fabian Immler. Formally verified computation of enclosures of solutions of ODEs. In *NASA Formal Methods*, pages 113–127. Springer, 2014.

[25] Fabian Immler and Johannes Hölzl. Numerical analysis of ordinary differential equations in Isabelle/HOL. In *Interactive Theorem Proving*, volume 7406 of *LNCS*, pages 377–392. Springer Berlin Heidelberg, 2012.

[26] Claude-Pierre Jeannerod and Siegfried M. Rump. On relative errors of floating-point operations: optimal bounds and applications. *Mathematics of Computation*, pages 803–819, 2016.

[27] Elizabeth A. Kalinina. The most precise computations using Euler's method in standard floating-point arithmetic applied to modelling of biological systems. *Computer Methods and Programs in Biomedicine*, 111(2):471 – 479, 2013.

[28] J. D. Lambert. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley & Sons, Inc., 1991.

[29] S. Lang. *Analyse réelle*. InterEditions, 1977.

[30] Gustavo Migoni, Ernesto Kofman, and Franois Cellier. Quantization-based new integration methods for stiff ordinary differential equations. *SIMULATION*, 88(4):387–407, 2012.

[31] Jean-Michel Muller, Nicolas Brunie, Florent de Dinechin, Claude-Pierre Jeannerod, Mioara Joldes, Vincent Lefèvre, Guillaume Melquiond, Nathalie Revol, and Serge Torres. *Handbook of Floating-Point Arithmetic*. Birkhäuser Basel, 2 edition, 2018.

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

[33] Ken Newman, Stephen Terrence Buckland, Byron Morgan, Ruth King, David Louis Borchers, Diana Cole, Panagiotis Besbeas, Olivier Gimenez, and Len Thomas. *Modelling population dynamics: Model formulation, fitting and assessment using state-space methods*. Methods in Statistical Ecology. Springer, 2014.

[34] Gerald D. Quinlan. Round-off error in long-term orbital integrations using multistep methods. *Celestial Mechanics and Dynamical Astronomy*, 58(4):339–351, 1994.

[35] N. Revol, K. Makino, and M. Berz. Taylor models and floating-point arithmetic: proof that arithmetic operations are validated in COSY. *The Journal of Logic and Algebraic Programming*, 64(1):135 – 154, 2005.

[36] R. D. Richtmyer and K. W. Morton. *Difference methods for initial-value problems*. Interscience Publishers, 1967.

[37] Robert D. Skeel. Symplectic integration with floating-point arithmetic and other approximations. *Applied Numerical Mathematics*, 29(1):3 – 18, 1999.

[38] Marc N. Spijker. Error propagation in Runge-Kutta methods. *Applied Numerical Mathematics*, 22:309–325, 1996.

[39] Theodore E. Sterne. The accuracy of numerical solutions of ordinary differential equations. *Mathematical Tables and Other Aids to Computation*, 7(43):159–164, 1953.