Formalisation en Coq des erreurs d'arrondi de méthodes de Runge-Kutta pour les systèmes matriciels

Florian Faissole

Formalisation en Coq des erreurs d'arrondi de méthodes de Runge-Kutta pour les systèmes matriciels

Florian Faissole

Inria, Université Paris-Saclay, F-91120 Palaiseau
LRI, CNRS & Univ. Paris-Sud, F-91405 Orsay
Email : florian.faissole@inria.fr

Résumé

Bien que les équations différentielles ordinaires soient omniprésentes dans la modélisation de systèmes physiques ou biologiques, leur résolution exacte est parfois fastidieuse voire impossible. L'utilisation de méthodes numériques, comme les méthodes de Runge-Kutta, permet d'obtenir des solutions approchées. Nous exhibons et formalisons en Coq des bornes sur les erreurs d'arrondi induites par l'implémentation en arithmétique à virgule flottante de méthodes de Runge-Kutta appliquées à des systèmes linéaires matriciels en tenant compte d'éventuels dépassements de capacité inférieurs.

1 Introduction

Les équations différentielles ordinaires modélisent de nombreux phénomènes physiques, biologiques ou économiques. Il n'existe cependant pas toujours de méthode directe pour les résoudre. Des méthodes numériques itératives ont été mises au point pour résoudre ces problèmes de façon approchée. L'essentiel des travaux scientifiques autour des méthodes numériques consiste à construire des méthodes relativement peu coûteuses permettant d'obtenir une approximation précise de la solution exacte et qui puissent s'appliquer à une classe importante de problèmes. Parmi ces méthodes de résolution, les méthodes de Runge-Kutta sont parmi les plus fréquemment utilisées (voir section 3). Les domaines d'application des équations différentielles étant parfois critiques (*e.g.* en aéronautique ou pour la modélisation de comportements robotiques), leur résolution est un objet d'étude intéressant pour des travaux de vérification formelle [13, 14, 5].

L'implémentation de ces méthodes en arithmétique à virgule flottante provoque des erreurs d'arrondi pouvant s'accumuler au cours des itérations. Ces erreurs étant dans la plupart des cas négligeables face aux erreurs de méthode, elles ont peu été étudiées par les numériciens. Des travaux ont néanmoins proposé des méthodes pour majorer les erreurs d'arrondi par des approches probabilistes tirant parti d'éventuelles compensations d'erreurs [10] ou en utilisant des techniques à base d'intervalles [8, 1] ou de modèles de Taylor [3, 17]. Cependant, les bornes exhibées ne tiennent compte ni des caractéristiques mathématiques détaillées des méthodes de Runge-Kutta, ni de la forme de l'implémentation choisie. Nous proposons une approche à grains fins (*i.e.* en décomposant l'analyse d'erreur au niveau des opérations élémentaires) qui tire parti de ces caractéristiques, ce qui rapproche nos travaux des résultats présentés par Fousse [9] pour les méthodes d'intégration numérique

ou par Boldo [4] qui borne et formalise en Coq les erreurs d'arrondi induites par un schéma de résolution de l'équation des ondes. Roux [18] propose une analyse d'erreur d'arrondi en Coq pour plusieurs algorithmes numériques impliquant des opérations matricielles, comme la décomposition de Cholesky. Cette formalisation repose sur une spécification de l'arithmétique à virgule flottante définie *via* un type `Record` et particulièrement adaptée à l'analyse d'erreur. Cette spécification est par ailleurs satisfaite par le format *binary64* de Flocq [7], une bibliothèque Coq d'arithmétique des ordinateurs.

Boldo, Faissole et Chapoutot [6] ont exhibé (sans garantie formelle) des bornes sur les erreurs d'arrondi induites par l'implémentation de méthodes de Runge-Kutta appliquées à des systèmes linéaires unidimensionnels (systèmes de la forme $\dot{y} = \lambda y$ où $\lambda \in \mathbb{R}$). Dans cet article, nous généralisons cette approche au cas des systèmes linéaires multidimensionnels, où λ est remplacé par une matrice carrée A à coefficients réels. Les opérations matricielles étant plus complexes que les opérations sur les nombres réels, les résultats prennent en compte un nombre plus important de paramètres, à commencer par la dimension du système. Nous proposons de plus une formalisation de ces résultats dans l'assistant de preuves Coq [1] en combinant la bibliothèque Flocq [7] et les matrices de la bibliothèque MathComp [15]. Nous tenons compte d'éventuels dépassements graduels de capacité inférieurs (*underflows*), dont la contribution impacte les bornes d'erreurs exhibées. En revanche, nous ne tenons pas compte des dépassements de capacité supérieurs (*overflows*). Notre approche diffère légèrement de celle adoptée par Roux [18] car nous utilisons directement la formalisation des formats de nombres flottants de la bibliothèque Flocq, sans passer par une spécification dédiée de l'arithmétique à virgule flottante.

La section 2 présente quelques rappels d'arithmétique à virgule flottante. La section 3 est dédiée à la présentation du problème et de notre méthodologie (basée sur la distinction entre erreurs locales et globales). Dans la section 4, nous présentons les éléments de base de la formalisation Coq. Dans les sections 5 et 6, nous bornons respectivement les erreurs locales et globales des méthodes de Runge-Kutta avant de conclure en section 7.

2 Prérequis d'arithmétique à virgule flottante

2.1 Formats de représentation des nombres flottants

Les formats de représentation des nombres à virgule flottante ainsi que les opérations sur ces nombres sont régis par la norme IEEE-754 [12, §3]. En analyse d'erreur, une définition relativement haut niveau (sans décrire la représentation bit à bit des nombres flottants) est suffisante. Un format flottant $\mathbb{F}$ est représenté par un tuple $(\beta, p, e_{\min}, e_{\max})$ où l'entier $\beta \geqslant 2$ est la base, $p \in \mathbb{N}$ est la précision, $e_{\min} \in \mathbb{Z}$ et $e_{\max} \in \mathbb{Z}$ sont les valeurs minimales et maximales de l'exposant. Un nombre flottant dans $\mathbb{F}$ est soit une valeur exceptionnelle parmi $+\infty$, $-\infty$ ou NaN, soit une valeur égale à $\pm d_0.d_1 \cdots d_{p-1} \times \beta^e$ (avec d_i des chiffres dans la base β) et tel que $e_{\min} \leq e \leq e_{\max}$. Dans ce qui suit, $\beta = 2$.

La Figure 1 présente la répartition des nombres flottants sur l'axe réel. Les nombres flottants dits normalisés vérifient $d_0 = 1$ et $e_{\min} \leq e \leq e_{\max}$. Le plus petit nombre flottant normalisé positif est $\xi = 2^{e_{\min}}$ et le plus grand nombre flottant normalisé est $\Omega = (2 - 2^{1-p})2^{e_{\max}}$. Lorsqu'un nombre flottant est plus petit que ξ, il est dit dénormalisé et on parle de dépassement de capacité inférieur (*underflow*). Nous avons alors $e = e_{\min}$ et $d_0 = 0$. On note $\eta = 2^{e_{\min}-p+1}$ le plus petit nombre flottant dénormalisé positif. Lorsque le résultat

1. Les fichiers sont disponibles en ligne : `https://www.lri.fr/~faissole/CoqRK`

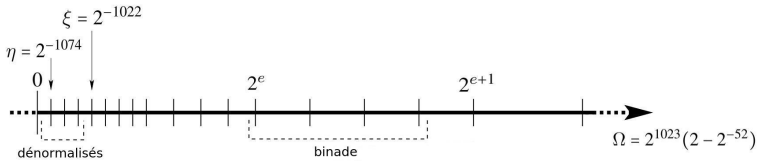

FIGURE 1 – Représentation des nombres flottants (format *binary64*) sur l'axe réel

d'une opération flottante n'est pas exactement représentable dans le format considéré, il est nécessaire d'arrondir le résultat vers un nombre flottant. Plusieurs modes d'arrondi sont définis dans la norme IEEE-754 [12], dont l'arrondi au plus proche. Lorsque le résultat est le point-milieu entre deux flottants consécutifs, une régle de bris d'égalité (*tie-breaking rule*) permet de choisir vers lequel arrondir, par défaut le nombre flottant pair est choisi.

Dans cet article, nous considérons un format flottant $\mathbb{F}$ prenant en compte les *underflows* (sauf mention du contraire). Nous considérons un mode d'arrondi au plus proche avec une règle de bris d'égalité quelconque, noté $\circ$. Nous noterons $\oplus$, $\ominus$, $\otimes$ et $\oslash$ les opérations arrondies correspondant aux opérations usuelles $+, -, \times$ et $/$, $\circ[\cdots]$ signifie que toutes les opérations à l'intérieur des crochets sont arrondies. Pour une même opération, le parenthésage à droite est implicite en l'absence de parenthèses, ainsi $\circ[ab + c] = (a \otimes b) \oplus c$ et $\circ[a + b + c] = a \oplus (b \oplus c)$. La convention s'étend aux opérations entre matrices et vecteurs. En particulier, pour $v \in \mathbb{R}^n$ un vecteur de taille n et $A, B \in \mathbb{R}^{n\times n}$ des matrices carrées de taille n, $\circ[A \times v]$ et $\circ[A \times B]$ font respectivement référence au produit de A par v et au produit de A par B en évaluant les sommations de la droite vers la gauche.

2.2 Fondements de l'analyse d'erreurs d'arrondi

Les analyses d'erreur que nous présentons sont basées sur quelques définitions et résultats fondamentaux (voir Higham pour une présentation détaillée de ces résultats [11, §3]). Nous commençons par présenter un résultat bornant l'erreur relative commise lors de l'arrondi d'un nombre réel de valeur absolue supérieure à ξ : si $x \in \mathbb{R}$ est tel que $\xi \leqslant |x|$, alors $\frac{|\circ(x)-x|}{|x|} \leqslant u = \frac{1}{2}\beta^{1-p}$. La quantité u, fondamentale en analyse d'erreur, est appelée unité d'arrondi. Par exemple, dans le format double précision *binary64*, $u = 2^{-53}$. Lorsque le nombre réel x est plus petit que ξ, l'erreur relative commise en arrondissant x peut devenir très grande, mais nous pouvons prouver que l'erreur absolue est bornée par $\frac{\eta}{2}$. Nous pouvons en déduire ce que Higham appelle modèle standard de l'arithmétique à virgule flottante [11, §2.2.], *i.e.* pour $x, y \in \mathbb{F}$, $\Diamond \in \{+, -, \times, /\}$, il existe $\delta, \varepsilon \in \mathbb{R}$ tels que $\circ(x \Diamond y) = (x \Diamond y)(1 + \varepsilon) + \delta$, $|\varepsilon| \leqslant u$, $|\delta| \leqslant \frac{\eta}{2}$ et $\varepsilon\delta = 0$. Si $\Diamond \in \{+, -\}, \delta = 0$.

Nous utiliserons régulièrement la notation $\gamma_d = \frac{du}{1-du}$ ($d \in \mathbb{N}$, $du < 1$) [11, §3.1.].

2.3 Flocq : bibliothèque d'arithmétique des ordinateurs en Coq

Pour formaliser nos résultats, nous utilisons Flocq, une bibliothèque Coq d'arithmétique des ordinateurs développée par Boldo et Melquiond [7] comportant une représentation abstraite des nombres flottants : les nombres dans un format donné sont des sous-ensembles des nombres réels `R` de la bibliothèque standard de la forme $m \cdot \beta^e$ où m et e sont des entiers. Par exemple, le format `FLX` correspond aux nombres flottants sans bornes sur les exposants : seule la précision p est prise en compte, avec la condition $|m| < \beta^p$. Le format `FLT` prend en compte les dépassements graduels de capacité inférieurs et est paramétré par

p et $e_{\min}$, les nombres flottants dans ce format vérifiant $|m| < \beta^p$ et $e \geq e_{\min}$. L'ensemble des résultats présentés dans cet article sont valides pour le format `FLX`. Nous avons ensuite étendu l'ensemble des résultats génériques pour le format `FLT`, *i.e.* l'ensemble des résultats à l'exception de l'instanciation aux méthodes d'Euler et de RK2.

3 Présentation du problème et méthodologie

Les méthodes de Runge-Kutta sont itératives. Elles consistent à discrétiser un intervalle de temps $[0, t_N]$ en un ensemble de points $t_0, \ldots, t_N$ et à construire itérativement les solutions approchées $y_0, \ldots, y_N$ en ces points. Ce sont des méthodes explicites à un pas constant h : la solution au temps $t_{n+1} = t_n + h$ est obtenue à partir de la solution au temps t_n.

Les systèmes linéaires multidimensionnels sont parmi les plus fréquemment rencontrés dans les domaines liés à des problèmes physiques. Les équations différentielles associées sont de la forme $\dot{y} = Ay \quad \text{avec} \quad y \in \mathbb{R}^d \quad \text{et} \quad A \in \mathbb{R}^{d\times d}$. L'application d'une méthode de Runge-Kutta explicite sur ce système linéaire conduit à une relation de récurrence de la forme $y_{n+1} = R(hA)y_n$ avec R un polynôme en hA (de la forme $\sum_i \alpha_i (hA)^i$, $\alpha_i \in \mathbb{R}$). Prenons l'exemple de deux méthodes classiques, les méthodes d'Euler et de RK2. Les polynômes associés à ces méthodes sont $R_{\text{Euler}}(hA) = (I + hA)$ et $R_{\text{RK2}}(hA) = (I + hA + 0{,}5h^2A^2)$. Le polynôme R est appelé fonction de stabilité linéaire pour les méthodes de Runge-Kutta car, dans le cas des systèmes linéaires, la condition $|||R(hA)|||_\infty < 1$ caractérise une classe de couples système-méthode dit stables (voir section 4.1 pour une définition de la norme vectorielle $\|\cdot\|_\infty$ et de sa norme matricielle subordonnée $|||\cdot|||_\infty$). La fonction R est purement mathématique et ne caractérise pas l'implémentation des méthodes en arithmétique à virgule flottante, qui suppose le choix d'un algorithme, noté $\widetilde{R}$ et qui dépend de trois paramètres : le pas h (considéré comme représentable dans le format flottant, contrairement à ce qui est fait dans [6]), une matrice $\widetilde{A}$ correspondant à la matrice "arrondie de A" (matrice obtenue après arrondi de chaque coefficient de A) ainsi que la solution $\widetilde{y_n}$ calculée à l'itération précédente (lors du calcul de $\widetilde{y_{n+1}}$). Ainsi, une implémentation de la méthode de Runge-Kutta est de la forme : $\widetilde{y_0} = \circ(y_0)$ et pour tout n, $\widetilde{y_{n+1}} = \widetilde{R}(h, \widetilde{A}, \widetilde{y_n})$. À un même polynôme R correspondent plusieurs implémentations $\widetilde{R}$ possibles [2]. Dans cet article, nous nous intéressons à une évaluation "à la Horner" des méthodes numériques (décrite ci-dessous pour Euler et RK2 respectivement) :

$$\widetilde{y_{n+1}} = \circ\left[\widetilde{y_n} + (h\widetilde{A})\widetilde{y_n}\right], \qquad \widetilde{y_{n+1}} = \circ\left[\widetilde{y_n} + (h\widetilde{A})\left(\widetilde{y_n} + (\tfrac{h}{2}\widetilde{A})\widetilde{y_n}\right)\right].$$

Nous souhaitons borner l'erreur d'arrondi globale induite par l'implémentation d'une méthode de Runge-Kutta après n itérations, *i.e.* $E_n = \|\widetilde{y_n} - y_n\|_\infty$. Afin d'y parvenir, nous commençons par étudier les erreurs dites locales. L'erreur locale commise à l'itération n ne quantifie que les erreurs commises par les calculs de l'itération courante et peut être définie par $\varepsilon_0 = \|\widetilde{y_0} - y_0\|_\infty$ (que nous supposons connue) et pour tout $n \in \mathbb{N}, \quad \varepsilon_{n+1} = \|\widetilde{R}(h, \widetilde{A}, \widetilde{y_n}) - R(hA)\widetilde{y_n}\|_\infty$. Si les résultats ont du être adaptés, la méthodologie est similaire à celle utilisée dans le cas unidimensionnel [6]. Elle consiste dans un premier temps à construire une borne sur les erreurs locales relatives par applications mécaniques et consécutives du Lemme 1 (voir section 5). Ensuite, cette borne sur les erreurs locales est utilisée pour borner l'erreur globale par application directe du Théorème 2 (voir section 6).

2. En effet, les opérations flottantes ne sont pas associatives et dépendent de l'ordre d'évaluation. Par ailleurs, les calculs peuvent être décomposés, factorisés, etc.

4 Éléments de base de la formalisation

4.1 Vecteurs et matrices

Les systèmes étudiés font intervenir des vecteurs et des matrices à coefficients réels. Nous utilisons essentiellement des vecteurs colonnes de taille d (dont le type Coq est noté `cV_d`) et des matrices carrées de taille d (de type `M_d`).

Nous procédons à une analyse par normes des erreurs d'arrondi, ce type d'analyse reposant sur des normes sous-multiplicatives [11, §6]. Les normes vectorielles $\|\cdot\|_1$ et $\|\cdot\|_\infty$ sont particulièrement adaptées à l'analyse d'erreurs et les normes matricielles subordonnées à ces normes sont faciles à exprimer en fonction des coefficients de la matrice. Par ailleurs, nous nous basons sur des résultats de Higham qui sont valables pour ces deux normes [11]. Nous travaillons ici avec la norme $\|\cdot\|_\infty$, définie comme $\|v\|_\infty = \max_i |v_i|$ pour $v \in \mathbb{R}^d$ et dont la norme subordonnée est définie par $|||A|||_\infty = \max_i \sum_j |A_{i,j}|$ pour $A \in \mathbb{R}^{d\times d}$. Ces normes sont formalisées à l'aide de grands opérateurs (*bigops*) de MathComp [2], un mécanisme permettant d'itérer des opérations comme l'addition ou le maximum :

```
Definition vec_norm {d} : cV_d → R := fun v ⇒ \big[Rmax / 0]_(i < d) Rabs (v i).
Definition mat_norm {d} : M_d → R :=
                fun A ⇒ \big[Rmax / 0]_(i < d) (\big[Rplus / 0]_(j < d) Rabs (A i j)).
```

Nous prouvons qu'il s'agit bien de normes et que $|||\cdot|||_\infty$ est sous-multiplicative, *i.e.* pour $v \in \mathbb{R}^d, A \in \mathbb{R}^{d\times d}$, $\|Av\|_\infty \leqslant |||A|||_\infty \|v\|_\infty$ (`mx_vec_norm_submult` en Coq) et pour $A, B \in \mathbb{R}^{d\times d}$, $|||AB|||_\infty \leqslant |||A|||_\infty |||B|||_\infty$ (`mx_norm_submult` en Coq). La sous-multiplicativité permet de démontrer des bornes sur les erreurs d'arrondi des produits matrice-vecteur (`mx_vec_prod_error`) et matrice-matrice (`mx_prod_error`) :

$$\|\circ[A\times v] - Av\|_\infty \leqslant \gamma_d |||A|||_\infty \|v\|_\infty + \frac{d}{2}(1+\gamma_{d-1})\eta.$$
$$|||\circ[A\times B] - AB|||_\infty \leqslant \gamma_d |||A|||_\infty |||B|||_\infty + \frac{d^2}{2}(1+\gamma_{d-1})\eta.$$

Nous prouvons ces propriétés pour des sommations évaluées de droite à gauche. Ces bornes restent cependant vraies quelque soit l'ordre d'évaluation [11, §3.5.].

4.2 Méthodes de Runge-Kutta

Nous définissons un type `Sc` (pour Schéma), dont les éléments définissent des relations de récurrence entre les vecteurs y_n et y_{n+1} et caractérisent de ce fait l'application des méthodes de Runge-Kutta à des systèmes multidimensionnels (de dimension d) :

```
Definition Sc (d : nat) : Type := (R → R) → cV_d → cV_d.
```

Sur la donnée d'une fonction $\mathcal{W} \in \mathbb{R} \to \mathbb{R}$ (par exemple, il peut s'agir d'un arrondi ou de la fonction identité) et d'un vecteur $y_n \in \mathbb{R}^d$, un élément de type `Sc` renvoie un vecteur $y_{n+1} \in \mathbb{R}^d$. On peut évaluer l'application de la méthode numérique `M : Sc` sur n itérations à partir de la condition initiale `y0` (`y0_tilde` étant le vecteur `y0` dont toutes les composantes ont été "arrondies" par la fonction $\mathcal{W}$) :

```
Definition meth_iter (M:Sc) n (y0 : cV_d) (W : R → R) := iter n (M W) y0_tilde
```

Nous définissons formellement les erreurs locales et globales (en valeur absolue) :

```
Definition error_loc (M : Sc) n (y0 : cV_d) (W : R → R) (*ε_{n+1} = ‖R̃(h,Ã,ỹ_n) − R(hA)ỹ_n‖_∞*)
  := vec_norm (M W (meth_iter M n y0 W) − M (fun x ⇒ x) (meth_iter M n y0 W)).
Definition error_glob (M : Sc) n (y0 : cV_d) (W : R → R)          (*E_n = ‖ỹ_n − y_n‖_∞*)
  := vec_norm (meth_iter M n y0 W − meth_iter M n y0 (fun x ⇒ x)).
```

5 Erreurs locales

Nous démontrons un résultat générique pour borner l'erreur d'arrondi locale commise lors d'une itération de la méthode. Ce résultat permet de traiter toute méthode à un pas explicite, d'ordre quelconque, implémentée suivant une évaluation "à la Horner" (voir Section 3)

Lemme 1. *Soit* $d \in \mathbb{N}^*$, $y \in \mathbb{R}^n$, $C_1, C_2, C_3, D_1, D_3 \in \mathbb{R}_+$, $A_1, A_2, A_3 \in \mathbb{R}^{n\times n}$, $X_1, X_3 \in \mathbb{F}^d$, $\widetilde{A_2} \in \mathbb{F}^{n\times n}$. *Soit* $\rho = C_2 u|||A_3|||_\infty + C_3 u|||A_2|||_\infty + C_2C_3u^2$.
Soit $\mathscr{C} = C_1u + \rho + u\left(|||A_1|||_\infty + C_1u\right) + (u + (1+u)\gamma_d)(\rho + |||A_2|||_\infty|||A_3|||_\infty)$.
Soit $\mathscr{D} = (1+u)\left(\frac{d}{2}(1+\gamma_{d-1}) + D_1 + D_3(1+\gamma_d)(C_2 + |||A_2|||_\infty)\right)$.
Supposons que :
- $\|X_1 - A_1 y\|_\infty \leqslant C_1 u\|y\|_\infty + D_1\eta$.
- $|||\widetilde{A_2} - A_2|||_\infty \leqslant C_2 u$.
- $\|X_3 - A_3 y\|_\infty \leqslant C_3 u\|y\|_\infty + D_3\eta$.

Alors $\|X_1 \oplus (\widetilde{A_2} \otimes X_3) - (A_1 + A_2A_3)y\|_\infty \leqslant \mathscr{C}\|y\|_\infty + \mathscr{D}\eta$.

En Coq, le lemme correspondant au Lemme 1 est nommé `build_bound_mult_loc`. La démonstration de ce résultat repose sur une analyse d'erreur en avant relativement naïve. La quantité $X_1 \oplus (\widetilde{A_2} \otimes X_3)$ correspond à un pas de l'évaluation "à la Horner" en arithmétique à virgule flottante (c'est-à-dire $\widetilde{R}(h, \widetilde{A}, \widetilde{y_n})$) et $(A_1 + A_2A_3)y$ correspond à l'évaluation mathématique $R(hA)\widetilde{y_n}$. Ces évaluations étant construites à partir d'évaluations de Horner plus simples, on peut reconstruire pas à pas une borne sur l'erreur locale de l'expression entière. En l'absence de dépassement de capacité inférieur, $\mathscr{D} = 0$.

Prenons l'exemple de la méthode RK2 (sans prise en compte de l'*underflow*). On instancie le Lemme 1 avec $y = \widetilde{y_n}$, $X_1 = \widetilde{y_n}$, $A_1 = I$, $\widetilde{A_2} = \circ\left[h\widetilde{A}\right]$, $A_2 = hA$, $X_3 = \circ\left[\widetilde{y_n} + \frac{h}{2}\widetilde{A}\widetilde{y_n}\right]$ et $A_3 = I + \frac{hA}{2}$. On peut vérifier que l'expression $\|X_1 \oplus (\widetilde{A_2} \otimes X_3) - (A_1 + A_2A_3)y\|_\infty$ correspond à l'erreur locale ε_{n+1} pour la méthode RK2. Il faut alors exhiber C_1, C_2 et C_3. Il apparaît de façon évidente que $C_1 = 0$ car $\|X_1 - A_1y\|_\infty = \|\widetilde{y_n} - I\widetilde{y_n}\|_\infty = 0$. Pour exhiber C_2, on borne $|||h\widetilde{A} - hA|||_\infty$, ce qui est assez naturel par dépliage de la définition de $|||\cdot|||_\infty$ et en appliquant des résultats génériques d'analyse d'erreurs d'arrondi. Enfin, il reste à exhiber C_3, *i.e.* à borner $\|X_3 - A_3y\|_\infty$. Pour y parvenir, on réapplique le Lemme 1 avec $y = X_1 = X_3 = \widetilde{y_n}$, $A_1 = A_3 = I$, $A_2 = \frac{hA}{2}$ et $\widetilde{A_2} = \circ\left[\frac{h}{2}\widetilde{A}\right]$. Il est trivial d'exhiber C_1 et C_3, C_2 étant obtenu de la même manière qu'à l'étape précédente.

Nous avons utilisé cette méthodologie pour borner les erreurs locales des méthodes d'Euler et de RK2 en *binary64* (en négligeant les *underflows*). Pour la méthode d'Euler :

$$\forall n \in \mathbb{N}^*, \varepsilon_n \leqslant (u + (u + 3{,}12\gamma_d)h|||A|||_\infty)\,\|y_{n-1}\|_\infty. \tag{1}$$

Le lemme Coq associé est nommé `Euler_loc_FLX`. Pour la méthode RK2 :

$$\forall n \in \mathbb{N}^*, \varepsilon_n \leqslant \left(u + (11{,}3u + 2{,}56\gamma_d)(h|||A|||_\infty + h^2|||A|||_\infty^2)\right)\|y_{n-1}\|_\infty. \tag{2}$$

Le lemme Coq associé est nommé `RK2_loc_FLX`. La méthodologie peut paraître fastidieuse mais il est néanmoins possible de partiellement l'automatiser. Tout d'abord, à chaque sous-application du Lemme 1, nous utilisons la tactique `eapply` de Coq, qui va permettre d'inférer automatiquement les valeurs des paramètres y, X_1, A_1, $\widetilde{A_2}$, A_2, X_3 et A_3. Il ne reste alors plus qu'à instancier manuellement les valeurs respectives de C_1, C_2 et C_3 puis à prouver les 3 hypothèses du lemme. Les bornes obtenues s'avèrent relativement lisibles (voir Équations (1) et (2)) et ne font pas apparaître de termes d'ordre supérieur (en u^2, u^3, etc). Nous avons en effet majoré ces termes à chaque étape de la construction de la borne d'erreur en utilisant la tactique automatique `interval` [16], ce qui nécessite des hypothèses sur u : en *binary64*, on a $u^2 \leqslant 2^{-53}u$ mais des hypothèses plus faibles suffisent, *e.g.* $u^2 \leqslant 0{,}01u$.

6 Erreurs globales

Dans de précédents travaux [6], un théorème général permet de construire de façon systématique une borne sur l'erreur globale d'une méthode à partir des bornes exhibées sur les erreurs locales précédentes. Nous généralisons ce résultat au cas multidimensionnel :

Théorème 2. Passage des erreurs locales à l'erreur globale
Soit $C, D \geqslant 0$. *Supposons que pour tout* $n \in \mathbb{N}^*, \varepsilon_n \leqslant C\|\widetilde{y_{n-1}}\|_\infty + D\eta$ *et que* $0 < C + |||R(hA)|||_\infty < 1$ *(condition de stabilité). Alors :*

$$\forall n, E_n \leqslant (C + |||R(hA)|||_\infty)^n \left(\varepsilon_0 + \frac{nC\|y_0\|_\infty}{C + |||R(hA)|||_\infty}\right) + nD\eta.$$

En Coq, le théorème correspondant au Théorème 2 est nommé `error_loc_to_glob`. Si l'énoncé du théorème est très proche de celui démontré dans [6], la démonstration l'est également. En effet, le choix de mener une analyse par normes a permis de généraliser le résultat appliqué aux systèmes scalaires unidimensionnels pour qu'il s'étende aux systèmes linéaires multidimensionnels, *i.e.* matriciels, en remplaçant les valeurs absolues par des normes. À partir des résultats exhibés dans la section 5, nous bornons l'erreur globale correspondant aux méthodes d'Euler et de Runge-Kutta d'ordre 2 par application directe du Théorème 2. Il suffit en effet de remplacer la variable C par la valeur correspondante dans le Théorème 2, *e.g.* $u + (u + 3{,}12\gamma_d)h|||A|||_\infty$ pour la méthode d'Euler.

7 Conclusion et perspectives

Nous avons proposé une méthodologie générique pour borner les erreurs d'arrondi associées à une certaine classe de méthodes numériques en tenant compte des dépassements de capacité inférieurs. Par rapport à nos travaux antérieurs [6], nous proposons une généralisation aux systèmes multidimensionnels *via* une analyse par normes. Cette généralisation est non-triviale car elle repose sur des résultats fondamentaux d'analyse matricielle, mais la méthodologie appliquée reste similaire. Nous instancions nos résultats à deux méthodes classiques (Euler et RK2) et exhibons des bornes qui, dans le cas de méthodes stables, sont quasi-linéaires en le nombre d'itérations. Par ailleurs, nous formalisons l'ensemble du raisonnement dans Coq. La formalisation des résultats génériques représente environ 3300 lignes de Coq en `FLX` et 3600 lignes en `FLT`. L'instanciation aux méthodes d'Euler et de RK2 représente environ 1200 lignes de Coq.

Nous envisageons de factoriser une partie des résultats commune aux formats `FLX` et `FLT`. Flocq contient en effet des résultats liant ces deux formats [7]. De plus, bien que nous nous basions sur la bibliothèque Mathematical Components, la formalisation n'utilise pas les tactiques propres à `ssreflect`, qui permettraient probablement de réduire la taille des démonstrations. Nous pourrions également combiner nos preuves à des définitions et résultats formalisés par Roux dans de précédents travaux [18], comme la notation γ (et ses propriétés), ainsi que certains résultats fondamentaux, *e.g.* une borne sur l'erreur d'arrondi du produit scalaire de deux vecteurs. Une autre perspective pour ce travail est l'extension de la méthodologie à d'autres classes de méthodes, comme les méthodes implicites et multi-pas, ou à d'autres classes de systèmes, comme les systèmes affines ou non-linéaires. Une autre perspective envisagée est la combinaison des erreurs d'arrondi et des erreurs de méthode. Pour évaluer la finesse de la borne obtenue, nous pourrions, comme dans le cas unidimensionnel [6], comparer numériquement la borne d'erreur à l'erreur d'arrondi effectivement commise. Enfin, nous envisageons de vérifier que la borne d'erreur d'arrondi obtenue est négligeable par rapport à l'erreur de méthode effectivement commise.

Références

[1] J. Alexandre dit SANDRETTO et A. CHAPOUTOT : Validated explicit and implicit Runge-Kutta methods. *Reliable Computing*, 22, 2016.

[2] Y. BERTOT, G. GONTHIER, S. O. BIHA et I. PASCA : Canonical big operators. *In International Conference on Theorem Proving in Higher Order Logics*, pages 86–101. Springer, 2008.

[3] M. BERZ et K. MAKINO : Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4(4):361–369, 1998.

[4] S. BOLDO : Floats & Ropes : a case study for formal numerical program verification. *In 36th International Colloquium on Automata, Languages and Programming*, volume 5556 de *LNCS - ARCoSS*, pages 91–102, Rhodos, Greece, juillet 2009. Springer.

[5] S. BOLDO, F. CLÉMENT, J.-C. FILLIÂTRE, M. MAYERO, G. MELQUIOND et P. WEIS : Wave Equation Numerical Resolution : a Comprehensive Mechanized Proof of a C Program. *Journal of Automated Reasoning*, 50(4):423–456, avril 2013.

[6] S. BOLDO, F. FAISSOLE et A. CHAPOUTOT : Round-off Error Analysis of Explicit One-Step Numerical Integration Methods. *In 24th IEEE Symposium on Computer Arithmetic*, pages 82–89, London, United Kingdom, juillet 2017.

[7] S. BOLDO et G. MELQUIOND : *Computer Arithmetic and Formal Proofs*. ISTE Press - Elsevier, décembre 2017.

[8] O. BOUISSOU et M. MARTEL : GRKLib : a guaranteed Runge-Kutta library. *In International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics*, 2006.

[9] L. FOUSSE : Multiple-precision correctly rounded Newton-Cotes quadrature. *ITA*, 41(1):103–121, 2007.

[10] P. HENRICI : *Error propagation for difference methods*. The SIAM series in applied mathematics. John Wiley, 1963.

[11] N. J. HIGHAM : *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd édition, 2002.

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

[13] F. IMMLER et J. HÖLZL : Numerical analysis of ordinary differential equations in Isabelle/HOL. *In Interactive Theorem Proving*, volume 7406 de *LNCS*, pages 377–392. Springer Berlin Heidelberg, 2012.

[14] F. IMMLER et C. TRAUT : The flow of ODEs : Formalization of Variational Equation and Poincaré Map. *Journal of Automated Reasoning*, 62(2):215–236, Feb 2019.

[15] A. MAHBOUBI et E. TASSI : Mathematical Components. `https://math-comp.github.io/mcb/`, 2017.

[16] G. MELQUIOND : Proving bounds on real-valued functions with computations. *In International Joint Conference on Automated Reasoning, IJCAR 2008*, Sydney, Australia.

[17] M. NEHER, K. R. JACKSON et N. S. NEDIALKOV : On Taylor model based integration of ODEs. *SIAM Journal on Numerical Analysis*, 45(1):236–262, 2007.

[18] P. ROUX : Formal Proofs of Rounding Error Bounds - With Application to an Automatic Positive Definiteness Check. *J. Autom. Reasoning*, 57(2):135–156, 2016.