Modeling round-off errors in hydrodynamic simulations

William Weens, Thibaud Vazquez-Gonzalez, Louise Ben Salem-Knapp

Modeling round-off errors in hydrodynamic simulations

William Weens[1,2], Thibaud Vazquez-Gonzalez[1], Louise Ben Salem-Knapp[1,3]

[1] CEA DAM DIF, F-91297, Arpajon Cedex
{william.weens, thibaud.vazquez-gonzalez, louise.bensalem-knapp}@cea.fr
[2] Université Paris-Saclay, CEA, Laboratoire en Informatique Haute Performance pour le Calcul et la simulation, 91680 Bruyères-le-Châtel, France
[3] Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria, Laboratoire Méthodes Formelles, 91190, Gif-sur-Yvette, France

Abstract. The growth of the computing capacities makes it possible to obtain more and more precise simulation results. These results are often calculated in *binary64* with the idea that round-off errors are not significant. However, exascale is pushing back the known limits and the problems of accumulating round-off errors could come back and require increasing further the precision. But working with extended precision, regardless of the method used, has a significant cost in memory, computation time and energy and would not allow to use the full performance of HPC computers. It is therefore important to measure the robustness of the *binary64* by anticipating the future computing resources in order to ensure its durability in numerical simulations. For this purpose, numerical experiments have been performed and are presented in this article. Those were performed with *weak floats* which were specifically designed to conduct an empirical study of round-off errors in hydrodynamic simulations and to build an error model that extracts the part due to round-off error in the results. This model confirms that errors remain dominated by the scheme errors in our numerical experiments.

Keywords: Floating-point · Round-off error · Hydrodynamics · HPC · Exascale

1 Industrial and scientific context

The HPC industry applies to an ever-increasing number of scientific domains — biology, molecular dynamics, meteorology, astrophysics, aerodynamics, etc. It must therefore evolve regularly to adapt to theoretical and material changes, as well as to new regulatory or commercial demands. The list of usual requirements for HPC simulation codes is then increased by new needs. To name a few requirements: improving speed, robustness, accuracy and maintainability, energy efficiency, portability on heterogeneous resources (GPU or FPGA type accelerators), or the measurement of uncertainties in the results [12].

In this context, simulations must both improve and guarantee the results while relying on constantly renewed computing resources (HPC) (GP-GPU) and on parallel computing (MPI, Multithreading, Vectorization) of which performance increases year after year. The strategy commonly used to take advantage of the possibilities promised by the *exascale* consists in relying on the exploitation of proven models and schemes and in increasing the size of the problem studied — such as the size of the mesh, the number of particles, elements, etc. This strategy of improved accuracy leads to a contradiction since finite precision calculations also introduce errors. Indeed, by increasing the size of the simulated problems, the number of operations also increases, thus degrading the results contrary to the expected effect. Therefore, in order to guarantee the quality of the results, it is important to measure the optimal point between their improvement by adding information versus their degradation by round-off errors due to floating-point arithmetic. In other words, in the current context, the question addressed here could be summarized as follows: *Is binary64 sufficient for exascale computing?*

The use of the common theoretical tools such as interval arithmetic, Taylor series or probability do not provide satisfactory answers to this question [6,23,11,8]. Because of the large number of operations in a simulation and the complexity of the programs, the bounds obtained to frame the error are too large to be useful. This is why new approaches have emerged and recent advances in formal proofs have allowed the determination of thin theoretical bounds for framing the error for a finite differences scheme on the wave equation[4] and the Runge-Kutta method[5]. The bounds obtained are compatible with the tolerances of the industry but for simulations of more complex problems, the empirical approach remains unavoidable today in order to detect the real weight of round-off errors in the numerical results.

In this paper, we focus on the measurement of numerical error in hydrodynamic simulations [7], which are the skeleton model of many more complete physical models. A new analysis tool, the *weak floats*, has been developed to improve the traditional empirical approach — extended precision — which is too slow to be conducted on large meshes. After introducing our approach, section 2 describes the implementation of the *weak floats* within a numerical error model. In section 3, the framework of the hydrodynamic applications is detailed: the equations, the discretisation method, the details of the HPC implementation and the test cases used. In section 4, the numerical results are presented. The tests are performed in dimension 1, using Godunov type numerical schemes of order 1. Also using several test cases of reference [24] with the floating types of the IEEE-754 standard [14] and the *weak floats*. Finally, the versatility of the developed tools, their ease of use, and their speed allowed us to obtain sufficient data to conclude this study, while realizing a proof of principle of their application for empirical analysis. By extrapolating the data obtained to exascale, an attempt is made to answer the problematic of required arithmetic precision for the exascale in the conclusion.

2 Model of the numerical error

2.1 Measurement of the error due to floating-point arithmetic

We are interested in the numerical error, denoted ε^{num}, which we define as the deviation between the theoretical result from the mathematical model, y^{model}, and the result obtained by the simulation, $y^{simulation}$. In this work, we focus on the case where y^{model} and $y^{simulation}$ are space-time fields. Using the L_2 norm:

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

This numerical error ε^{num} has two sources: the error due to the discretization of the equations ε^{scheme} and the error created by finite precision computations ε^{float}.

Every numerical scheme introduces errors. But if the scheme is consistent, the discretization error ε^{scheme} can be seen as the error caused by the lack of information. Therefore, it is sufficient to add information (cells, particles, etc.) to decrease this error — the decrease of the error compared to the injected information defines the order of the scheme. Processing this additional information involves more calculations and therefore more operations. The error from finite precision computations ε^{float} can be seen as the accumulation of errors due to rounding committed for each operation [17] [20]. We understand then that these two sources of errors will counteract each other. To be more precise, to reduce ε^{scheme}, we must do more operations but each operation can introduce an additional round-off error increasing then ε^{float}. Therefore there is a minimum for the numerical error ε^{num} which corresponds to a certain problem size that should not be exceeded.

Fig. 1 illustrates (on a floating-point number) what this divergence point means. When the simulation is not accurate enough, the last bits of the mantissa do not contain relevant information (gray pattern error), so the round-off error is hidden. As the calculations proceed, the round-off error accumulates in the last bits and moves up to the significant part of the information. If it exceeds the area of irrelevant information, then the round-off error appears and spoils the relevant information.

In a simulation, this backtracking process is complex. The round-off errors can be compensated, amplified, and occur more or less quickly in certain cells and on certain quantities. The error can contaminate the cell or spread. This behavior depends on the code but also on the input data.

The possibility of detecting round-off errors ε^{float} in a numerical simulation implies constraints on the choice of the numerical experiment. A study can be conducted only if: i) the problem is mathematically well defined, at least in some configurations, ii) there exists convergent numerical schemes to solve the problem, iii) there exists implementations of these schemes that do not introduce systematic errors (via non-ordered sums on sets for example), iv) the software/-component stack (libraries, compilers, chips) is reliable, v) reproducibility [4] is

[4] If several realizations of the same experiment under identical conditions give exactly identical results, then the experiment is said to be reproducible.

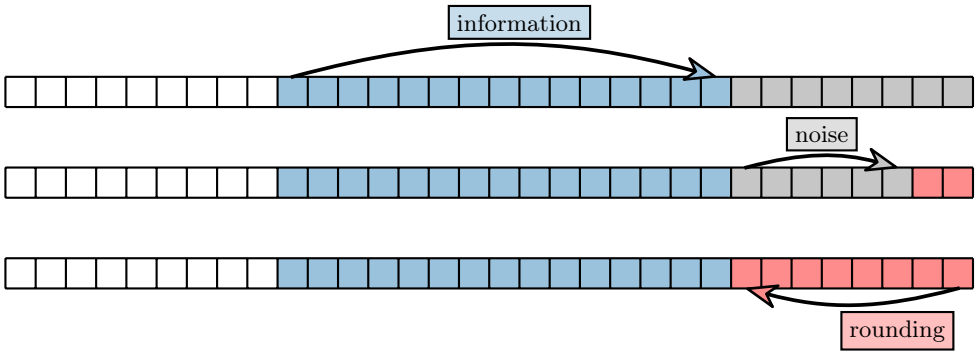

Fig. 1. Idealized illustration of the encounter between the round-off error (red) and the relevant information (blue) in a floating-point number. The irrelevant information (gray) is the scheme error.

ensured, vi) the computational speed is high enough to explore large problem sizes.

2.2 Error model on an explicit numerical scheme

Similarly to an ordinary differential equation [2], the expected behavior of the numerical error for a one-dimensional problem discretized by a numerical scheme of order 1 is:

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

where C_1, C_2 are constants specific to the case studied and to the scheme, and h a parameter characterizing the finiteness of discretization of the equations (for example equal to the space step for a 1D spatial discretization).

The convergence order of the scheme can be seen in the first member $C_1 h$. Without round-off errors, the numerical error would tend toward 0 with h, giving results more and more accurate. The second member $C_2 h^{-1}$ is identified as the accumulation of the round-off error, which depends on the number of operations and thus on the space step. Indeed, in the case of explicit schemes, the number of iterations in time of the simulation N_{it} — on which depends N_{op} the number of operations — depends on the time step Δt, itself constrained by Courant-Friedrichs-Lewy (CFL) number [27], [19] which relates it to the space step h.

When the number of cells becomes infinite, the asymptotic behavior of the two components of the numerical error is:

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

The decomposition of the numerical error into two sources is illustrated qualitatively in fig. 2 by showing the part of ε^{num} due to discretization in blue hatching and the part due to round-off errors in green hatching. This illustration allows to interpret the results presented in the following sections.

It is important to note that when ε^{float} becomes significant compare to ε^{scheme}, then the results deviate quickly from the analytical solution and the

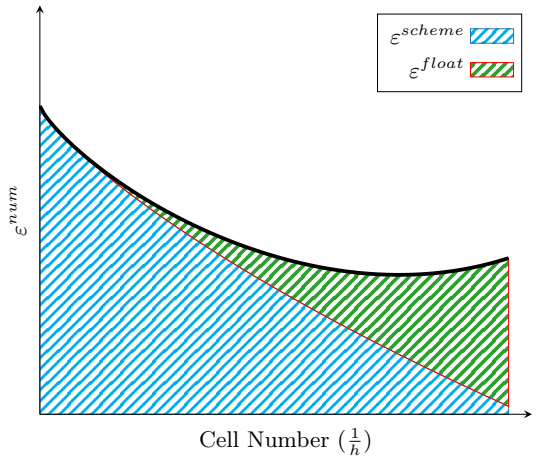

Fig. 2. Qualitative representation of the behavior of the numerical error and of its components with a convergent scheme.

curve diverges. The simulation loses all its meaning and the total error (the measured one) suddenly increases. The quantity of interest that seemed relevant to us is the number of cells at which the divergence occurs. This divergence point depends on the floating-point precision and is written N_d^{double} for *binary64* numbers.

2.3 Weak floats

To certify that the code is used within a valid domain, it is necessary to obtain an estimate of the divergence point of the *binary64* N_d^{double}. It is precisely for this purpose that the *weak floats* have been developed.

The idea of *weak floats* is to use precisions smaller than the *binary64*, to find the divergence points for these sub-precisions, and to extrapolate the values in order to estimate the divergence point of the *binary64*. A similar approach exploiting a sub-precision has been used in [15].

To speed-up the computation, operations between *weak floats* are carried out on a CPU stem type (*binary32*, *binary64*, etc.), which mantissa size is higher than the *weak floats* mantissa, then the result is truncated and rounded. The *weak floats* have therefore the same number of exponent bits as the chosen stem type (*binary64* in our simulations). Due to the truncation and rounding operations, it is impossible for the compilers to use many optimizations. The execution is thus largely slowed down by the *weak floats* while the precision is reduced.

In the following, *weak* N designates a *binary64* floating-point number of which only the N first *bits* of the mantissa are used, the others being truncated (thus a precision of $N+1$ *bits* with the implicit *bit*).

N.B. All *weak floats* are thus numbers obtained by two rounding operations. The first operation is the rounding operation of the IEEE-754 standard on the *binary64* (hardware) and the second is the one of the weak floats (software). With the nearest rounding mode (*nearest*), the double rounding is not equal

to the last rounding $\circ_m(\circ_n(x)) \neq \circ_m(x))$ with $m < n$ (see [21]). This double rounding is thus a rounding mode in itself, different from those of the standard; it has no significant impact on the production of round-off errors and does not produce a visible bias on the results.

3 Description of the numerical experience

3.1 Mathematical model

The hydrodynamic equations — also called Euler equations — are a good candidate to evaluate the impact of round-off errors. This system of coupled differential equations is at the heart of many numerical simulations performed in industry and academia. It still contains a large number of open problems both numerically and theoretically, but it can be solved numerically with schemes of which the convergence is proven and exact solutions exist for certain initial conditions.

The system of Euler equations for a fluid is written:

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

with ρ, E, p and $\boldsymbol{u}$ respectively the density, the total specific energy, the pressure and the velocity vector of the considered fluid.

The system of equations (4) is closed using an equation of state relating pressure, energy and density. Only the ideal gas law is used in the following. This equation of state is written:

$$p = (\gamma - 1)\rho e \ , \tag{5}$$

with γ the polytropic coefficient and e the specific internal energy of the fluid and $E = e + \frac{\boldsymbol{u}^2}{2}$.

3.2 Numerical schemes

The study of the impact of round-off errors is based on comparisons of results from simulations performed with a numerical scheme based on Godunov's method [9], widely documented in the literature [27] [19].

In his founding paper of 1959, Godunov introduced a new discretization for hyperbolic systems of conservation laws such as the Euler equations using the solution of the Riemann problem. In the original version of the method, Godunov uses an exact solver. Later, Lax-Friedrichs [18] and Rusanov [22] independently developed an approximate solver, much less expensive in computation time and accurate to the order of the scheme. Many other Riemann solvers exist and the quality of each of them depends on the domain of use. This study explores the impact of round-off errors on the Rusanov solver.

The convergence of the scheme is only possible if the time step Δt respects the CFL condition [27], [19]:

$$\Delta t \leq a \frac{\Delta x}{\max\limits_{\{i \in N\}} (|u_i + c_i|, |u_i - c_i|)} \ , \tag{6}$$

with a the dimensionless CFL number, N the number of cells and u_i, c_i the fluid velocity and the sound velocity in the cell i.

3.3 Software and hardware environment HPC

As mentioned above, the convergence of the numerical scheme and the reproducibility of the results obtained by the computational code are essential for the study of round-off errors. The possibility for the scheme to perform numerical simulations on a very large number of cells is an additional constraint which requires the development of a code capable of exploiting HPC resources.

In order to respect all the imperatives stated above, the VARIANT code developed for this study is entirely parallelized by sub-domain of calculation. Communication between all sub-domains is provided using the MPI library. Load balancing on the sub-domains is ensured by an equal division of all sub-domains, taking into account the domain edge cells, without any conditional branching on the computations to be performed in each cell (no test on the cell values to select the operations). Each sub-domain with its edge cells is itself processed in parallel using *multithreading*.

One executable is compiled for each type of the floating-point number. These types are: *binary32, binary64* or *weak floats*. All the rounding modes of the standard IEEE 754 [14] are implemented for the *weak floats* (with a nuance for the nearest mode, see 2.3).

3.4 Test cases

The behavior of the numerical scheme implemented is studied using several test cases from the literature. These specific test cases were selected for this study because they have an analytical solution allowing precise measurements of the numerical error. They also involve several characteristics found in large-scale numerical simulations performed in industry or academia (shock, contact discontinuity, relaxation, large displacement, etc.).

The Sod shock tube [24] is a two-state Riemann problem. The computational domain $[0, 1]$ is separated at the position $x = 0.5$, the state on the left has a density $\rho = 1$ and a high pressure $p = 1$ and the state on the right has a density $\rho = 0.125$ and a low pressure $p = 0.1$. Both fluids are described by the ideal gas law (5) with a coefficient $\gamma = 1.4$. This 1D test is performed with Neumann boundary conditions (wall type). At the initial time, both fluids are at rest, i.e. without initial velocity. As soon as $t > 0$, the pressure difference will create a shock which will propagate towards the right. The gas on the left will expand and the gas on the right will compress. The exact solution is given by the solution of a Riemann problem.

The double rarefaction or centered rarefaction is a 1D test which consists in separating a fluid in the middle of the domain $[0, 1]$ to create a vacuum. We initially define a constant pressure $p = 0.4$ and a constant density $\rho = 1$. The internal energy is calculated with the equation (5) and the velocities are constant and opposite. The left half of the domain goes to the left and the right half of the domain goes to the right, which results in a low density in the center and two expansion profiles that meet. The exact solution is also given by solving a Riemann problem.

Advection of a gaussian and a crenel. This one-dimensional test transports a density profile at constant velocity. This test involves a constant velocity $u = 1$ and a constant pressure $p = 1$. The first advection case transports a density profile which has a centered Gaussian shape and the second one transports a crenel. The advection of the density profile being performed at constant speed on a numerical domain $[0, 1]$ with periodic boundary conditions, the analytical solution of the density profile at the final time $t_f = 4$ (corresponding to 4 full turns of the numerical domain) is given by $\rho(x, t_f) = \rho(x, 0)$.

4 Results & Discussions

The results on the different cases are presented in this section, using several schemes and precisions with as reference scheme for all the results: the order 1, in dimension 1, with the Rusanov solver.

To understand what is hidden in a numerical error, the first result presented is the test case of the advection of the Gaussian with different precisions (variations on the size of the mantissa). The following results are compared on graphs in logarithmic scale. On the x-axis is the cell number and on the y-axis is the difference in L^2 norm between the average density per cell $\bar{\rho}$ obtained by the simulation and the analytical or semi-analytical solution.

$$\varepsilon^{num} := \|\bar{\rho}_{sim.} - \bar{\rho}_{exact}\|_{L^2} \tag{7}$$

Next, to isolate the effect of round-off errors alone, the CFL number and the final time are varied. Finally, the divergence point is measured and its relation with the size of the mantissa is quantified.

4.1 Variation on the size of the mantissa

Where the round-off error causes the results to diverge

Fig. 3 is a summary of a set of simulations concerning the advection of a Gaussian simulated with different *weak floats*. Each point of each curve is the numerical error ε^{num} of a complete simulation. On this figure, it is first interesting to notice that the evolution of the error is consistent with the prediction of the qualitative model — equation (2) and fig. 2. We can also observe that the *binary64* follows the order of the scheme while the other precisions diverge from

its curve. The divergence point of each type of floating-point number increases with the size of the mantissa.

The *binary64* does not seem to be affected by round-off errors. The convergence is still valid even at 1 million cells. The computation time required to reach the divergence point of the *binary64* is too large and this point cannot be directly determined numerically.

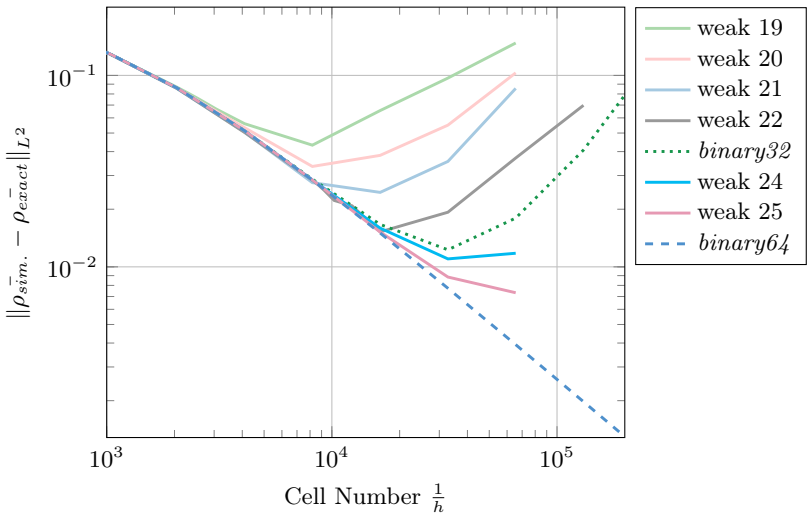

Fig. 3. Error curve of the advection of a Gaussian simulated with different precisions using a Rusanov solver at order 1.

4.2 Variation on the CFL number

Where we isolate the round-off error

Fig. 4 presents a simulation of the Gaussian advection case at order 1 with the Rusanov solver using different CFL numbers. The CFL numbers used are 0.5 (default), 0.25 and 0.125. The mantissa sizes shown are 21 and 23 (*binary32*).

These curves highlight the contribution of the round-off error in the total numerical error. For the same simulation, the decrease of the CFL number will increase the number of iterations without significantly decreasing the error due to the scheme, as long as the state is converged. The only cause of the increase of the error is then the increase of the number of operations (since all the other parameters are identical). With each division of the CFL number, the divergence point decreases for each precision used (see table 1). Finally, the smaller the mantissa is, the smaller the divergence point will be ($N^m_{div} \sim m$). It confirms that the error is due to a defect in the ability of the mantissa to absorb the round-off error of the calculations. This effect does not show up in the formula (2), which invalidate its simplicity and requires a more complete model to be developed.

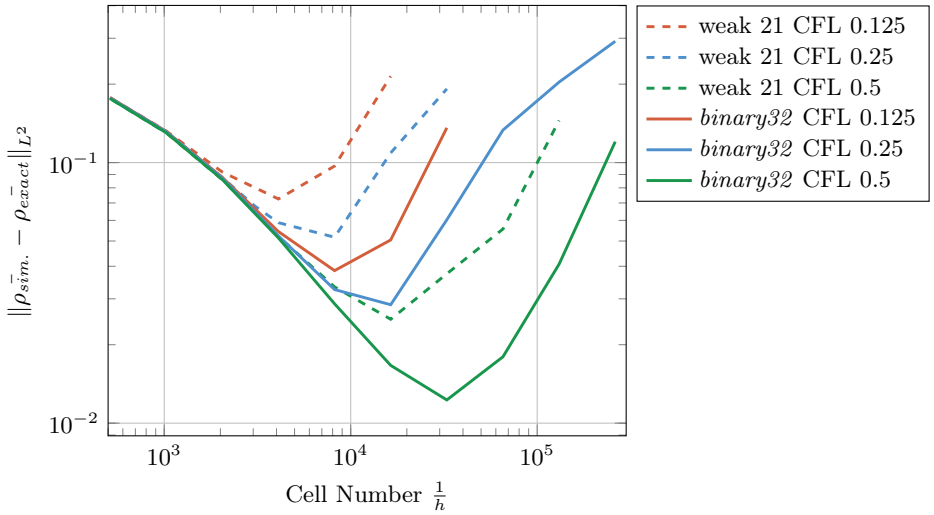

Fig. 4. Error curve of the advection of a Gaussian simulated with different CFL numbers using a Rusanov solver at order 1. *binary32* in solid line and *weak floats 21* in dashed line. The same color is used for a given CFL number.

Table 1. Mantissa size, CFL, number of iterations and divergent mesh size.

mantissa size	CFL	N_{it}	N_{div}
21	0.5	161 338	3 968
21	0.25	182 143	2 240
21	0.125	227 634	1 408
23	0.5	343 495	8 448
23	0.25	395 539	4 864
23	0.125	519 364	3 200
25	0.5	666 172	16 384
25	0.25	832 242	10 240
25	0.125	1 124 164	6 912

4.3 Variation on the final time

Fig. 5 presents the same case as figure 4 using different fraction of the final time. These new final times are $0.25T$, $0.5T$ and $1.0T$ with T the default final time value. The mantissa sizes shown are 21, 23 (*binary32*) and 25.

The variation of the final time in Fig. 5, like the variation of the CFL number, isolates the effect of the number of operations. By increasing the final time, the simulated cases are different. Since it is a transport of a Gaussian on a periodic domain, only the number of turns of the domain changes. The numerical error then becomes larger when the case simulates more turns but the curves remain parallel to each other. However, with more operations, the accumulation of rounding errors is larger and the point of divergence decreases as shown in table 2.

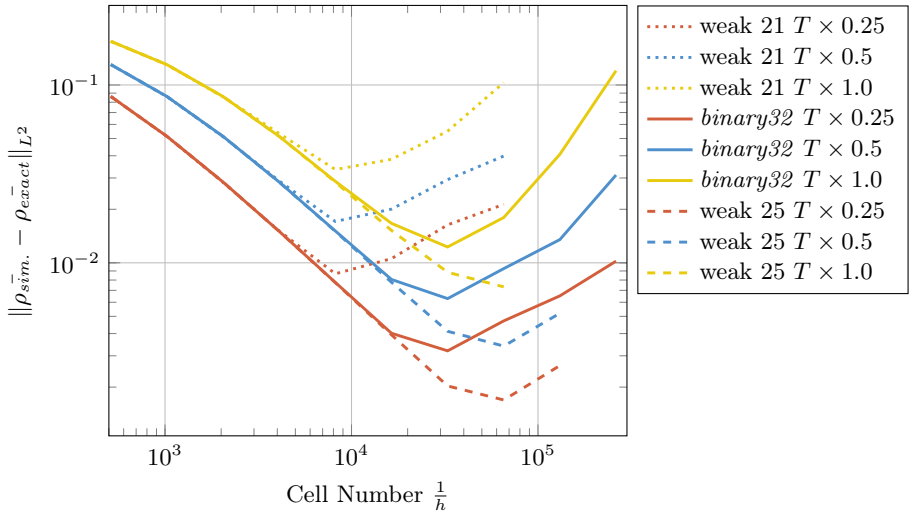

Fig. 5. Error curve of the advection of a Gaussian simulated with different final times using a Rusanov solver at order 1. *binary32* in solid line, *weak floats 21* in dotted line and *weak floats 25* in dashed line. The same color is used for a given final time.

Table 2. Mantissa size, final time and divergent mesh size.

mantissa size	0.25 T	0.5 T	1.0 T
21	4864	4608	3968
23	10240	9728	8448
25	20480	19456	16384

4.4 Prediction of the divergence point

Where we build up an error model

By varying the CFL number, the final time and the mantissa size, we observe a quasi-linearity of the divergence point between mantissa size and number of operations.

A systematic search for the divergence point for a given test case confirms this linearity as shows in fig. 6. This figure plots the divergence point obtained with respect to the mantissa. By linear regression, we can see that the linearity (in logarithmic scale) is very close from each experimental observation and each case can be fit with great accuracy with two parameters α, β by a basic model: $2^{\alpha m+\beta}$ where m is the size of the mantissa.

The test cases seem to produce round-off errors regularly during the calculation. The error production rate increases linearly with respect to the mantissa. Concretely, this implies that an additional *bit* in the mantissa allows to increase exponentially the number of cells needed (and thus of operations) to reach the divergence point. The divergence point is not doubled for each additional bit

because with more cells, the results become more accurate and require more bits in the mantissa.

Although the cases used are different, it is quite remarkable that a single linear model is sufficient to represent accurately the round-off error produced in the simulations. We think that this result can be related to an analogous numerical experiment conducted in [26].

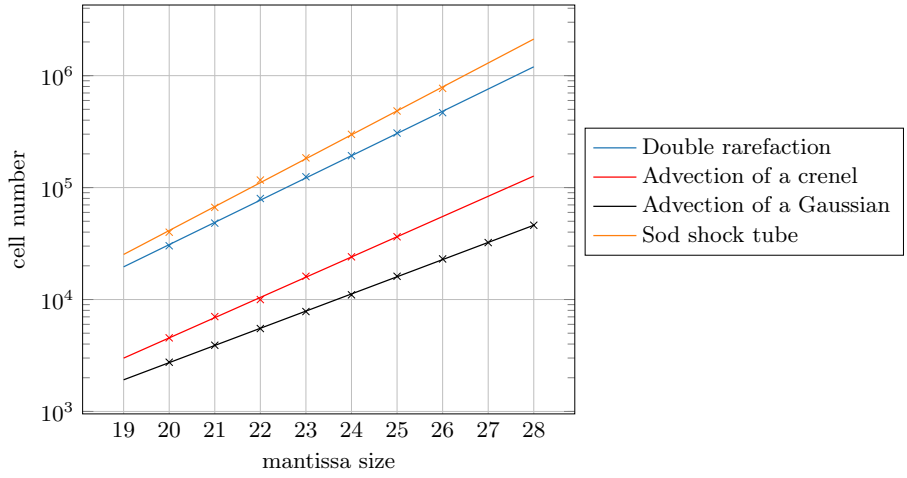

Fig. 6. Number of cells of the divergence point as a function of the mantissa size for four cases. The curves are plotted in logarithmic scale on the y-axis. The lines are obtained by linear regression.

We used the linear model $2^{\alpha m+\beta}$, to extrapolate divergence point of the different test cases for *binary64*. Once the parameters α and β are calibrated by linear regression for each case, the extrapolated point is then simply given by the formula applied with $m = 52$. We conclude that in dimension one, the number of operations needed to make the *binary64* diverge is 220 million cells with 9 billion iterations for the Gaussian advection and 286 billion cells with 251 billion iterations for the Sod shock tube (see table 3). These values are orders of magnitude greater than the capacities of exascale supercomputers.

Table 3. Extrapolation of the number of cells needed to observe the divergence of the *binary64* as a function $2^{\alpha m+\beta}$ with the parameters obtained by linear regression.

test case	number of cells	number of iterations	α
Double rarefaction	6.98×10^{10}	5.75×10^{10}	0.659
Advection of a crenel	2.75×10^{9}	9.82×10^{10}	0.600
Advection of a Gaussian	2.21×10^{8}	8.99×10^{9}	0.509
Sod shock tube	2.86×10^{11}	2.51×10^{11}	0.710

5 Conclusion

The question of round-off errors is of proeminent concern for all HPC simulation codes ([10], [15], [25], etc.). Although the problem is addressed in many textbooks ([13], [1], [16] [17], [2]), an analysis of the impact of finite precision arithmetics on a complete computational code is missing in the literature.

Thanks to the *weak floats* and the performances on large mesh sizes of the massively parallel computational code VARIANT [29], an in-depth study could be conducted. The impact of round-off errors has been analyzed in detail on simulations of ideal gas hydrodynamics in HPC conditions. This study is important as it gives confidence on the future HPC developments. Indeed, the results show that:

- the round-off error is dominated by the scheme error and validate *binary64* on large meshes;
- a linear model is sufficient and surprisingly accurate to predict the rounding error in our simulations.
- the tools and the model developped could determine when it is legitimate to use single precision (*binary32*) in addition to an approach as in [28].

The next step is to get rid of the dependence on the test-case to calibrate the model. Since it is possible to compute bounds on the round-off error on each routine of the code and since these routines are common to the majority of numerical schemes, it is possible to assemble the theoretical bounds to give an estimate of a global bound on the round-off error [3]. Theoretical bounds are wider than empirical bounds but they allow to validate any initial condition including chaotic and turbulent cases.

References

1. Ascher, U.M., Greif, C.: A First Course in Numerical Methods. Society for Industrial and Applied Mathematics, USA (2011)
2. Atkinson, K.E., Han, W., Stewart, D.: Euler's method, chap. 2, pp. 15–36. John Wiley & Sons, Ltd (2011)
3. Boldo, S., Ben Salem-Knapp, L., Weens, W.: Bounding the round-off error of the upwind scheme for advection, submitted
4. 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 (Apr 2013)
5. Boldo, S., Faissole, F., Chapoutot, A.: Round-off error and exceptional behavior analysis of explicit Runge-Kutta methods. IEEE Transactions on Computers (2019)
6. Chapoutot, A., Alexandre dit Sandretto, J., Mullier, O.: Validated Explicit and Implicit Runge-Kutta Methods. In: Small Workshop on Interval Methods. Prague, Czech Republic (Jun 2015)
7. Euler, L.: Principes généraux du mouvement des fluides. Mémoires de l'Académie Royale des Sciences et des Belles Lettres de Berlin **11**, 274–315 (1755)
8. Gautschi, W.: Numerical Analysis: An Introduction. Birkhauser Boston Inc., Cambridge, MA, USA (1997)

9. Godunov, S.K.: Eine Differenzenmethode für die Näherungsberechnung unstetiger Lösungen der hydrodynamischen Gleichungen. Mat. Sb., Nov. Ser. **47**, 271–306 (1959)
10. Harvey, R., Verseghy, D.L.: The reliability of single precision computations in the simulation of deep soil heat diffusion in a land surface model. Climate Dynamics **46**(11), 3865–3882 (Jun 2016)
11. Henrici, P.: Error propagation for difference methods. The SIAM series in applied mathematics, J. Wiley, New York, London (1963)
12. 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 (2 2020)
13. Higham, N.J.: 2. Floating Point Arithmetic, pp. 35–60. SIAM (2002)
14. IEEE: IEEE Standard for Floating-Point Arithmetic. Institute of Electrical and Electronics Engineers IEEE Std 754-2008 pp. 1–70 (Aug 2008)
15. 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)
16. Kahan, W.: Pracniques: Further remarks on reducing truncation errors. Commun. ACM **8**(1), 40– (Jan 1965)
17. Knuth, D.E.: The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Addison-Wesley, Boston, third edn. (1997)
18. Lax, P.D.: Weak solutions of nonlinear hyperbolic equations and their numerical computation. Communications on Pure and Applied Mathematics **7**(1), 159–193 (1954)
19. LeVeque, R.J.: Finite Volume Methods for Hyperbolic Problems. Cambridge Texts in Applied Mathematics, Cambridge University Press (2002)
20. 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), ACM G.1.0; G.1.2; G.4; B.2.0; B.2.4; F.2.1., ISBN 978-3-319-76525-9
21. 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 Basel (Jul 2018)
22. Rusanov, V.V.: The calculation of the interaction of non-stationary shock waves with barriers. Zh. Vychisl. Mat. Mat. Fiz. pp. 267–279 (1961)
23. Alexandre dit Sandretto, J., Chapoutot, A.: Validated Simulation of Differential Algebraic Equations. In: Small Workshop on Interval Methods. Prague, Czech Republic (Jun 2015)
24. Sod, G.A.: A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws. Journal of Computational Physics **27**(1), 1–31 (1978)
25. Spiegel, S.C., Huynh, H., DeBonis, J.R.: A Survey of the Isentropic Euler Vortex Problem using High-Order Methods, chap. 1, p. 1. arc.aiaa.org (2015)
26. Thornes, T., Düben, P., Palmer, T.: A power law for reduced precision at small spatial scales: Experiments with an sqg model. Quarterly Journal of the Royal Meteorological Society **144**(713), 1179–1188 (2018)
27. Toro, E.F.: Riemann Solvers and Numerical Methods for Fluid Dynamics. Springer Berlin Heidelberg„ Berlin, Heidelberg : (2009)
28. Váňa, F., Düben, P., Lang, S., Palmer, T., Leutbecher, M., Salmond, D., Carver, G.: Single precision in weather forecasting models: An evaluation with the ifs. Monthly Weather Review **145**(2), 495 – 502 (01 Feb 2017)

29. 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)