Round-off error and exceptional behavior analysis of explicit Runge-Kutta methods
Sylvie Boldo, Florian Faissole, Alexandre Chapoutot

To cite this version:
Sylvie Boldo, Florian Faissole, Alexandre Chapoutot. Round-off error and exceptional behavior analysis of explicit Runge-Kutta methods. 2018. <hal-01883843>

HAL Id: hal-01883843
https://hal.archives-ouvertes.fr/hal-01883843
Submitted on 28 Sep 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Round-off error and exceptional behavior analysis of explicit Runge-Kutta methods

Sylvie Boldo\textsuperscript{1}, Florian Faissole\textsuperscript{1}, and Alexandre Chapoutot\textsuperscript{2}

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

\textsuperscript{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, \textit{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 \textit{fixed step-size} or \textit{variable step-size}, \textit{i.e.}, time instants are equidistant from each other or not; they can be \textit{single step} or \textit{multi-step}, \textit{i.e.}, they use one or several initial conditions; and they can be \textit{explicit} or \textit{implicit}, \textit{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.\footnote{https://fr.mathworks.com/help/simulink/index.html} In this article, we focus on explicit single-fixed-step-size methods belonging to the class of Runge-Kutta methods.
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 definition 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 consider 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

\[ \dot{y} = f(t, y) \quad \text{with} \quad y(0) = y_0 \quad \text{and} \quad t \in [0, t_{\text{end}}]. \]  

(1)

The function \( f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m \) is the dynamic, \( y \in \mathbb{R}^m \) is the vector of state variables and \( \dot{y} \) is the derivative of \( y \) with respect to time \( t \). We shall always assume at least that \( f \) is globally Lipschitz in \( y \), so Equation (1) admits a unique solution for a given initial condition \( y_0 \) [21]. The exact solution of Equation (1) is denoted by \( y(t; y_0) \). Usually, a closed form of \( y(t; y_0) \) is not computable and numerical methods to approximate it are used.
Runge-Kutta methods.

The goal of a numerical simulation when solving Equation (1) is to compute a sequence of time instants $0 = t_0 < t_1 < \cdots < t_N = t_{\text{end}}$ (not necessarily equidistant) and a sequence of states $y_0, \ldots, y_N$ such that $\forall \ell \in [0, N]$, $y_\ell \approx y(t_\ell; y_{\ell-1})$. These sequences are the result of a numerical integration method such as Runge-Kutta methods.

A Runge-Kutta method, starting from an initial value $y_t$ at time $t_\ell$ and a finite time horizon $h$, the step size, produces an approximated solution $y_{r+1}$ at time $t_{r+1}$, with $t_{r+1} - t_\ell = h$, of the solution $y(t_\ell; y_{t_\ell})$. Furthermore, to compute $y_{r+1}$, a Runge-Kutta method computes $s$ evaluations of $f$ 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

$$y_{t+1} = y_t + h \sum_{i=1}^{s} b_i k_i,$$

with $k_i$ defined by

$$k_i = f\left(t + c_i h, y_t + h \sum_{j=1}^{s} a_{ij} k_j\right).$$

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].

Depending on the form of the matrix $A$, made of the coefficients $a_{ij}$, a Runge-Kutta method can be

- *explicit*, such as the classical Runge-Kutta method of order 4. It means the computation of the intermediate $k_i$ only depends on the previous steps $k_j$ for $j < i$;
- *diagonally implicit*, such as SDIRK4, the diagonally implicit fourth-order method. It means the computation of an intermediate step $k_i$ involves the value $k_i$ and so non-linear systems in $k_i$ must be solved. A method is *singly diagonally implicit* if the coefficients on the diagonal are all equal;
- *fully implicit*, such as the Runge-Kutta fourth-order method with a Gauss-Legendre formula. It means the computation of intermediate steps involves the solution of a non-linear system of equations in all the values $k_i$ for $i = 1, 2, \cdots, s$.

The order of a Runge-Kutta method is $p$ if and only if the local truncation error, *i.e.*, the distance between the exact solution $y(t_\ell; y_{t-1})$ and the numerical solution $y_\ell$ is such that

$$y(t_\ell; y_{t-1}) - y_\ell = O(h^{p+1}).$$

Basically, the order of a Runge-Kutta method gives an indication on the magnitude of the method error for one integration step. Obviously, the higher the order of the method is, the more accurate the result is. Nevertheless, higher order methods are associated to an increase of the computation time because they need more stages $s$. There is a clear relation between the number of stages $s$ and the order $p$ for explicit methods up to order 4 that is $p = s$ for $s = 1, 2, 3, 4$. Otherwise, higher order explicit methods have order lower than their number of stages. The highest order is attained with implicit Gauss-Legendre Runge-Kutta methods for which $p = 2s$.

### 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-size values 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 $\lambda \in \mathbb{C}$):

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

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 $\lambda$. 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. In the general setting of high-dimensional linear systems where constants are represented by a matrix $M$, complex values may come from the diagonalization of $M$ in order to produce uncoupled linear differential equations. Nevertheless, it is sufficient to consider scalar problems, as defined in Equation (4), to study the linear stability property of numerical integration methods.

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

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

3
Example 1. Application of forward Euler’s method produces
\[ y_{n+1} = y_n + h\lambda y_n = (1 + h\lambda)y_n = R_{\text{Euler}}(h,\lambda)y_n. \]

Example 2. Application of RK2 (a.k.a. explicit midpoint) method on Equation (4) produces
\[
\begin{align*}
  k_1 & = \lambda y_n \\
  k_2 & = \lambda(y_n + 0.5hk_1) = (\lambda + 0.5h\lambda^2)y_n \\
\end{align*}
\]
\[ y_{n+1} = y_n + hk_2 = (1 + h\lambda + 0.5h^2\lambda^2)y_n = R_{\text{RK2}}(h,\lambda)y_n. \quad (6) \]

Example 3. Application of RK4 method on Equation (4) produces
\[
\begin{align*}
  k_1 & = \lambda y_n \\
  k_2 & = \lambda(y_n + 0.5hk_1) = (\lambda + 0.5h\lambda^2)y_n \\
  k_3 & = \lambda(y_n + 0.5hk_2) = (\lambda + 0.5h^2\lambda^2 + 0.5h^2\lambda^3)y_n \\
  k_4 & = \lambda(y_n + hk_3) = (\lambda + h\lambda^2 + 0.5h^2\lambda^3 + 0.5h^3\lambda^4)y_n \\
\end{align*}
\]
\[ y_{n+1} = y_n + h\left(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4\right) = \left(1 + \lambda h + \frac{1}{2}(\lambda h)^2 + \frac{1}{6}(\lambda h)^3 + \frac{1}{24}(\lambda h)^4\right)y_n = R_{\text{RK4}}(h,\lambda)y_n. \quad (7) \]

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 and a rational function for implicit Runge-Kutta methods. The region of absolute convergence of Equation (5) determines the different values of \(\lambda h\) for which the method is stable. These regions can be determined, see [28] for more details.

Note that for explicit Runge-Kutta methods the region of stability is defined by a closed space. In other terms, there is always a limitation on the value of the step size \(h\) to preserve the stability of the solution. Nevertheless, implicit methods such as Gauss-Legendre methods do not suffer of this limitation: they are stable for any value of the step size \(h\). They have a stronger stability property named \(A\)-stability for which the region of stability is the negative real-part half-space of the complex plane. Note also that the method SDIRK4 is also \(A\)-stable. It is the main reason why implicit Runge-Kutta methods are used to solve stiff problems. 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 this function and his FP counterpart \(\tilde{R}\), which is an algorithm depending on the implementation of the method.

In the rest of this article, variables mark with a tilde, e.g., \(\tilde{y}_n\) or \(\tilde{\lambda}\), will represent FP numbers. Operations \(\oplus, \odot, \oslash, \ominus\) are the FP counterpart of the mathematical operations, correctly rounded to nearest. To concisely denotes 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, \(\circ[a+b\times c\oslash d]\) stands for \(a\oplus(b\otimes c)\oslash d\).

We assume that \(h\) is an FP number, given without initial error. An explicit integration method corresponding to Equation (5) is implemented by the algorithm
\[
\begin{cases}
  \tilde{y}_0 \approx y_0 \in \mathbb{R} \\
  \forall n, \tilde{y}_{n+1} = \tilde{R}(h,\tilde{\lambda},\tilde{y}_n) \in \mathbb{R}.
\end{cases} \quad (8)
\]

Example 4. Let us consider the RK2 method on the linear IVP-ODE problem defined in Equation (4), an associated algorithm is defined by
\[
\tilde{y}_{n+1} = \circ \left[ \tilde{y}_n + h\tilde{\lambda}\tilde{y}_n + hh\frac{1}{2}\tilde{\lambda}\tilde{y}_n \right].
\]
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:

\[
\tilde{y}_{n+1} = \circ \left[ y_n + \frac{h}{6} \tilde{y}_n + \frac{h^2}{3} \tilde{\lambda} \tilde{y}_n + \frac{h^3}{6} \tilde{\lambda}^2 \tilde{y}_n + \frac{h^4}{24} \tilde{\lambda}^3 \tilde{y}_n \right].
\]

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 \( \tilde{R}(h, \lambda, y_n) \) are 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 these two values.

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 \( \epsilon \). We denote as \( \Omega \) the largest positive FP number, that is \( 2^{e_{\max}} (2 - 2^{-p}) \). We denote as \( \xi \) the smallest positive normal number, that is \( \xi = 2^{e_{\min}} \). We denote as \( \eta \) 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 \( u_{664} = 2^{-53} \), \( \Omega_{664} \approx 2^{1024} \), \( \xi_{664} = 2^{-1022} \) and \( \eta_{664} = 2^{-1074} \). Given a generic FP format, we want to bound the global round-off error of a Runge-Kutta method as defined above denoted by \( E_n \):

\[ E_n = \tilde{y}_n - y_n. \]  

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

\[ \epsilon_0 = |\tilde{y}_0 - y_0| \]

\[ \forall n \in \mathbb{N}^*, \quad \epsilon_n = |\tilde{R}(h, \tilde{\lambda}, \tilde{y}_{n-1}) - R(h, \lambda)\tilde{y}_{n-1}|. \]

With similar notations, our previous work proved the following result [7]: Let \( C > 0 \). Suppose that we have \( \forall n \in \mathbb{N}^*, \epsilon_n \leq C|\tilde{y}_n| \). Then \( \forall n \),

\[ |E_n| \leq (C + |R(h, \lambda)|)^n \left( \epsilon_0 + n \frac{C|y_0|}{C + |R(h, \lambda)|} \right). \]

But it 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 \( \tilde{y}_0 \) is neither an infinity, nor a NaN. Overflow is rather easily taken into account: it will be prevented in Section 5, as we will prove that no infinity can be produced provided a mild hypothesis on the input.

Underflow is heavier 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 \( \epsilon_n \leq Cu|\tilde{y}_{n-1}| \) with \( C \) being a small constant (such as 11 or 28). We now assume the local error is bounded by \( \epsilon_n \leq Cu|\tilde{y}_{n-1}| + D\eta \), with both \( C \) and \( D \) being small constants and \( \eta \) 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 6.2 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 (that is $nD\eta$), but we want to keep the preceding result without $\eta$ when no underflow occurred at all. For that, we provide a value $M$ such that, when $|y_n| \geq M$, the local error is bounded only by the relative error term: $\varepsilon_n \leq Cu|y_{n-1}|$ (without $D\eta$). For instance, if we were looking at a single multiplication, we have that if $|y_n| \geq 16\xi$, then $o(|y_n|/12)$ 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 $C \geq 0$, $M > 0$. Suppose that:

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

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

Proof. Let $n \in \mathbb{N}^*$. Suppose that $|y_n| \geq M$. We have:

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

By triangular inequality and by definition of local errors:

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

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

$M \leq |y_{n-1}| \leq (Cu + |R(h, \lambda)|)|y_{n-1}| \leq |y_{n-1}|$.

Thus, as long as no underflow occurs, the computed $y_i$ are non-increasing. However, as soon as 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 $y_0$ 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:

$|\circ(t) - t| \leq |t| \frac{u}{1 + u} + \frac{\eta}{2}$
Our main concern is the occurrence of "local overflow": local overflow means that there is an overflow in an intermediate computation of a given step of the method. For a Runge-Kutta method described by the iterative process 
\[
\begin{align*}
\tilde{y}_{n+1} &= \bigoplus_{i=0}^{s} \tilde{\alpha}_i \otimes \tilde{y}_n \quad \text{i.e.} \quad \tilde{y}_{n+1} = (\tilde{\alpha}_0 \otimes \tilde{y}_n \otimes (\tilde{\alpha}_1 \otimes \tilde{y}_n \otimes (\cdots \otimes (\tilde{\alpha}_s \otimes \tilde{y}_n) \cdots)) \end{align*}
\]
local overflow means that at least one sub-sum \( \bigoplus_{i=0}^{s} \tilde{\alpha}_i \otimes \tilde{y}_n \) (with \( k \leq s \)) overflows.

**Lemma 4. Local overflow**

Let \( s \in \mathbb{N} \). Let \( (\tilde{\alpha}_i)_{i \in \mathbb{N}} \) a sequence of FP numbers. Let \( y \) a FP number. Let \( 0 < V \). Suppose that:

- \( \sum_{i=0}^{s} |\tilde{\alpha}_i| \leq V \);
- \( |y| \leq \frac{\Omega}{1 + (s + 1)u} V \);
- \( (s + 1)u \leq 1 \);
- \( \xi \leq 2^{p_{\max}} - 1 \).

Then: \( \forall k \in \mathbb{N}, k \leq s, \)

\[
\left| \bigoplus_{i=0}^{k} \tilde{\alpha}_{i+s-k} \otimes y \right| \leq \Omega.
\]

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

\[
\left| \bigoplus_{i=0}^{k} \tilde{\alpha}_{i+s-k} \otimes y - \sum_{i=0}^{k} \tilde{\alpha}_{i+s-k} \otimes y \right| \leq \frac{ku}{1 + u} \sum_{i=0}^{k} |\tilde{\alpha}_{i+s-k} \otimes y|.
\]

Thus:

\[
\left| \bigoplus_{i=0}^{k} \tilde{\alpha}_{i+s-k} \otimes y \right| \leq \left( \frac{ku}{1 + u} + 1 \right) \sum_{i=0}^{k} |\tilde{\alpha}_{i+s-k} \otimes y|.
\]

Hence, by triangular inequalities inside the sum:

\[
\left| \bigoplus_{i=0}^{k} \tilde{\alpha}_{i+s-k} \otimes y \right| \leq \left( \frac{ku}{1 + u} + 1 \right) \sum_{i=0}^{k} |\tilde{\alpha}_{i+s-k} \otimes y|.
\]

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

\[
\sum_{i=0}^{k} |\tilde{\alpha}_{i+s-k} \otimes y| \leq \sum_{i=0}^{k} |y||\tilde{\alpha}_{i+s-k}|(1 + u) + \frac{\eta}{2}
\]

\[
\leq \frac{\Omega}{V(1 + (s + 1)u)} V(1 + u) + (k + 1)\frac{\eta}{2} = \frac{\Omega}{1 + (s + 1)u}(1 + u) + (k + 1)\frac{\eta}{2}.
\]

So, we have:

\[
\left| \bigoplus_{i=0}^{k} \tilde{\alpha}_{i+s-k} \otimes y \right| \leq \left( 1 + (k + 1)u \right) \left( \frac{\Omega}{1 + (s + 1)u}(1 + u) + (k + 1)\frac{\eta}{2} \right)
\]

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

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

\[
\left| \bigoplus_{i=0}^{k} \tilde{\alpha}_{i+s-k} \otimes y \right| < \Omega + (1 + (k + 1)u)(k + 1)\frac{\eta}{2}
\]

\[
\leq \Omega + \frac{\eta}{u} = \Omega + 2\xi \leq \Omega + 2 \times 2^{p_{\max}} - 1 = \Omega + \ulp(\Omega).
\]

Thus, as \( \bigoplus_{i=0}^{k} \tilde{\alpha}_{i+s-k} \otimes y \) is an FP number,

\[
\left| \bigoplus_{i=0}^{k} \tilde{\alpha}_{i+s-k} \otimes y \right| \leq \Omega.
\]
Table 1: Values of $V$, $s$, and $\Omega$ for classic methods in binary64

<table>
<thead>
<tr>
<th>Method</th>
<th>$V$</th>
<th>$s$</th>
<th>$\Omega$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler</td>
<td>3</td>
<td>2</td>
<td>$\frac{\Omega}{1+3u}$ $\simeq$ 5.99 $\times$ 10$^{307}$</td>
</tr>
<tr>
<td>RK2</td>
<td>5</td>
<td>3</td>
<td>$\frac{\Omega}{5+20u}$ $\simeq$ 3.6 $\times$ 10$^{307}$</td>
</tr>
<tr>
<td>RK4</td>
<td>16.5</td>
<td>11</td>
<td>$\frac{\Omega}{16.5+198u}$ $\simeq$ 1.09 $\times$ 10$^{307}$</td>
</tr>
</tbody>
</table>

As the sequence $(\tilde{y}_i)_{i \in \mathbb{N}}$ 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 $C \geq 0$, $M > 0$. Let a Runge-Kutta method described by the relation $\forall n \in \mathbb{N}, \tilde{y}_{n+1} = \bigoplus_{i=0}^{s} \tilde{\alpha}_i \otimes \tilde{y}_n$. Suppose that:

- $\sum_{i=0}^{s} |\tilde{\alpha}_i| \leq V$;
- $(s+1)u \leq 1$;
- $\forall n \in \mathbb{N}^*, |\tilde{y}_n| \geq M \Rightarrow \varepsilon_n \leq Cu|\tilde{y}_{n-1}|$;
- $M \leq \frac{\Omega}{1+(s+1)u}V$;
- $\xi \leq 2^{e_{\text{max}}-p}$.
- $|\tilde{y}_0| \leq \frac{\Omega}{1+(s+1)u}V$.

Then, forall $n \in \mathbb{N}$, no overflow occurs in the computation of $\tilde{y}_n$ by the Runge-Kutta method.

**Proof.** Let us prove the result by induction. If $n = 0$, as $\tilde{y}_0$ is finite, there is no overflow. Let $n$ be a given natural number. Suppose there is no overflow up to iteration $n$.

If $|\tilde{y}_n| \geq M$, then by Lemma 2, $|\tilde{y}_n| \leq |\tilde{y}_0| \leq \frac{\Omega}{1+(s+1)u}V$. If $|\tilde{y}_n| < M$ then $|\tilde{y}_n| < \frac{\Omega}{1+(s+1)u}V$ too. Thus, in both cases, by Lemma 4, forall $k \leq s$, we have:

$$\bigoplus_{i=0}^{k} \tilde{\alpha}_{i+k} \otimes \tilde{y}_n \leq \Omega.$$ 

Thus, no overflow occurs in the computation of $\tilde{y}_{n+1}$. 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 $|\tilde{y}_0|$ 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 really smaller than the values exhibited in Table 1.
- Furthermore, Theorem 5 assumes $\xi \leq 2^{e_{\text{max}}-p}$. Actually, all useful FP formats verify this assumption. For instance, in binary64, $\xi = 2^{-1022}$ and $2^{e_{\text{max}}-p} = 2^{970}$.

Underflow cannot be treated separately. In the next sections, we study its contribution to 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 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:

- \( |X_1 - \alpha_1 y| \leq C_1 u |y| + D_1 \eta \);
- \( |X_2 - \alpha_2 y| \leq C_2 u |y| + D_2 \eta \).

Then:

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

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

\[
|X_1 + X_2 - (\alpha_1 + \alpha_2) y| \\
\leq |X_1 + X_2 - (X_1 + X_2)| + |X_1 + X_2 - (\alpha_1 + \alpha_2) y| \\\n\leq \frac{u}{1 + u} \times |X_1 + X_2| + (C_1 u + C_2 u) |y| + (D_1 + D_2) \eta \\\n\leq \left( (C_1 + C_2) u + \frac{u}{1 + u} \times (|\alpha_1| + |\alpha_2| + (C_1 + C_2) u) \right) |y| + \eta \left( 1 + \frac{2u}{1 + u} \right) (D_1 + D_2) \\\n\leq \left( (C_1 + C_2) u + \frac{u}{1 + u} \times (|\alpha_1| + |\alpha_2| + (C_1 + C_2) u) \right) |y| + \eta (1 + u) (D_1 + D_2)
\]

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:

- \( |X_1 - \alpha_1 y| \leq C_1 u |y| + D_1 \eta \);
- \( |y| \geq M \Rightarrow |X_1 - \alpha_1 y| \leq C_1 u |y| \);
- \( |X_2 - \alpha_2 y| \leq C_2 u \);
- \( \left| \frac{x_1 - \alpha_2}{\alpha_2} \right| \leq P_2 < 1 \);
- \( M \geq \frac{\xi}{|\alpha_2| (1 - P_2)} \).

Then,

\[
|y| \geq M \Rightarrow |X_1 \oplus (X_2 \otimes y) - (\alpha_1 + \alpha_2) y| \\
\leq |y| \left( C_1 u + C_2 u + \frac{u}{1 + u} (|\alpha_1| + |\alpha_2| + 2C_1 u + 2C_2 u) + \left( \frac{u}{1 + u} \right)^2 (C_2 u + |\alpha_2|) \right)
\]

and

\[
|X_1 \oplus (X_2 \otimes y) - (\alpha_1 + \alpha_2) y| \\
\leq |y| \left( C_1 u + C_2 u + \frac{u}{1 + u} (|\alpha_1| + |\alpha_2| + 2C_1 u + 2C_2 u) + \left( \frac{u}{1 + u} \right)^2 (C_2 u + |\alpha_2|) \right) + \eta (1 + u) \left( \frac{1}{2} + D_1 \right).
\]

**Proof.**

- Assume \( |y| < M \).

The proof relies on an application of Lemma 6 with \( X_1, \alpha_1 \) and \( \alpha_2 \) being the same. But the \( X_2 \) of Lemma 6 is then \( X_2 \otimes y \). We have the expected assumptions on \( X_1, \alpha_1, C_1 \) and \( D_1 \). As for \( C_2 \) and
\[ D_2, \text{ we need to bound the error on } X_2 \otimes y: \]

\[ |X_2 \otimes y - \alpha_2 y| \leq |X_2 \otimes y - X_2 \times y| + |y| |X_2 - \alpha_2| \]

\[ \leq \frac{u}{1 + u} |X_2| |y| + \eta \frac{\gamma}{2} + |y| C_2 u \]

\[ \leq \frac{u}{1 + u} (C_2 u + |\alpha_2|) |y| + |y| C_2 u + \eta \frac{\gamma}{2} \]

\[ = |y| \left( C_2 u + \frac{u}{1 + u} (C_2 u + |\alpha_2|) \right) + \eta \frac{\gamma}{2}. \]

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

\[ |X_1 \otimes X_2 \otimes y - (\alpha_1 + \alpha_2)y| \]

\[ \leq (C_1 u + C_2 u + \frac{u}{1 + u} (C_2 u + |\alpha_2|)) + \]

\[ \frac{u}{1 + u} ((|\alpha_1| + |\alpha_2| + C_1 u + C_2 u + \frac{u}{1 + u} (C_2 u + |\alpha_2|)) |y| + \eta (1 + u) \left( \frac{\gamma}{2} + D_1 \right) \]

\[ = (C_1 u + C_2 u + \frac{u}{1 + u} ((|\alpha_1| + 2|\alpha_2| + C_1 u + 2C_2 u) + \]

\[ (\frac{u}{1 + u})^2 (C_2 u + |\alpha_2|)) |y| + \eta (1 + u) \left( \frac{\gamma}{2} + D_1 \right). \]

\[ \bullet \text{ Assume } |y| \geq M. \]

We know that \( |y| \geq \frac{\xi}{|x_2| (1 - \frac{\gamma}{2})} \). Moreover, \( \left| x_2 - \frac{\alpha_2}{\alpha_2} \right| \leq P_2 \) i.e. \( |X_2 - \alpha_2| \leq P_2 |\alpha_2| \) and thus:

\[ |\alpha_2| - |X_2| \leq P_2 |\alpha_2| \text{ i.e. } |X_2| \geq (1 - P_2) |\alpha_2| \]

and as \( P_2 < 1 \): \( |X_2 y| \geq (1 - P_2) |\alpha_2| \frac{\xi}{|x_2| (1 - \frac{\gamma}{2})} = \xi \)

So \( |X_2 \otimes y - X_2 y| \leq \frac{\eta}{1 + u} |X_2| |y| \) because the multiplication of \( X_2 \) and \( y \) does not underflow.

Then, as in the other case, we apply Lemma 6 with \( X_1, \alpha_1 \) and \( \alpha_2 \) being the same, \( X_2 \) being \( X_2 \otimes y \).

We have the right assumption on \( X_1, \alpha_1 \) and \( C_1 \). As for \( C_2 \), we need to bound the error on \( X_2 \otimes y: \)

\[ |X_2 \otimes y - \alpha_2 y| \leq |X_2 \otimes y - X_2 \times y| + |y| |X_2 - \alpha_2| \]

\[ \leq \frac{1}{1 + u} |X_2| |y| + |y| C_2 u \]

\[ \leq \frac{1}{1 + u} (C_2 u + |\alpha_2|) |y| + |y| C_2 u \]

\[ = |y| \left( C_2 u + \frac{u}{1 + u} (C_2 u + |\alpha_2|) \right). \]

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

\[ |X_1 \otimes X_2 \otimes y - (\alpha_1 + \alpha_2)y| \]

\[ \leq (C_1 u + C_2 u + \frac{u}{1 + u} (C_2 u + |\alpha_2|) + \]

\[ u(|\alpha_1| + |\alpha_2| + C_1 u + C_2 u) + \]

\[ \frac{u}{1 + u} (C_2 u + |\alpha_2|)) |y| \]

\[ = (C_1 + C_2) u \]

\[ + \frac{u}{1 + u} ((|\alpha_1| + 2|\alpha_2| + (C_1 + 2C_2) u) + \]

\[ \left( \frac{u}{1 + u} \right)^2 (C_2 u + |\alpha_2|)) |y|. \]

\[ \square \]

The hypotheses of this lemma may seem redundant. There is indeed a bound \( P_2 \) on the relative round-off error bound of \( \alpha_2 \) and a bound \( C_2 \) on its absolute error bound. Of course, one could deduce one from the other. But in our application, \( \alpha_2 \) will not be a known constant but will live in an interval, so we need
a correct 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, $\alpha_2$ may be $\frac{h^2\lambda^2}{4}$ and $X_2$ may be $\frac{h^2\lambda^2}{2}$. As $-3 \leq h\lambda \leq 0$, we only know that $0 \leq \alpha_2 \leq 1.5$ and we bound the round-off errors whatever the effective values of $h$ and $\lambda$. If we had the value of $\alpha_2$, 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 $\alpha_2$ 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 $\frac{u}{\eta}$ provided by Jeannerod and Rump [26]. If an underflow may have occurred, a term depending on $\eta$ 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]. 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 IEEE-754 format. To refine our results, we focus on stable methods and use region of stability for $h\lambda$ (see Section 2.1). 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 $-2 \leq h\lambda \leq -2^{-100}$ and $2^{-60} \leq h \leq 1$. Let $n \in \mathbb{N}$. Then: $|\tilde{y}_n| \geq \frac{\varepsilon}{2(1-2\eta)} \Rightarrow \varepsilon_{n+1} \leq 11.01u|\tilde{y}_n|$ and $\varepsilon_{n+1} \leq 11.01u|\tilde{y}_n| + (0.5 + u)\eta$.

**Proof.** This proof relies on an application of Lemma 7 with $X_1 = \tilde{y}_n$, $\alpha_1 = 1$, $C_1 = 0$ and $D_1 = 0$. And $X_2 = h\tilde{y}_n$, $\alpha_2 = h\lambda$. We then need a correct $C_2$. Using Gappa, we obtain $|X_2 - h\lambda| \leq 6 \times 2^{-53}$ from intervals on $h\lambda$ and $h$, so we choose $C_2 = 6$. Moreover, Gappa provides a bound on the relative error $|X_2 - h\lambda| \leq 2 \times 2^{-53}$. So we choose $P_2 = 2u$. The application of Lemma 7 then gives that $|\tilde{y}_n| \geq \frac{\varepsilon}{2(1-2\eta)}$ implies

$$\varepsilon_{n+1} \leq |\tilde{y}_n|(6u + u(1 + 2|h\lambda| + 12u) + u(6u + |h\lambda|)) + (0.5 + u)\eta.$$

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

$$\varepsilon_{n+1} \leq |\tilde{y}_n|(6u + u(1 + 2|h\lambda| + 12u) + u(6u + |h\lambda|)) + (0.5 + u)\eta.$$

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

$$\varepsilon_{n+1} \leq |\tilde{y}_n|(11u + 14u^2 + 6u^3) \leq 11.01u|\tilde{y}_n| + (0.5 + u)\eta,$$

and $|\tilde{y}_n| \geq \frac{\varepsilon}{2(1-2\eta)} \Rightarrow \varepsilon_{n+1} \leq |\tilde{y}_n|(11u + 14u^2 + 6u^3) \leq 11.01u|\tilde{y}_n|.$$

**Lemma 9. Bound on Runge-Kutta 2 local round-off errors.** Let us assume stability, that $-2 \leq h\lambda \leq -2^{-100}$ and $2^{-60} \leq h \leq 1$. Then for all $n$, $|\tilde{y}_n| \geq \frac{\varepsilon}{2(1-8\eta)} \Rightarrow \varepsilon_{n+1} \leq 28.01u|\tilde{y}_n|$ and $\varepsilon_{n+1} \leq 28.01u|\tilde{y}_n| + 1.01u$.

**Proof.** As explained, we have for the RK2 method $\tilde{y}_{n+1} = \circ \left[ \tilde{y}_n + h\lambda\tilde{y}_n + hh\frac{1}{2}h\lambda\tilde{y}_n \right]$. Stability means that $[1 + h\lambda + h\frac{1}{2}h\lambda^2] \leq 1$. We want to bound the corresponding $\varepsilon_n$. As in Euler’s method, we rely on Lemma 7. We put in Table 2 the corresponding $\alpha$, $C$, $P_2$ 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

$$|\circ \left[ hh\frac{1}{2}h\lambda \right] - \frac{h^2\lambda^2}{2} | \leq 2^{-49} = 16u$$
contrarily to Lemmas 8 and 9,
Here, $\xi$ is not the $\xi$-value corresponding to the last term.
so that $\xi = \frac{\epsilon}{2(1 - 8u)}$. Then for all $u$, $\|y_{n}\| > 0.5(1 - 4u)$ \(\Rightarrow\) $\xi_{n+1} \leq 164u |y_{n}|$ and $\xi_{n+1} \leq 164u |y_{n}| + 5.6\eta$.

Lemma 10. Bound on Runge-Kutta 4 local round-off errors Let us assume stability, that $-3 \leq h \lambda \leq -2^{-100}$ and $2^{-60} \leq h \leq 1$. Then for all $u$, $|y_{n}| > 0.5(1 - 4u)$ \(\Rightarrow\) $\xi_{n+1} \leq 164u |y_{n}|$ and $\xi_{n+1} \leq 164u |y_{n}| + 5.6\eta$.

Proof. The RK4 method is defined by Equation (9). In this case, stability means $1 + h \lambda + \frac{h^2 \lambda^2}{2} + \frac{h^3 \lambda^3}{12} + \frac{h^4 \lambda^4}{24}$ \(\leq\) 1 which leads to $-3 \leq h \lambda \leq 0$. Let us bound the local error $\xi_{n}$. As previously, we rely on Lemma 7 and on Gappa under the given hypotheses on $h \lambda$ and $h$. We put in Figure 3 the corresponding $\xi$, $C$, $P_2$, $D$ 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.

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.

<table>
<thead>
<tr>
<th>FP term</th>
<th>$y_n$</th>
<th>$P_2$</th>
<th>$M_{loc}$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h \otimes \lambda$</td>
<td>$0$</td>
<td>$-\lambda$</td>
<td>$0$</td>
<td>$-\lambda$</td>
</tr>
<tr>
<td>$\otimes \frac{y_n + h\lambda y_n}{2}$</td>
<td>$6$</td>
<td>$2u$</td>
<td>$-\lambda$</td>
<td>$-\lambda$</td>
</tr>
<tr>
<td>$\otimes \frac{y_n + h\lambda y_n}{2}$</td>
<td>$11 + 15u$</td>
<td>$-\lambda$</td>
<td>$\frac{\xi}{2(1 - 8u)}$</td>
<td>$0.5 + u$</td>
</tr>
<tr>
<td>$\frac{y_n + \lambda y_n}{2}$</td>
<td>$16$</td>
<td>$8u$</td>
<td>$-\lambda$</td>
<td>$-\lambda$</td>
</tr>
<tr>
<td>$\frac{y_n + \lambda y_n}{2}$</td>
<td>$28.01$</td>
<td>$-\lambda$</td>
<td>$\frac{\xi}{2(1 - 8u)}$</td>
<td>$1.01$</td>
</tr>
</tbody>
</table>

Table 2: Steps for the local round-off error bound of RK2.
Lemma 11. Let $C \geq 0$. 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

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

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

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

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

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

As $C \geq 0$, this inequality holds and so the result. \qed

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. Absolute error of explicit one-step method. Let $C \geq 0$, $D \geq 0$, $M > 0$. Suppose that:

- $\forall n \in \mathbb{N}^*, |\tilde{y}_n| \geq M \Rightarrow \varepsilon_n \leq Cu|\tilde{y}_{n-1}|$;
- $\forall n \in \mathbb{N}^*, \varepsilon_n \leq Cu|\tilde{y}_{n-1}| + Dn$;
- $0 < Cu + |R(h, \lambda)| < 1$.

Then:

- $\forall n, |\tilde{y}_n| \geq M \Rightarrow |E_n| \leq (Cu + |R(h, \lambda)|)^n \left( \varepsilon_0 + n \frac{Cu|y_0|}{Cu + |R(h, \lambda)|} \right)$.
- $\forall n, |E_n| \leq (Cu + |R(h, \lambda)|)^n \left( \varepsilon_0 + n \frac{Cu|y_0|}{Cu + |R(h, \lambda)|} \right) + nDn$.

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

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

and both results hold.

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

- Assume $|\tilde{y}_{n+1}| \geq M$.

By Lemma 1, we have $|\tilde{y}_n| \geq M$. 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:

$$|E_{n+1}| \leq |\tilde{y}_{n+1} - R(h, \lambda)\tilde{y}_n| + |R(h, \lambda)\tilde{y}_n - R(h, \lambda)y_n| = \varepsilon_{n+1} + |R(h, \lambda)||E_n|.$$ 

Let us use the hypothesis on $\varepsilon_{n+1}$ to bound it:

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

Then, using Equation (12):
Using the induction hypothesis, we now have

\[ |E_{n+1}| \leq (C_u + |R(h, \lambda)|)(C_u + |R(h, \lambda)|)^n \times \left( \varepsilon_0 + n \frac{C_u |y_0|}{C_u + |R(h, \lambda)|} \right) + C_u |R(h, \lambda)|^n |y_0| = (C_u + |R(h, \lambda)|)^{n+1} \left( \varepsilon_0 + n \frac{C_u |y_0|}{C_u + |R(h, \lambda)|} \right) + C_u |R(h, \lambda)|^n |y_0|. \]

Applying Lemma 11, we get:

\[ |E_{n+1}| \leq (C_u + |R(h, \lambda)|)^{n+1} \left( \varepsilon_0 + (n+1) \frac{C_u |y_0|}{C_u + |R(h, \lambda)|} \right). \]

Assume \( |\tilde{y}_{n+1}| < M \).

Bounding \( e_{n+1} \) from the assumptions, we have: \( |E_{n+1}| \leq |e_{n+1}| + |R(h, \lambda)||E_n| \)

\[ \leq C_u |\tilde{y}_n| + D\eta + |R(h, \lambda)||E_n| \]

\[ = C_u |\tilde{y}_n| + C_u |R(h, \lambda)|^n |y_0| + D\eta + |R(h, \lambda)||E_n| \]

\[ = (C_u + |R(h, \lambda)|)|E_n| + C_u |R(h, \lambda)|^n |y_0| + D\eta. \]

Using the induction hypothesis, we now have

\[ |E_{n+1}| \leq (C_u + |R(h, \lambda)|)^n \times \left( (C_u + |R(h, \lambda)|)^n \left( \varepsilon_0 + n \frac{C_u |y_0|}{C_u + |R(h, \lambda)|} \right) + nD\eta \right) + C_u |R(h, \lambda)|^n |y_0| + D\eta \]

\[ = (C_u + |R(h, \lambda)|)^{n+1} \left( \varepsilon_0 + n \frac{C_u |y_0|}{C_u + |R(h, \lambda)|} \right) + C_u |R(h, \lambda)|^n |y_0| + (n+1)D\eta. \]

Applying Lemma 11, we get:

\[ |E_{n+1}| \leq (C_u + |R(h, \lambda)|)^{n+1} \left( \varepsilon_0 + (n+1) \frac{C_u |y_0|}{C_u + |R(h, \lambda)|} \right) + (n+1)D\eta. \]

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 \( \eta \).

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

\[ \left| \frac{\tilde{y}_n - y_0}{y_0} \right| \leq \left( \frac{C_u + |R(h, \lambda)|}{|R(h, \lambda)|} \right)^n \left( \left| \frac{\tilde{x}_0 - y_0}{y_0} \right| + n \frac{C_u}{C_u + |R(h, \lambda)|} \right). \]

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

\[ \left| \frac{\tilde{y}_n - y_0}{y_0} \right| \leq \left| \frac{\tilde{x}_0 - y_0}{y_0} \right| + n \frac{C_u}{|R(h, \lambda)|}. \] (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 \( \tilde{y}_n \).
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 $C = 11.01$, $D = 1 + u$ and $M = \frac{4}{u + 2}$.

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

$$|E_n| \leq (11.01u + |R(h, \lambda)|)^n \left(\varepsilon_0 + n \frac{11.01u|y_0|}{11.01u + |R(h, \lambda)|}\right) + n(1 + u)\eta$$

and

$$|\tilde{y}_n| \geq \frac{\xi}{2(1 - 2^n)} \Rightarrow |E_n| \leq (11.01u + |R(h, \lambda)|)^n \left(\varepsilon_0 + n \frac{11.01u|y_0|}{11.01u + |R(h, \lambda)|}\right)$$

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 $-2 \leq h\lambda \leq -2^{-100}$ and $2^{-60} \leq h \leq 1$. Then $\forall n$,

$$|E_n| \leq (28.01u + |R(h, \lambda)|)^n \left(\varepsilon_0 + n \frac{28.01u|y_0|}{28.01u + |R(h, \lambda)|}\right) + 2.01n\eta$$

and

$$|\tilde{y}_n| \geq \frac{\xi}{2(1 - 8n)} \Rightarrow |E_n| \leq (28.01u + |R(h, \lambda)|)^n \left(\varepsilon_0 + n \frac{28.01u|y_0|}{28.01u + |R(h, \lambda)|}\right).$$

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 $-3 \leq h\lambda \leq -2^{-100}$ and $2^{-60} \leq h \leq 1$. Then $\forall n$,

$$|E_n| \leq (164u + |R(h, \lambda)|)^n \left(\varepsilon_0 + n \frac{164u|y_0|}{164u + |R(h, \lambda)|}\right) + 11.01n\eta$$

and

$$|\tilde{y}_n| \geq \frac{\xi}{2(1 - 4n)} \Rightarrow |E_n| \leq (164u + |R(h, \lambda)|)^n \left(\varepsilon_0 + n \frac{164u|y_0|}{164u + |R(h, \lambda)|}\right).$$

8 Related work

In this section, a review on the previous work on the numerical accuracy of one-step numerical integration methods is given.

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 [4,15,19,20,34,37] (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 [4,19,20,34]. 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., $h \odot f$, is not exactly computed and proposes a problem-dependant 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 truncature error is given to understand what kind of error is preponderant during the numerical integration.

[16,22,27,38,39] 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
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, $h = \frac{1}{4\pi}$ and we compute $n = 1,000$ iterations. To compute the wanted errors, we have compared them with a multi-precision computation of 1,000 bits 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 $(Cu + |R(h, \lambda)|)^n \approx n\frac{Cu[|n|]}{C u[|R(h, \lambda)|]}$ 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 1,000-bit MPFR exp function), the round-off error bound and the real error bound. 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 $\lambda$ is then a matrix and 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 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 to find 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


