Round-off Error Analysis of Explicit One-Step Numerical Integration Methods

Sylvie Boldo, Florian Faissole, Alexandre Chapoutot

Round-off Error Analysis of Explicit One-Step Numerical Integration 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

Ordinary differential equations are ubiquitous in scientific computing. Solving exactly these equations is usually not possible, except for special cases, hence the use of numerical schemes to get a discretized solution. We are interested in such numerical integration methods, for instance Euler's method or the Runge-Kutta methods. As they are implemented using floating-point arithmetic, round-off errors occur. In order to guarantee their accuracy, we aim at providing bounds on the round-off errors of explicit one-step numerical integration methods. Our methodology is to apply a fine-grained analysis to these numerical algorithms. Our originality is that our floating-point analysis takes advantage of the linear stability of the scheme, a mathematical property that vouches the scheme is well-behaved.

1 Introduction

Numerical simulation is an essential tool nowadays to study, design and verify complex systems appearing in various domains such as cars, ships, airplanes, rockets, satellites, nuclear plants. Numerical simulation is used to solve mathematical problems such as *Initial Value Problems* (IVP) for *Ordinary Differential Equations* (ODEs). Such class of equations is important to model the temporal evolution of physical quantities, *e.g.*, speed or temperature, and usually they come from the application of physical laws, as for example the Newton's second law.

As a closed form solution of IVP-ODEs does not exist in general, numerical integration methods are applied. The overall process followed by numerical integration methods is to build step by step, from initial conditions, an approximation of the solution of IVP-ODEs for particular time instants. There is a wide variety of numerical integration methods depending on different features. 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. For example, the well known forward Euler's method is an explicit single-step and fixed step-size numerical integration method while methods based on *Backward Difference Formula* (BDF) are implicit multi-step and usually variable step-size methods, see [12] for more details. In this article, we focus our work on explicit single-step and fixed-step size methods belonging to the class of Runge-Kutta methods.

All these numerical integration methods have different properties, which make them suitable for different classes of IVP ODEs as stiff and non-stiff problems. In consequence, numerical simulation tools usually provide several integration methods to cope with a large class of problems. For example, Simulink is a tool to model and simulate dynamical systems widely used in the industry, it provides 13 numerical integration methods among them 11 belong to the Runge-Kutta family. The main scientific work on numerical integration methods is to develop new methods which are able to deal efficiently with the largest class of problems. That is why most of the research is focused on increasing the order of the method (see Section 2 for a more precise definition) while keeping the computation complexity as low as possible, and on defining more stable numerical methods (see Section 2 for a more precise definition). The stability concept has many definitions but, mainly, in the context of numerical integration methods, the stability depends on the step-size of the temporal discretization. A method is stable for a particular class of stable IVP ODEs, *e.g.*, IVP whose solutions converge to zero, if there exists a step-size value which makes the numerical solution preserving the stability of the IVP ODE solution,

e.g., the numerical solution also converges towards zero. We refer to [16] for a more complete presentation of stability notions of numerical integration methods.

Surprisingly, numerical accuracy problems in computations have been studied even before computers when considering hand calculations [8]. Even if mathematicians are aware of the round-off errors, they are usually dismissing this problem. This may be due to the fact that they do not have the tools to bound efficiently such errors, contrarily to method errors they have been taught. Usually, some rule of thumb applies [19]:

> *When stable difference equations are used, the rounding errors are not amplified as time goes on; they merely accumulate roughly in proportion to the square root of the number of steps in the calculation.*

But no evidence is given to support these claims. Our goal is to verify this kind of statement and exhibit the needed hypotheses.

There are other works where more precise bounds are given. In many cases, probabilistic bounds are provided to support the previous rule of thumb [14]: assuming that the round-off error of one step is a random variable with mean zero and variance proportional to the square of the round-off unit, the error contribution due to round-off will grow like (Brownian motion) the square-root of time. Unfortunately, round-off errors are not always well-behaved for probabilistic analysis. This is noted especially for Runge-Kutta methods, where round-off error experimentally seems to be linear, and accuracy improvement is obtained using error compensation for the method coefficients [13].

Another work bounds the round-off errors using a big O of the machine epsilon [11]. But as the machine epsilon is also a constant, this does not give a useful bound as we do not have any idea of the constant behind the big O. Modern techniques from Computer Science mostly rely on interval analysis [7, 1, 2], even if stochastic arithmetic was once used [3]. The error bounds are proved, but they are not very tight as they do not make use of the mathematical properties of the numerical methods. A nice solution is to use Taylor models to get bounds on both the method and the round-off errors [4, 18]. This greatly limits the dependency problem, but it does not take the numerical method properties, especially stability, into account. We aim at providing better error bounds by a very fine-grained analysis of the algorithms and by taking the mathematical properties, mostly the stability of the method, into account.

This article is organized as follows. Section 2 describes the numerical methods we are interested in, their uses and their properties. Section 3 gives results on the global error of a large category of methods, assuming we can bound the local errors of these methods. Section 4 then presents how to bound these local round-off errors on concrete examples: Euler's method and higher-order Runge-Kutta methods. Section 5 gives a generic methodology for bounding the round-off error of the methods we are interested in. Section 6 concludes and gives some perspectives.

2 Properties of numerical integration methods

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 flow, $\mathbf{y} \in \mathbb{R}^m$ 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 [12] for a given initial condition $\mathbf{y}_0$. The exact solution of Equation (1) is denoted by $\mathbf{y}(t; \mathbf{y}_0)$.

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 $\mathbf{y}_0$, ..., $\mathbf{y}_N$ such that $\forall \ell \in [0, N]$, $\mathbf{y}_\ell \approx \mathbf{y}(t_\ell; \mathbf{y}_{\ell-1})$. These sequences are the result of a numerical integration method such as Runge-Kutta methods.

A Runge-Kutta method, starting from an initial value $\mathbf{y}_\ell$ 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* [12] which has the form

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} \equiv \begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b} \end{array} .$$

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

(a) RK4

$$\begin{array}{c|ccccc} \frac{1}{4} & \frac{1}{4} & & & & \\ \frac{3}{4} & \frac{1}{2} & \frac{1}{4} & & & \\ \frac{11}{20} & \frac{17}{50} & \frac{-1}{25} & \frac{1}{4} & & \\ \frac{1}{2} & \frac{371}{1360} & \frac{-137}{2720} & \frac{15}{544} & \frac{1}{4} & \\ 1 & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \frac{1}{4} \\ \hline & \frac{25}{24} & \frac{-49}{48} & \frac{125}{16} & \frac{-85}{12} & \frac{1}{4} \end{array}$$

(b) SDIRK4

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

(c) Gauss-Legendre

Figure 1: Different kinds of Runge-Kutta methods

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

- *explicit*, such as the classical Runge-Kutta method of order 4 given in Figure 1(a). It means the computation of the intermediate $\mathbf{k}_i$ only depends on the previous steps $\mathbf{k}_j$ for $j < i$;
- *diagonally implicit*, such as the diagonally implicit fourth-order method given in Figure 1(b). It means the computation of an intermediate step $\mathbf{k}_i$ involves the value $\mathbf{k}_i$ and so non-linear systems in $\mathbf{k}_i$ 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 given in Figure 1(c). It means the computation of intermediate steps involves the solution of a non-linear system of equations in all the values $\mathbf{k}_i$ 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 $\mathbf{y}(t_\ell; \mathbf{y}_{\ell-1})$ and the numerical solution $\mathbf{y}_\ell$ 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. 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 $p = 2s$.

Another important feature 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 a non empty set of 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 $\lambda \in \mathbb{C}$):

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

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. 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

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

Example 1. *Application of forward Euler's method on Equation* (4) *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 RK4 method on Equation* (4) *produces*

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

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

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 it is a rational function for implicit Runge-Kutta methods. The region of absolute convergence of Equation (5) determines the different values of λh for which the method is stable. Figure 2 represents regions of convergence of the stability functions of explicit Runge-Kutta methods up to order 4.

Many other explicit Runge-Kutta methods exist, that satisfy the condition $|R(h, \lambda)| < 1$, and the associated regions of stability can also be determined, see [16] for more details.

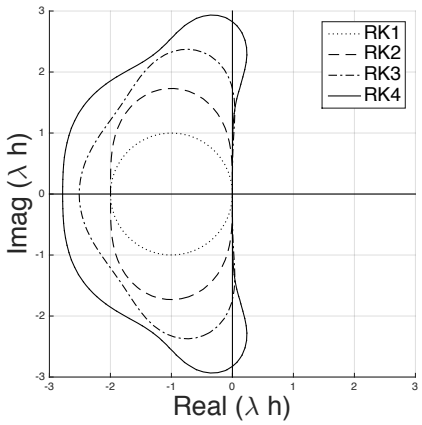

Figure 2: Linear stability region of explicit Runge-Kutta methods up to order 4.

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, that is 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 stable methods.

3 From local to global round-off errors

We assume a radix 2 floating-point (FP) format with precision p. We assume rounding to nearest, and we denote by $u = 2^{-p}$ the machine epsilon. For the sake of readability, we use the following notation: $\circ[\cdots]$ means that each operation inside the brackets is a rounded operations. For example, $\circ[a + b + c \times d]$ is $a \oplus b \oplus c \otimes d$. We also assume there is neither underflow, nor overflow.

3.1 General description of our schemes

Below, 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 $\widetilde{R}$, which is an algorithm depending on the implementation of the method. An explicit method corresponding to the Equation (5) is implemented by the algorithm

$$\begin{cases} \widetilde{y_0} \approx y_0 \in \mathbb{R} \\ \forall n, \widetilde{y_{n+1}} = \widetilde{R}(\widetilde{h}, \widetilde{\lambda}, \widetilde{y_n}) \in \mathbb{R} \ . \end{cases} \tag{7}$$

Example 3. *Let us consider the RK4 method whose Butcher tableau is given in Figure 1(a). The exact iterative relation $y_{n+1} = R_{RK4}(h, \lambda)y_n$ is given in Equation* (6)*. However, the FP implementation automatically obtained is more intricate:*

$$\begin{aligned} \widetilde{y_{n+1}} = \circ \Bigg[\widetilde{y_n} + \frac{\widetilde{h}}{6}\widetilde{\lambda}\widetilde{y_n} + \frac{\widetilde{h}}{3}\widetilde{\lambda}\widetilde{y_n} + \frac{\widetilde{h}^2}{6}\widetilde{\lambda}^2\widetilde{y_n} + \frac{\widetilde{h}}{3}\widetilde{\lambda}\widetilde{y_n} + \\ \frac{\widetilde{h}^2}{6}\widetilde{\lambda}^2\widetilde{y_n} + \frac{\widetilde{h}^3}{12}\widetilde{\lambda}^3\widetilde{y_n} + \frac{\widetilde{h}}{6}\widetilde{\lambda}\widetilde{y_n} + \frac{\widetilde{h}^2}{6}\widetilde{\lambda}^2\widetilde{y_n} + \\ \frac{\widetilde{h}^3}{12}\widetilde{\lambda}^3\widetilde{y_n} + \frac{\widetilde{h}^4}{24}\widetilde{\lambda}^4\widetilde{y_n} \Bigg] \ . \end{aligned} \tag{8}$$

The implementation is correct when the algorithm corresponds to the numerical method. It means that, if $\widetilde{R}(h, \lambda, y_n)$ were computed without rounding error, it would be mathematically equal to $R(h, \lambda)y_n$.

3.2 Stability and the sum of the epsilons

To bound the global round-off error after n iterations of the scheme, the first step is to express this error according to all previous local round-off errors. We denote by $\varepsilon_n > 0$ the local error resulting from iteration n. We set $\varepsilon_0 = |\widetilde{y_0} - y_0|$ and

$$\forall n \in \mathbb{N}^*, \quad \varepsilon_n = |\widetilde{R}(\widetilde{h}, \widetilde{\lambda}, \widetilde{y_{n-1}}) - R(h, \lambda)\widetilde{y_{n-1}}| \ . \tag{9}$$

E_n stands for the global round-off error after n iterations, *i.e.*,

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

We first prove intermediate lemmas to relate local errors and global errors, which will be useful later on.

Lemma 1. *For all $n \in \mathbb{N}$, $|E_{n+1}| \leqslant \varepsilon_{n+1} + |R(h, \lambda)|\,|E_n|$.*

Proof. The proof is a simple unfolding of the definitions given in Equations (9) and (10) and a triangular inequality:

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

and thus the result. □

We now prove that the global error may be bounded by the sum of the last error ε_n and the errors accumulated in the previous iterations:

Lemma 2. *For all $n \in \mathbb{N}$, $|E_n| \leqslant \sum_{k=0}^{n} |R(h, \lambda)|^{n-k}\varepsilon_k$.*

Proof. We do an induction on n. For $n = 0$, we have easily $|E_0| = \sum_{k=0}^{0} |R(h, \lambda)|^{0-k}\varepsilon_k = |R(h, \lambda)|^0\varepsilon_0 = \varepsilon_0$.

Suppose that the property is true for $n \in \mathbb{N}$. Then by Lemma 1 and the hypothesis of induction, one has

$$\begin{aligned} |E_{n+1}| &\leqslant \varepsilon_{n+1} + |R(h, \lambda)|\,|E_n| \\ &\leqslant \varepsilon_{n+1} + |R(h, \lambda)| \sum_{k=0}^{n} |R(h, \lambda)|^{n-k}\varepsilon_k \\ &\leqslant \varepsilon_{n+1} + \sum_{k=0}^{n} |R(h, \lambda)|^{n+1-k}\varepsilon_k \\ &= \sum_{k=0}^{n+1} |R(h, \lambda)|^{n+1-k}\varepsilon_k \end{aligned}$$

and thus the result. □

There may be an accumulation of round-off errors, but the global error does not diverge too fast. If the method is stable, 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. Note that the result applies to a large range of one-step explicit methods. In contrast, the bounds on the local round-off errors ε_i strongly depend on the chosen method, as shown in Section 4.

3.3 Global error

In order to get a more precise global error bound, we suppose we can bound the local error ε_n at iteration $n \in \mathbb{N}^*$ by a constant of the form $C \times |\widetilde{y_{n-1}}|$. It is indeed reasonable and we give such constants C for classic methods in Section 4. Then it is possible to bound the global error of the scheme.

Theorem 3. Absolute error of explicit one-step method. *Let $C > 0$. Suppose that $\forall n \in \mathbb{N}^*, \varepsilon_n \leqslant C|\widetilde{y_{n-1}}|$. Then $\forall n$,*

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

Proof. We do an induction on n. For $n = 0$, we have:
$|E_0| = \varepsilon_0 = (C + |R(h,\lambda)|)^0 \left(\varepsilon_0 + 0\frac{C|y_0|}{C+|R(h,\lambda)|}\right)$.

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

$$\begin{aligned}
\varepsilon_{n+1} &\leqslant C|\widetilde{y_n}| \\
&\leqslant C\left(|\widetilde{y_n} - y_n| + |y_n|\right) \\
&= C|E_n| + C\left|R(h,\lambda)\right|^n |y_0| \ . \qquad (11)
\end{aligned}$$

Let us assume the induction hypothesis and let us try to bound $|E_{n+1}|$, first by using Lemma 1 and Equation (11)

$$\begin{aligned}
|E_{n+1}| &\leqslant \varepsilon_{n+1} + |R(h,\lambda)|\,|E_n| \\
&\leqslant C\,|E_n| + C\,|R(h,\lambda)|^n\,|y_0| + |R(h,\lambda)|\,|E_n| \\
&= (C + |R(h,\lambda)|)|E_n| + C\,|R(h,\lambda)|^n\,|y_0| \ .
\end{aligned}$$

Using the induction hypothesis, we now have

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

To end the proof, we have to check that this latest value is smaller than $(C + |R(h,\lambda)|)^{n+1} \left(\varepsilon_0 + (n+1)\frac{C|y_0|}{C+|R(h,\lambda)|}\right)$. By simplifying the $(C + |R(h,\lambda)|)^{n+1}\varepsilon_0$ value, it remains to prove that

$$\begin{aligned}
(C + |R(h,\lambda)|)^{n+1} n\frac{C|y_0|}{C + |R(h,\lambda)|} + C\,|R(h,\lambda)|^n\,|y_0| & \\
\leqslant (C + |R(h,\lambda)|)^{n+1}\,(n+1)\frac{C|y_0|}{C + |R(h,\lambda)|} \ .&
\end{aligned}$$

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

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

This is true if $y_0 = 0$ or $C = 0$. In the other cases, we have left to prove that

$$\begin{aligned}
|R(h,\lambda)|^n &\leqslant (C + |R(h,\lambda)|)^{n+1}\frac{1}{C + |R(h,\lambda)|} \\
&= (C + |R(h,\lambda)|)^n \ .
\end{aligned}$$

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

This theorem bounds the global round-off error with a rather simple formula. As soon as local round-off errors may be bounded as expected (as it is the case in the next section), we have a reasonable bound on the final round-off error of such computations. This error is exhibited for some common methods in the next section.

Note also that from Theorem 3 we can bound the relative round-off error[1]:

$$\left|\frac{\widetilde{y_n}-y_n}{y_n}\right| \leqslant \left(\frac{C+|R(h,\lambda)|}{|R(h,\lambda)|}\right)^n \left(\frac{\varepsilon_0}{|y_0|}+n\frac{C}{C+|R(h,\lambda)|}\right) \ .$$

If we assume that $C \ll |R(h,\lambda)|$, let us note $\varepsilon_0^{rel} = \frac{\varepsilon_0}{|y_0|}$, this reduces to

$$\left|\frac{\widetilde{y_n}-y_n}{y_n}\right| \lesssim \varepsilon_0^{rel} + n\frac{C}{|R(h,\lambda)|} \ . \tag{12}$$

Note that C is never above a small constant number of u (at most $200u$ for our examples, see Section 4). Note also that the computation of $\widetilde{y_n}$ requires many FP operations whose number is proportional to the number of iterations n, for example, for RK4 more than 70 flops per iteration are involved. Thus, the bound provided by Theorem 3 and Equation (12) is very tight.

4 Bounds on local and global round-off errors

The preceding section has demonstrated how to bound the global error of a scheme from its local round-off errors. In particular, if the local round-off errors are reasonable (meaning a small multiple of u), then the global error will be linear in the number of iterations, which is about the best we might expect, given the number of flops involved. It remains to bound the local error of the methods. This section is organized as follows. Section 4.1 gives some preliminary results. Then Section 4.2 describes Euler's method and bounds its local and global round-off errors. Section 4.3 describes Runge-Kutta methods, and errors are given for both Runge-Kutta 2 and Runge-Kutta 4 methods. We now assume that computations are done in the *binary64* IEEE-754 format.

4.1 Floating-point preliminaries

Before going into the various methods we are interested in, we focus on bounding some local round-off errors to get a sharp bound on ε_n. Let us first consider the addition of two FP values that both roughly correspond to multiple of a given value y. Then the error can be bounded from the previous errors and the multiplicative factors.

Lemma 4. *Let $y \in \mathbb{R}$. Let $C_1, C_2 \in \mathbb{R}_+^*$. Let $\alpha_1, \alpha_2 \in \mathbb{R}$. Let X_1 and X_2 be FP numbers such that $|X_1 - \alpha_1 y| \leqslant C_1|y|$ and $|X_2 - \alpha_2 y| \leqslant C_2|y|$. Then*
$|X_1 \oplus X_2 - (\alpha_1+\alpha_2)y|$
$$\leqslant (C_1 + C_2 + u \times (|\alpha_1| + |\alpha_2| + C_1 + C_2))\ |y| \ .$$

Proof. The proof is quite straightforward from basic FP facts and the hypotheses

$$\begin{aligned}
&| X_1 \oplus X_2 - (\alpha_1+\alpha_2)y \ | \\
&\quad \leqslant |X_1 \oplus X_2 - (X_1 + X_2)| + |X_1 + X_2 - (\alpha_1+\alpha_2)y| \\
&\quad \leqslant u \times |X_1 + X_2| + (C_1+C_2)|y| \\
&\quad \leqslant u \times ((C_1 + |\alpha_1|)|y| + (C_2 + |\alpha_2|)|y|) + (C_1+C_2)|y| \\
&\quad = (C_1 + C_2 + u \times (|\alpha_1| + |\alpha_2| + C_1 + C_2))\ |y| \ .
\end{aligned}$$

□

Then we consider a similar kind of lemma, but the operations are a FP multiplication by the scaling factor followed by an addition.

Lemma 5. *Let $y \in \mathbb{R}$. Let $C_1, C_2 \in \mathbb{R}_+^*$. Let $\alpha_1, \alpha_2 \in \mathbb{R}$. Let X_1 and X_2 be FP numbers such that both $|X_1 - \alpha_1 y| \leqslant C_1|y|$ and $|X_2 - \alpha_2| \leq C_2$. Then*
$|X_1 \oplus (X_2 \otimes y) - (\alpha_1+\alpha_2)y|$
$$\begin{aligned}
&\leqslant |y|\,(C_1 + C_2 + u\,(|\alpha_1| + 2|\alpha_2| + C_1 + 2C_2) \\
&\qquad + u^2\,(C_2 + |\alpha_2|)) \ .
\end{aligned}$$

Proof. The proof relies on an application of Lemma 4 with X_1, α_1, and α_2 being the same. But the X_2 of Lemma 4 is then $X_2 \otimes y$. We have the right assumption on X_1, α_1, and C_1. As for C_2, we need to bound the error on $X_2 \otimes y$:

[1] the published version of this article contains an error which is fixed in the formula below

$$
\begin{aligned}
|X_2 \otimes y - \alpha_2 y| &\leqslant |X_2 \otimes y - X_2 \times y| + |y|\,|X_2 - \alpha_2| \\
&\leqslant u\,|X_2|\,|y| + |y|\,C_2 \\
&\leqslant u\,(C_2 + |\alpha_2|)\,|y| + |y|\,C_2 \\
&= |y|\ (C_2 + u(C_2 + |\alpha_2|)) \quad .
\end{aligned}
$$

So we can take for C_2 the value $C_2 + u\,(C_2 + |\alpha_2|)$ and have all the required hypotheses. Applying Lemma 4, we get

$$
\begin{aligned}
&|X_1 \oplus X_2 \otimes y - (\alpha_1 + \alpha_2)y| \\
&\quad \leqslant \big(C_1 + C_2 + u(C_2 + |\alpha_2|) + \\
&\qquad u(|\alpha_1| + |\alpha_2| + C_1 + C_2 + u(C_2 + |\alpha_2|))\big)\,|y| \\
&\quad = (C_1 + C_2 + u\,(|\alpha_1| + 2|\alpha_2| + C_1 + 2C_2) + \\
&\qquad u^2(C_2 + |\alpha_2|)\big)\ |y| \quad .
\end{aligned}
$$

□

These results are not difficult, but they are precisely tailored for their use in the next sections for bounding the local round-off errors of the numerical methods.

4.2 Euler's method

4.2.1 Description of the method

As a first example, we focus on the Runge-Kutta method of order 1, well-known as Euler's method. It is the simplest case of Runge-Kutta methods for solving ODEs. The method error (*i.e.*, the difference between the exact solution and the discretized one) is larger than the errors made with higher-order methods. In contrast, the implementation of the Euler's method involves less FP operations at each iteration. Thus, the round-off errors should be easier to bound. Although it is a simple scheme, the precise analysis of round-off errors is not trivial.

In the linear unidimensional case, the stability function of the method is $R_{\text{Euler}}(h, \lambda) = 1 + h\lambda$, see Example 1, and the method can be described by the following expression

$$
\begin{cases}
y_0 = y(0) \\
\forall n, \quad y_{n+1} = y_n + h\lambda y_n \quad .
\end{cases}
\tag{13}
$$

Round-off errors can occur because of various factors, and accumulate at each iteration. The algorithm corresponding to the Euler's method is

$$
\begin{cases}
\widetilde{y_0} = y_0 + \varepsilon_0 \\
\forall n, \quad \widetilde{y_{n+1}} = \widetilde{y_n} \oplus \widetilde{h} \otimes \widetilde{\lambda} \otimes \widetilde{y_n} \quad .
\end{cases}
\tag{14}
$$

Under reasonable hypotheses of stability, see Figure 2, one has $|R_{\text{Euler}}(h, \lambda)| = |1 + h\lambda| \leqslant 1$, and thus $-2 \leqslant h\lambda \leqslant 0$.

4.2.2 Bound on local round-off errors of Euler's method

Lemma 6. 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:* $\varepsilon_{n+1} \leqslant 11.01u\ |\widetilde{y_n}|$.

Proof. This proof relies on an application of Lemma 5 with $y = \widetilde{y_n}$. Then $X_1 = \widetilde{y_n}$, $\alpha_1 = 1$ and $C_1 = 0$. And $X_2 = \widetilde{h} \otimes \widetilde{\lambda}$, $\alpha_2 = h\lambda$. We then need a correct C_2.

Gappa is a tool to formally verify properties on numerical programs using FP arithmetic [9, 10]. It bounds FP errors and values, and produces a proof term which could be checked by proof assistants such as Coq. Using Gappa, we obtain $|X_2 - h\lambda| \leqslant 6 \times 2^{-53}$ from intervals on $h\lambda$ and on h, so we choose $C_2 = 6u$.

The application of Lemma 5 then gives

$$
\varepsilon_{n+1} \leqslant |\widetilde{y_n}|\,\big(6u + u\,(1 + 2|h\lambda| + 12u) + u^2(6u + |h\lambda|)\big) \quad .
$$

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

$$
\varepsilon_{n+1} \leq |\widetilde{y_n}|\,\big(11u + 14u^2 + 6u^3\big) \leqslant 11.01u\,|\widetilde{y_n}| \quad .
$$

□

Remark that if we assume that both h and λ do not have any round-off error (they may be powers of two), then this round-off error reduces to $6.01u$.

4.2.3 Bound on global error of Euler's method

Now we can give a bound on the global round-off error of the Euler's method instantiating Theorem 3 with $C = 11.01u$.

Theorem 7. 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* $\forall n$,

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

The bound on the error only depends on the magnitude of y_0. Indeed, the influence of the following discrete values of the numerical solution is determined by the algorithm. Thus, our error analysis is valid for various scientific fields using Euler's method. For instance, the magnitude of the computed solutions are really larger in astrophysics [5] (from 10^3 to 10^{14}) than in chemistry [17] (from 10^{-12}).

4.3 Higher-order Runge-Kutta methods

4.3.1 Description of classical Runge-Kutta methods

Here we give a description of two classical higher-order Runge-Kutta methods: the midpoint method RK2 and the well-known RK4 method.

The RK2 method is described by the linear stability function $R_{\text{RK2}}(h,\lambda) = 1 + h\lambda + 0.5(h\lambda)^2$. The RK4 method is described by the Butcher tableau of Figure 1(a). Even if one can use a more efficient method, the RK4 scheme is not too difficult to implement and is frequently used for industrial applications. The method error is better than with Euler and RK2 methods: the global one is $\mathcal{O}(h^4)$. However, it involves a very large number of FP operations at each step, and the round-off error made using this scheme becomes less satisfying, as shown in Section 4.3.2.

4.3.2 Bound on local round-off errors of RK methods

Lemma 8. Bound on Runge-Kutta 2 local round-off errors. *Let us assume the stability, and that* $-2 \leqslant h\lambda \leqslant -2^{-100}$ *and* $2^{-60} \leqslant h \leqslant 1$. *Then for all* n, $\varepsilon_{n+1} \leqslant 28.01u\,|\widetilde{y_n}|$.

Proof. As explained, we have for the RK2 method

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

Stability means that $|1 + h\lambda + \frac{h^2\lambda^2}{2}| \leqslant 1$. We want to bound the corresponding ε_n. As in Euler's method, we rely on Lemma 5. We put in Figure 3 the corresponding α and C of each FP term.

The first 3 lines are similar to the local error of Euler's method of Section 4.2.2 (as the hypotheses on $h\lambda$ are the same).

Then Gappa gives us the bound of Line 4

$$\left|\circ\left[\widetilde{h}\widetilde{h}\frac{1}{2}\widetilde{\lambda}\widetilde{\lambda}\right] - \frac{h^2\lambda^2}{2}\right| \leqslant 2^{-49} = 16u$$

under the given hypotheses on $h\lambda$ and h.

FPterm	α	C
$\widetilde{y_n}$	1	0
$\widetilde{h} \otimes \widetilde{\lambda}$	$h\lambda$	$6u$
$\circ\left[\widetilde{y_n} + \widetilde{h}\widetilde{\lambda}\widetilde{y_n}\right]$	$1 + h\lambda$	$11u + 15u^2$
$\circ\left[\widetilde{h}\widetilde{h}\frac{1}{2}\widetilde{\lambda}\widetilde{\lambda}\right]$	$\frac{h^2\lambda^2}{2}$	$16u$
$\circ\left[\widetilde{y_n} + \widetilde{h}\widetilde{\lambda}\widetilde{y_n} + \widetilde{h}\widetilde{h}\frac{1}{2}\widetilde{\lambda}\widetilde{\lambda}\widetilde{y_n}\right]$	$1 + h\lambda + \frac{h^2\lambda^2}{2}$	$28.01u$

Figure 3: Steps for the local round-off error bound of RK2.

The last line corresponds to using Lemma 5 with the C and α of Lines 3 and 4, while bounding the u^2 and u^3 terms. □

Lemma 9. Bound on Runge-Kutta 4 local round-off errors. *Let us assume the stability, and that* $-3 \leqslant h\lambda \leqslant -2^{-100}$ *and* $2^{-60} \leqslant h \leqslant 1$. *Then for all* n, $\varepsilon_{n+1} \leqslant 194u\,|\widetilde{y_n}|$.

Proof. As explained, the RK4 method is defined by Equation (8). In this case, stability means that $|1 + h\lambda + \frac{h^2\lambda^2}{2} + \frac{h^3\lambda^3}{12} + \frac{h^4\lambda^4}{24}| \leqslant 1$ which leads to $-3 \leqslant h\lambda \leqslant 0$. We want to bound the corresponding ε_n. As previously, we rely on Lemma 5 and on Gappa under the given hypotheses on $h\lambda$ and h. We put in Figure 4 the corresponding α and C of some FP terms. □

FPterm	C	
$\widetilde{y_n}$	0	
$\circ\left[\widetilde{h}\widetilde{\frac{1}{6}}\widetilde{\lambda}\right]$	$2u$	$(\star)$
$\circ\left[\widetilde{y_n} + \widetilde{h}\widetilde{\frac{1}{6}}\widetilde{\lambda}\right]$	$4u$	
$\circ\left[\widetilde{h}\widetilde{\frac{1}{3}}\widetilde{\lambda}\right]$	$4u$	$(\star)$
$\circ\left[\widetilde{h}\widetilde{h}\widetilde{\frac{1}{6}}\widetilde{\lambda}\widetilde{\lambda}\right]$	$12u$	$(\star)$
$\circ\left[\widetilde{h}\widetilde{h}\widetilde{h}\widetilde{\frac{1}{12}}\widetilde{\lambda}\widetilde{\lambda}\widetilde{\lambda}\right]$	$28u$	$(\star)$
$\circ\left[\widetilde{h}\widetilde{h}\widetilde{h}\widetilde{h}\widetilde{\frac{1}{24}}\widetilde{\lambda}\widetilde{\lambda}\widetilde{\lambda}\widetilde{\lambda}\right]$	$53u$	$(\star)$
$\dots$	$\dots$	
$\circ\left[\widetilde{y_n} + \dots + \widetilde{h}\widetilde{h}\widetilde{h}\widetilde{h}\widetilde{\frac{1}{24}}\widetilde{\lambda}\widetilde{\lambda}\widetilde{\lambda}\widetilde{\lambda}\widetilde{y_n}\right]$	$194u$	

Figure 4: Steps for the local round-off error bound of RK4. The bounds provided by Gappa are marked with $(\star)$.

4.3.3 Bound on global round-off errors of RK methods

The reasoning is the same for higher-order methods as for Euler's method. We instantiate the general Theorem 3 with the constants provided by Lemma 8 and Lemma 9.

Theorem 10. Bound on absolute error of RK2 method. *Let us assume the stability, and that* $-2 \leqslant h\lambda \leqslant -2^{-100}$ *and* $2^{-60} \leqslant h \leqslant 1$. *Then for all* n,

$$|E_n| \leqslant (28.01u + |R(h,\lambda)|)^n \left(\varepsilon_0 + \frac{n \cdot 28.01u|y_0|}{28.01u + |R(h,\lambda)|}\right) \quad .$$

Theorem 11. Bound on absolute error of RK4 method. *Let us assume the stability, and that* $-3 \leqslant h\lambda \leqslant -2^{-100}$ *and* $2^{-60} \leqslant h \leqslant 1$. *Then for all* n,

$$|E_n| \leqslant (194u + |R(h,\lambda)|)^n \left(\varepsilon_0 + \frac{n \cdot 194u|y_0|}{194u + |R(h,\lambda)|}\right) \quad .$$

5 Methodology

From the previous theorems and applications, we may exhibit a generic methodology for bounding the round-off error of an explicit single-step and fixed-step size method belonging to the class of Runge-Kutta methods when the ODE is linear. The algorithm corresponding to this method may be stated as

$$y_{n+1} = \sum_{i=0}^{M} \alpha_i y_n \quad ,$$

with α_i depending both on the ODE and the step-size, see Equation (8) for an example of such algorithm. Preceding examples of those α_i are $h\lambda$, $\frac{h^2\lambda^2}{6}$, and $\frac{h^4\lambda^4}{24}$. The FP program is then $\oplus_{i=0}^{M} \widetilde{\alpha_i} \otimes \widetilde{y_n}$ and we want to bound its round-off error compared to an ideal execution on real numbers.

The methodology may then be stated as

- compute an error bound on the initial value y_0. This depends whether it is a constant, such as one, or the result of a computation such as an exponential.
- compute an absolute error bound on each of the α_i. This has been done here using the Gappa tool, and under the stability hypotheses for better results. But any method and any additional knowledge (such as h is correct) may be used to get these error bounds.

- compute a bound on the local round-off error by mechanically applying M times Lemma 5. This requires bounding terms such as $|\sum_{i=0}^{k} \alpha_i|$ with the stability hypothesis. Note that higher-order terms (proportional to u^2 or u^3) appear. They have to be taken care of, either by keeping them or by slightly increasing the term proportional to u, as done here.
- relying on Theorem 3, compute a bound on the global round-off error.

This methodology has been applied to the Euler's method in Section 4.2 and to RK2 and RK4 in Section 4.3. This could have been similarly applied to higher-order explicit one-step Runge-Kutta methods.

6 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. For that, we have provided a generic methodology that relies on the stability property of the methods.

Nevertheless, we have made several assumptions. The first one is the fact we assumed there is neither underflow, nor overflow. Overflow is easy to take into account: given the particular form of these algorithms (with only additions and multiplications), if an overflow occurs, then the final result will be either NaN or an infinity. There is therefore an easy check *a posteriori* for overflow. Underflow is more difficult to detect. As far as additions are concerned, there is no problem as underflowing additions are correct. But multiplications may cause huge relative errors and Lemma 5 does not hold anymore. We are convinced that, in physical problems we study in practice, this does not happen. Of course, we are planning to work on more concrete hypotheses on the Butcher tableau and the input in order to guarantee that no underflow occur at any time in the algorithm.

An interesting remark appears when looking precisely at the local round-off errors of the studied methods. A preponderant term is that of the higher-degree computation, for instance $h^4\lambda^4/24$ for RK4. And this value is computed at each iteration and has always the same (large) round-off error. It is therefore unlikely that this error will be compensated each time. The compensation therefore seems less plausible than what mathematicians usually expect, maybe meaning a larger error bound.

A natural perspective is to enlarge the class of methods we consider. This includes variable-step methods, where the h is a function of n. The difficulty lies in the choice of the value h_n, that depends on the dynamics of the system. Here, we only consider unidimensional problems, but we are also interested in multi-dimensional problems, where the scalar λ is replaced by a matrix. The corresponding stability conditions would then depend on the eigenvalues of the matrix.

A second natural perspective is to study round-off error in other numerical integration methods, *e.g.*, the multi-step methods belonging into Adams-Bashworth family. Other kind of methods belongs to implicit numerical integration methods which usually require the solution of fixed-point equations and so involve more complex numerical algorithms which may produce more round-off errors.

A last perspective on round-off error analysis for numerical integration methods should consider non-linear ordinary differential equations. 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.

Another perspective is to get interested into filters [20]. The domain may seem quite distant, but the problems to solve are quite similar (iterative numerical methods) and a closer look at both methodologies would probably benefit both sides.

The pen-and-paper proofs displayed here are quite complex to check and to be convinced of. A solution would be to formally prove the theorems presented here in order to increase the trust in those results and probably help to find the values. Moreover, the methodology of Section 5 seems quite systematic, therefore an automation may even be possible. 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 library for FP arithmetic [6] and mixes well with Gappa while Isabelle/HOL has a library about ODEs [15].

Finally, we want to consider at the same time the method error and round-off error. A fine-grained study of the total error would permit to decide whether a scheme is better than another for a given class of problems. Mathematicians only consider the order of a scheme to choose it, and a Runge-Kutta scheme of order p gives a global method error of $\mathcal{O}(h^p)$. For example, a RK4 scheme is has a much smaller method error than a Euler's method. However, it makes more computations and therefore involves a larger round-off error, as shown is this paper. Deciding which one to use, depending for example on the maximum time, would be quite valuable in practice.

Acknowledgment

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 and by FastRelax ANR-14-CE25-0018-01.

References

[1] Julien Alexandre dit Sandretto and Alexandre Chapoutot. Validated Explicit and Implicit Runge-Kutta Methods. *Reliable Computing*, 22, 2016.

[2] Julien Alexandre dit Sandretto and Alexandre Chapoutot. Validated Simulation of Differential Algebraic Equations with 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] 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.

[5] 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.

[6] 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.

[7] Olivier Bouissou and Matthieu Martel. GRKLib: a guaranteed Runge-Kutta library. In *International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics*. IEEE, 2006.

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

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

[10] 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.

[11] Walter Gautschi. *Numerical Analysis: An Introduction*. Birkhauser, 1997.

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

[13] Ernst Hairer, Robert I McLachlan, and Alain Razakarivony. Achieving brouwer's law with implicit Runge–Kutta methods. *BIT Numerical Mathematics*, 48(2):231–243, 2008.

[14] Peter Henrici. *Error propagation for difference methods*. The SIAM series in applied mathematics. John Wiley, 1963.

[15] Fabian Immler. Formally verified computation of enclosures of solutions of ordinary differential equations. In *NASA Formal Methods*, pages 113–127. Springer, 2014.

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

[17] K.B. Lipkowitz and D.B. Boyd. *Reviews in Computational Chemistry*. Number vol. 10 in Reviews in Computational Chemistry. Wiley, 2009.

[18] Markus Neher, Kenneth R Jackson, and Nedialko S Nedialkov. On Taylor model based integration of ODEs. *SIAM Journal on Numerical Analysis*, 45(1):236–262, 2007.

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

[20] Anastasia Volkova, Thibault Hilaire, and Christoph Q. Lauter. Determining Fixed-Point Formats for a Digital Filter Implementation using the Worst-Case Peak-Gain measure. In *Asilomar Conference on Signals, Systems and Computers*, November 2015.