La double précision suffit-elle à l'exascale ?

Louise Ben Salem-Knapp[1,2], Thibaud Vazquez-Gonzalez[1] et William Weens[1,3]
{louise.bensalem-knapp, thibaud.vazquez-gonzalez, william.weens}@cea.fr

[1]*CEA DAM DIF, F-91297, Arpajon Cedex, France ;* [2]*Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France ;* [3]*Université Paris-Saclay, CEA, Laboratoire en Informatique Haute Performance pour le Calcul et la Simulation, 91680 Bruyères-le-Châtel, France*

Résumé

La croissance des capacités de calcul des machines permet d'obtenir des résultats de simulation de plus en plus précis. Ces résultats sont souvent calculés en binary64 (double précision) avec l'idée que les erreurs d'arrondi ne sont pas significatives. Or, l'*exascale* permet d'augmenter le nombre d'opérations et des problèmes d'accumulation d'erreurs d'arrondi pourraient apparaître. Augmenter la précision des nombres flottants pour remédier à ce problème n'est pas envisageable car le surcoût en mémoire, en temps de calcul et en énergie ferait perdre une partie importante des performances des nouvelles machines. Il est donc important de mesurer la robustesse du binary64 en anticipant les ressources de calcul à venir afin d'assurer la pérennité de celle-ci dans les simulations numériques. C'est dans ce but que des expériences numériques ont été réalisées et sont présentées dans cet article. En montrant que les erreurs d'arrondi restent dominées par les erreurs du schéma, les résultats ont permis de valider le binary64 dans ces simulations.

1 Introduction

L'industrie de la simulation HPC s'applique à des domaines scientifiques toujours plus nombreux — biologie, météorologie, astrophysique, aérodynamique, etc. —, et doit donc évoluer régulièrement pour s'adapter aux changements théoriques et matériels, ainsi qu'aux nouvelles demandes réglementaires ou commerciales. Par exemple, les résultats de simulation doivent de plus en plus souvent être assortis d'une mesure des incertitudes [5].

Dans un contexte où les ressources de calcul (HPC) sont toujours renouvelées (GP-GPU), de plus en plus parallélisées (MPI, Multithreading, Vectorisation) et les performances constamment améliorées, il n'est pas surprenant de voir les attentes vis-à-vis des résultats de simulation se renforcer. Pour des résultats à la

fois plus précis et mieux contrôlés, la stratégie couramment employée pour profiter des possibilités promises par le HPC consiste à appliquer des schémas numériques éprouvés à des problèmes de taille augmentée. Cependant, cette stratégie génère en elle-même davantage d'erreurs, ce qui peut paraître contradictoire. En effet, comme les calculs en précision finie introduisent des erreurs d'arrondi, plus la taille des problèmes simulés augmente, plus le nombre d'opérations augmente, et avec lui la proportion d'erreur, dégradant ainsi les résultats. C'est pourquoi il est important de rechercher le meilleur compromis entre réduction des erreurs par une description plus fidèle du système étudié et dégradation des résultats par les erreurs d'arrondi dues à l'arithmétique flottante. D'où la question pour la prochaine génération de super-calculateurs : *la double précision suffit-elle à l'exascale ?*

L'utilisation des outils théoriques usuels ne fournit pas de réponses satisfaisantes à cette question. À cause du grand nombre d'opérations dans une simulation et de la complexité des programmes, les bornes d'erreur obtenues sont souvent trop grandes pour être utiles. Des approches d'encadrement des erreurs d'arrondi basées sur les preuves formelles ont ainsi récemment vu le jour. L'objectif de celles-ci est de déterminer des bornes théoriques fines d'encadrement qui soient davantage compatibles avec les tolérances de l'industrie. On notera en particulier les travaux de [2] pour l'étude d'un schéma en différences finies sur l'équation d'onde, ainsi que les travaux de [3] pour l'analyse de la méthode de Runge-Kutta. Néanmoins, pour des programmes de simulations plus complexes, l'approche empirique reste nécessaire afin de détecter le poids réel des erreurs d'arrondis dans les résultats numériques.

Le travail présenté dans cet article s'inscrit dans cette volonté de mieux quantifier l'erreur numérique liée à l'arithmétique flottante dans les simulations d'hydrodynamique. Elle s'appuie sur une méthode d'estimation de l'erreur d'arrondi par sous précision dont l'implémentation dans nos outils d'analyse a été nommé *weak floats*. La polyvalence des outils développés, leur facilité d'emploi et leur rapidité d'exécution ont permis d'obtenir des données suffisantes pour réaliser une preuve de concept de leur application pour une analyse empirique de nombreux modèles physiques plus complets.

La section 2 introduit le modèle d'erreur numérique et présente les *weak floats*. La section 3 décrit ensuite le modèle physique et les cas tests utilisés. Les résultats numériques obtenus avec les types flottants de la norme IEEE-754 [6] [10] et les *weak floats* sont ensuite analysés en section 4. Enfin, la section 5 conclut le travail, et propose une tentative de réponse à la question du titre en se basant sur une extrapolation des résultats obtenus.

2 Modèle de l'erreur numérique

2.1 Mesure de l'erreur due à l'arithmétique flottante

Nous nous intéressons à l'erreur numérique, notée ε^{num}, que nous définissons comme l'écart entre le résultat théorique issu du modèle mathématique, y^{model}, et

le résultat obtenu par la simulation, $y^{simulation}$. Dans ce travail, nous avons utilisé la norme L_2 et nous nous concentrons sur le cas où y^{model} et $y^{simulation}$ sont des champs spatio-temporels, si bien que :

$$\varepsilon^{num} := ||y^{model} - y^{simulation}||_{L_2} . \tag{1}$$

L'erreur numérique ε^{num} se caractérise par deux contributions : l'erreur due à la discrétisation des équations, ε^{scheme}, et l'erreur créée par les calculs en précision finie, ε^{float}. En effet, toute discrétisation d'équations continues introduit une erreur, mais si le schéma numérique considéré est consistant, cette erreur de discrétisation ε^{scheme} peut se voir comme l'erreur causée par le manque d'information. Il suffirait donc d'ajouter de l'information (des mailles, des particules, etc.) pour diminuer cette erreur — la diminution de l'erreur par rapport à l'information injectée définissant l'ordre du schéma. Par ailleurs, l'erreur provenant des calculs en précision finie ε^{float} résulte de l'accumulation d'erreurs due à l'arrondi commis pour chaque opération [8] [10].

Pour des raisons de stabilité numérique, plus on discrétise finement les équations (ce qui revient à ajouter de l'information), plus on augmente le nombre d'opérations. Il est ainsi clair que les deux erreurs, ε^{scheme} et ε^{float}, jouent dans des sens opposés. Pour être plus précis, c'est-à-dire pour réduire ε^{scheme}, il est nécessaire d'effectuer plus d'opérations. Mais chaque opération peut introduire une erreur d'arrondi supplémentaire, ce qui risque d'augmenter ε^{float}. On s'attend ainsi à ce que ε^{num} admette un minimum en fonction du nombre d'opérations sur lequel repose la simulation, correspondant à une certaine taille de maillage qu'il ne faudrait pas dépasser.

2.2 Modèle d'erreur sur un schéma numérique explicite

De même que pour une équation différentielle ordinaire [1], le comportement attendu de l'erreur numérique pour un problème en dimension 1 discrétisé par un schéma numérique à l'ordre 1 est le suivant :

$$\varepsilon^{num} \leq C_1 h + C_2 h^{-1} , \tag{2}$$

où C_1, C_2 sont des constantes propres au cas étudié et au schéma, et h un paramètre caractérisant la finesse de discrétisation des équations (par exemple égal au pas d'espace pour une discrétisation spatiale en 1D). On retrouve l'ordre de convergence du schéma numérique dans le premier membre, $C_1 h$. Le second membre, $C_2 h^{-1}$, fait référence à l'accumulation de l'erreur d'arrondi. Il dépend du nombre d'opérations et donc du pas d'espace. Lorsque le nombre de mailles devient infini, et donc que h tend vers 0, on s'attend à ce que le comportement asymptotique de ces deux contributions à l'erreur numérique soit donné par :

$$\lim_{h \to 0} \varepsilon^{scheme} = 0 , \quad \lim_{h \to 0} \varepsilon^{float} = +\infty . \tag{3}$$

Cette décomposition de l'erreur numérique est illustrée qualitativement dans la fig. 1, qui souligne la part de ε^{num} due à la discrétisation ε^{scheme} en hachures bleues et celle due aux erreurs d'arrondi ε^{float} en hachures vertes.

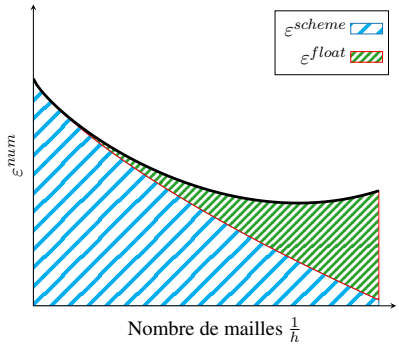

FIGURE 1 – Représentation qualitative du comportement de l'erreur numérique et de ses composantes dans un schéma convergent.

Pour certifier que le code est utilisé dans son domaine de validité, il semble ainsi nécessaire d'obtenir une estimation du point de divergence de l'erreur numérique, c'est-à-dire le nombre de mailles pour lequel la solution simulée commence à s'écarter significativement de la convergence théorique du schéma.

2.3 Les weak floats

Dans ce travail, nous proposons d'utiliser les *weak floats* pour mesurer la part de l'erreur d'arrondi dans les simulations. L'idée des *weak floats* est d'utiliser des précisions inférieures au binary64, de trouver les points de divergence pour ces sous-précisions, puis d'extrapoler les valeurs obtenues pour estimer le point de divergence du binary64. Une approche similaire exploitant une sous-précision a été utilisée dans [7]. Pour des raisons de rapidité, les opérations entre *weak floats* s'effectuent sur un type souche (*single*, *double*, *long double*), dont la taille de la mantisse est supérieure à celle du *weak float*, puis le résultat est tronqué et arrondi. Les *weak floats* conservent la plage d'exposants du flottant utilisé comme souche. Dans la suite, *weak* N désigne un flottant en binary64 dont les N premiers *bits* de la mantisse sont utilisés, les autres étant tronqués (donc une précision de $N+1$ *bits* avec le *bit* implicite).

On notera qu'en plus des opérations de troncature et d'arrondi s'ajoute l'impossibilité pour les compilateurs d'utiliser de nombreuses optimisations. Ceci se traduit par un ralentissement de l'exécution pour les *weak floats* alors même que la précision est réduite.

3 Description de l'expérience numérique

Les équations de l'hydrodynamique — appelées aussi équations d'Euler — sont un bon candidat pour évaluer l'impact des erreurs d'arrondi. Ce système

d'équations différentielles couplées est au coeur d'un grand nombre de simulations numériques réalisées dans l'industrie et le monde académique. C'est pourquoi il est important de pouvoir garantir une certaine qualité des résultats obtenus.

Le système d'équation d'Euler pour un fluide s'écrit

$$\partial_t \begin{pmatrix} \rho \\ \rho \boldsymbol{u} \\ \rho E \end{pmatrix} + \boldsymbol{\nabla} \cdot \begin{pmatrix} \rho \boldsymbol{u} \\ \rho \boldsymbol{u} \otimes \boldsymbol{u} + p \mathbb{I}_3 \\ (\rho E + p) \boldsymbol{u} \end{pmatrix} = \mathbf{0} \, , \tag{4}$$

avec ρ, E, p et $\boldsymbol{u}$ respectivement la densité, l'énergie totale spécifique, la pression et le vecteur vitesse du fluide considéré.

Le système d'équation (4) est fermé en utilisant une équation d'état reliant la pression, l'énergie et la densité. Seule l'équation d'état des gaz parfaits est utilisée dans la suite. Cette équation d'état s'écrit : $p = (\gamma - 1)\rho e$, avec γ le coefficient polytropique et e l'énergie interne spécifique du fluide.

Dans les simulations présentées, nous avons calculé l'approximation numérique des solutions avec un schéma de type Godounov à l'ordre 1 avec le solveur de Riemann approchée Rusanov (méthode de Godounov [4], [11], [9]).

Nous avons retenu deux cas tests pour cette étude. Les cas utilisés possèdent une solution analytique permettant des mesures précises de l'impact des erreurs d'arrondi. Ils font aussi intervenir plusieurs caractéristiques que l'on retrouve dans les simulations numériques à grande échelle réalisées dans l'industrie ou le monde académique (choc, discontinuité de contact, détente, grand déplacement, etc.)

Le premier cas-test est une advection à vitesse constante d'un profil de densité gaussien sur un domaine avec des conditions aux bords périodiques. La solution analytique du profil de densité au temps final est identique au profil initial puisqu'elle correspond à un tour complet.

Le second cas-test, qui correspond au tube à choc de Sod, est un problème de Riemann à deux états. Ce test 1D est réalisé avec des conditions aux bords de Neumann. À l'instant initial, les deux fluides sont au repos, et soumis à des pressions différentes. Dès $t > 0$, la différence de pression entre les deux états va créer un choc, une discontinuité de contact et une détente.

Remarque. Dans les cas tests utilisés, on observe que le schéma numérique ne produit pas d'événements catastrophiques (perte sévère de nombres significatifs). L'erreur apparaît de façon inhomogène en espace, mais elle apparaît progressivement avec le nombre d'opérations sans saut brutal.

4 Résultats & discussions

Les résultats de cette section sont présentés sous forme de graphiques en échelle logarithmique avec : en abscisse le nombre de mailles et en ordonnée la différence en norme L^2 entre la densité moyenne par maille obtenue par la simulation, notée $\bar{\rho}_{sim.}$, et celle de la solution analytique, notée $\bar{\rho}_{exact}$: $\varepsilon^{num} := \|\bar{\rho}_{sim.} - \bar{\rho}_{exact}\|_{L^2}$.

Dans le cas général, lorsque la solution analytique n'est pas connue, il est possible d'utiliser la solution obtenue avec le binary64 comme solution de référence, s'il est possible de s'assurer de sa convergence en maillage.

4.1 Variations sur la taille de la mantisse

La fig. 2 est un résumé d'un ensemble de simulations concernant l'advection d'une gaussienne simulée avec différents *weak floats*. Chaque point de chaque courbe correspond à l'erreur numérique ε^{num} issue d'une simulation complète. Sur cette figure, il est d'abord intéressant de remarquer que l'on retrouve bien une évolution de l'erreur conforme à la prédiction du modèle qualitatif (équation (2) et fig. 1). On observe également que le binary64 suit l'ordre du schéma. À l'inverse, les autres précisions divergent par rapport à cette pente et on peut voir que l'abscisse du point de divergence de chaque type de nombre flottant croit avec la taille de la mantisse.

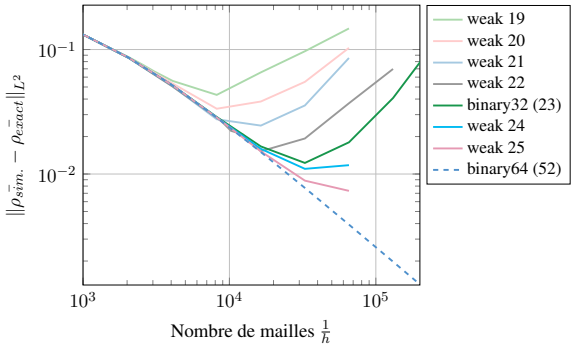

FIGURE 2 – Courbe d'erreur de l'advection d'une gaussienne simulée avec différentes précisions.

4.2 Prédiction du point de divergence du binary64

La figure 2 nous montre que le binary64 ne semble pas affectée par l'erreur d'arrondi pour un nombre de mailles allant jusqu'au million. Cela ne signifie pas que le point de divergence du binary64 n'existe pas, mais qu'il ne peut simplement pas être directement déterminé par la simulation. Il semble néanmoins possible de prédire ce point par extrapolation. En effet, en faisant varier la taille de mantisse, on observe une quasi-linéarité du point de divergence entre la taille de la mantisse et le nombre d'opérations.

Une recherche systématique du point de divergence pour un cas-test donné permet de tracer les deux graphiques de la fig. 3. Cette figure montre, pour chaque précision étudiée, le point de divergence obtenu. Par régression linéaire, on confirme expérimentalement la linéarité en échelle logarithmique. Chaque cas semble produire de l'erreur d'arrondi régulièrement au cours du calcul. Le taux de production de l'erreur croit régulièrement et sa valeur dépend de la mantisse utilisée. Concrètement, pour les cas étudiés, cela implique qu'un *bit* supplémentaire dans la mantisse

permet de repousser exponentiellement le nombre de mailles nécessaires (et donc d'opérations) pour atteindre le point de divergence. Le point de divergence n'est pas pour autant doublé pour chaque *bit* de mantisse ajouté car avec plus de mailles, les résultats deviennent plus précis et nécessitent plus de *bits* dans la mantisse.

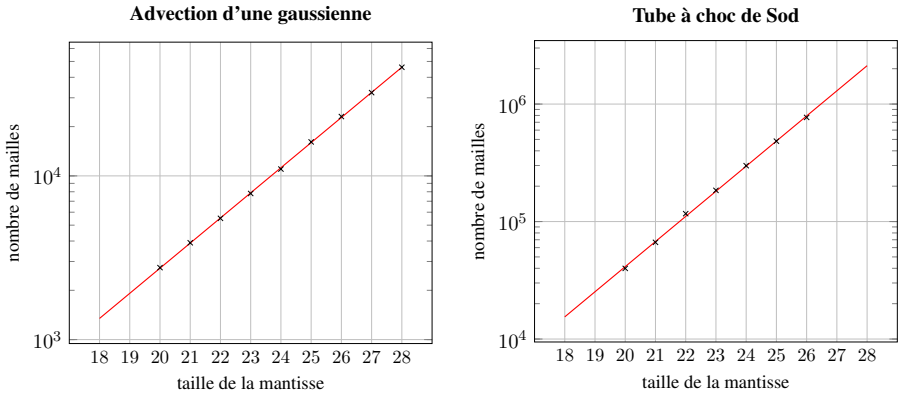

FIGURE 3 – Nombre de mailles du point de divergence en fonction de la mantisse pour deux cas. Les diagrammes sont tracés en échelle logarithmique sur l'axe des ordonnées. La droite (en rouge) est obtenue par régression linéaire.

Après extrapolation par régression linéaire, les droites obtenues sont très proches des points de divergence et autorisent donc une extrapolation. On en conclut qu'en dimension 1, le nombre d'opérations nécessaire pour faire diverger le binary64 est de 220 millions de mailles avec 9 milliards d'itérations pour l'advection et de 286 milliards de mailles avec 251 milliards d'itérations pour le tube à choc. Ces valeurs sont plusieurs ordres de grandeur au-dessus des tailles utilisées ou qui seront utilisées sur l'*exascale*.

5 Conclusion

Grâce aux *weak floats* et aux résultats obtenus sur des maillages de taille élevée du code de calcul massivement parallèle VARIANT [12], une étude de l'impact des erreurs d'arrondi a pu être menée sur des simulations d'hydrodynamique de gaz parfaits en condition HPC.

Les simulations présentées dans cette étude ont pu atteindre le point de divergence sur plusieurs tailles de mantisse. Nous avons constaté que ces points de divergence croissent linéairement avec la taille de la mantisse. Cette croissance régulière nous a autorisé à extrapoler le nombre de mailles nécessaires pour atteindre le point de divergence du binary64. Ce nombre se trouve être beaucoup trop grand pour être simulé, même sur les machines exaflopiques. Ce résultat renforce notre confiance dans le binary64 pour l'utilisation de maillages extrêmement fins en hydrodynamique et permet même d'envisager de réduire la précision sur certains calculs sans impacter le résultat.

La prochaine étape consiste à s'intéresser à des codes plus complexes que les cas-tests étudiés dans ce travail, d'un point de vue empirique (par exemple avec les

weak floats) mais également théorique, ces deux approches étant complémentaires. Pour la partie théorique, comme il semble possible de calculer les bornes de l'erreur d'arrondi sur chaque routine du code, et comme ces routines sont communes à de nombreux schémas (reconstruction, limiteurs, solveurs, Runge-Kutta), il s'agirait de chercher à assembler les bornes théoriques pour construire une borne globale de l'erreur d'arrondi. Ces bornes seront plus larges que les bornes empiriques, mais elles devraient permettre de valider n'importe quel cas test.

Références

[1] Atkinson, K.E., Han, W., Stewart, D. : Euler's method, chap. 2, pp. 15–36. John Wiley & Sons, Ltd (2011)

[2] Boldo, S., Clément, F., Filliâtre, J.C., Mayero, M., Melquiond, G., Weis, P. : Wave equation numerical resolution : a comprehensive mechanized proof of a C program. Journal of Automated Reasoning **50**(4), 423–456 (04 2013)

[3] Boldo, S., Faissole, F., Chapoutot, A. : Round-off error and exceptional behavior analysis of explicit Runge-Kutta methods. IEEE Transactions on Computers (2019)

[4] Godunov, S.K. : Eine Differenzenmethode für die Näherungsberechnung unstetiger Lösungen der hydrodynamischen Gleichungen. Mat. Sb., Nov. Ser. **47**, 271–306 (1959)

[5] Heroux, M.A., Carter, J., Thakur, R., Vetter, J.S., McInnes, L.C., Ahrens, J., Munson, T., Neely, J.R. : Ecp software technology capability assessment report. www.exascaleproject.org (02 2020)

[6] IEEE : IEEE Standard for Floating-Point Arithmetic. Institute of Electrical and Electronics Engineers IEEE Std 754-2008 pp. 1–70 (2008)

[7] Izquierdo, L.R., Polhill, J.G. : Is Your Model Susceptible to Floating-Point Errors ? Journal of Artificial Societies and Social Simulation **9**(4), 1–4 (2006)

[8] Knuth, D.E. : The Art of Computer Programming, Volume 2 : Seminumerical Algorithms. Addison-Wesley, Boston, third edn. (1997)

[9] LeVeque, R.J. : Finite Volume Methods for Hyperbolic Problems. Cambridge Texts in Applied Mathematics, Cambridge University Press (2002)

[10] Muller, J.M., Brunie, N., de Dinechin, F., Jeannerod, C.P., Joldes, M., Lefèvre, V., Melquiond, G., Revol, N., Torres, S. : Handbook of Floating-Point Arithmetic, 2nd edition. Birkhäuser Boston (2018)

[11] Toro, E.F. : Riemann Solvers and Numerical Methods for Fluid Dynamics. Springer Berlin Heidelberg (2009)

[12] Weens, W. : Toward a predictive model to monitor the balance between discretization and rounding errors in hydrodynamic simulations. SIAM Conference on Parallel Processing for Scientific Computing (2020)