Bounding the Round-Off Error of the Upwind Scheme for Advection

Louise Ben Salem-Knapp [†*], Sylvie Boldo[*], and William Weens[†‡]
[*] Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria, Laboratoire Méthodes Formelles, 91190,
Gif-sur-Yvette, France.
[†] CEA DAM DIF, F-91297, Arpajon Cedex, France
[‡] Université Paris-Saclay, CEA, Laboratoire en Informatique Haute Performance pour le Calcul et la Simulation,
91680 Bruyères-le-Châtel, France

***Abstract*—Numerical simulations are carefully-written programs, and their correctness is based on mathematical results. Nevertheless, those programs rely on floating-point arithmetic and the corresponding round-off errors are often ignored. This article deals with a specific simple scheme applied to advection, that is a particular equation from hydrodynamics dedicated to the transport of a substance. It shows a tight bound on the round-off error of the 1D and 2D upwind scheme, with an error roughly proportional to the number of steps. The error bounds are generic with respect to the format and exceptional behaviors are taken into account. Some experiments give an insight of the quality of the bounds.**

***Index Terms*—Round-off error; Floating-Point; Numerical Scheme; Advection; Hydrodynamics;**

I. Introduction

Physical phenomena are often modelled by differential equations. As there is usually no closed formula for the solution, numerical schemes are used to efficiently get an approximation. The corresponding programs rely on floating-point arithmetic [1]. Even if the mathematics are well studied (order of convergence, stability properties and so on), the round-off errors are often ignored. This may be due to the difficulty of the task, or to the lack of knowledge of computer arithmetic by mathematicians, or to the trust in results obtained with the double precision so far, which starts to erode since HPC entered the game.

We are interested in hydrodynamics, that is to say the study of fluids in motion. This has many applications, such as petroleum through pipelines, aerodynamics or meteorology [2]. There is therefore a large literature of dedicated or applicable schemes [3], [4]. As the simulation expertise grows, it is required not only to provide results, but also to guarantee their quality [5]. We focus here on the upwind scheme, that is a simple finite difference scheme introduced by Courant, Isaacson, and Rees [6], applied to a linear advection equation.

As numerical integration methods are iterative methods, these round-off errors may accumulate and bias the final result. Surprisingly, numerical accuracy problems in computations have been studied even before computers when considering hand calculations [7]. A common claim in mathematics is that the round-off errors "merely accumulate roughly in proportion to the square root of the number of steps in the calculation" [8], but without any proof. This is called "Brouwer's law" and has been studied more deeply by Henrici [9]. Other works dating back from the 60s are compiled in [10], where the author presents "some quite heavy analysis" which has "no practical value". This analysis may seem better than ours as it tackles all linear multistep methods. Unfortunately, the floating-point part is quite naive: the bound does not depend upon the used precision. The local error (the round-off error made at one iteration, same definition as ours in Section III-A) is bounded by a constant times h^{p+1} with p the order of the method and h the time step. With our modern point of view, we are aware that the existence of such constant is not very useful as it might be quite large: for instance, a value such as $2^{2^{100}}$ would mean the round-off error is way beyond any computed value. Moreover, for a reasonable constant, we are convinced that this does not hold in some cases, such as with a limited precision or a large time step. Therefore, we think a more modern analysis with a precise given bound, based on current floating-point knowledge, is valuable.

On the computer science side, automatic methods such as interval arithmetic [11] or even Taylor models [12] do not provide good bounds, as these iterative methods directly re-use the previously-computed value(s). Experiments have shown that the errors are reasonably bounded [13] and that it is possible to use reduced precision to increase computation speed [2], [14], but a lot of work remains to prove it. Bounding one iterative step can be done automatically, but the hard point is the dissemination of the previous errors throughout the iterations. We tackle this point here with first a bound on one step before providing a global bound. We have sought genericity in terms of the floating-point format and the various input values to be applicable in as many cases as possible.

This article is organized as follows. Section II describes the 1D-upwind scheme in details. Section III bounds the round-off error of this scheme and Section IV presents a simple experiment to give an idea of the quality of the bound. Then, Section V tackles the 2D case using the same methodology as before. Section VI concludes.

This work was partly supported by the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation Programme – Grant Agreement n°810367.
This work was partly supported by the NuSCAP project (ANR-20-CE48-0014) of the French national research agency (ANR).

II. About the 1D Upwind Scheme

We will first give some details about the linear advection equation we want to solve, that is to say:

$$\begin{cases} \dfrac{\partial\rho}{\partial t}(x,t) + u\dfrac{\partial\rho}{\partial x}(x,t)=0 \\ \rho(x,0)=\rho_0(x) \\ \rho(0,t)=\rho_l(t) \ , \ \rho(L,t)=\rho_r(t) \end{cases} \tag{E.1}$$

with ρ the density of the fluid and $u \neq 0$ the wave propagation speed in direction x. ρ_0, ρ_r, and ρ_l are known functions used to define the initial condition and the boundary conditions of the problem. This is a very common Initial Boundary Value Problem (IBVP) defined on $[0,L]\times[0,T]$. Applied mathematicians have developed numerous schemes with discretizations of time and space. The chosen discretization is detailed in Section II-A; the upwind scheme is then described in Section II-B and the program in Section II-C.

A. Discretization of the Domain

We discretized the space interval $[0,L]$ and the time interval $[0,T]$ with respectively M and N evenly distributed points. We used $\Delta x = L/M$ as space step size and $\Delta t = T/N$ as time step size. The mesh points are defined by:

$$(x_i,t^n) := (i\Delta x, n\Delta t) \text{ with } i=0,...,M, n=0,...,N.$$

We denote $\rho_i^n \equiv \rho(x_i,t^n)$ the value of the density ρ at point (x_i,t^n) of this regular grid. If we have access to the fluid density at time t^n, then we know the set of discrete values ρ_i^n with $i=0,...,M$. Solving the IBVP Equation (E.1) then requires finding the solution at the next time t^{n+1}, thus determining the set of ρ_i^{n+1} with $i=0,...,M$.

B. The First-Order Upwind Scheme

We study a first-order upwind scheme using one-sided approximation for the spatial derivative of the density.

The key element of the upwind scheme is that the discretization of $\frac{\partial\rho}{\partial x}$ depends upon the sign of the wave propagation speed u.

- If $u>0$, mesh points from the left side are used. The approximation of the spatial derivative at (x_i,t^n) is: $\frac{\partial\rho}{\partial x}(x_i,t^n) = (\rho_i^n - \rho_{i-1}^n)/\Delta x$.
- If $u<0$, mesh points from the right side are used. The approximation of the spatial derivative at (x_i,t^n) is: $\frac{\partial\rho}{\partial x}(x_i,t^n) = (\rho_{i+1}^n - \rho_i^n)/\Delta x$.

For the approximation of the time derivative of the density at (x_i,t^n), we used a first-order forward approximation: $\frac{\partial\rho}{\partial t}(x_i,t^n) = (\rho_i^{n+1}-\rho_i^n)/\Delta t$.

For $u>0$, the first-order upwind scheme is then:
$(\rho_i^{n+1}-\rho_i^n)/\Delta t + u(\rho_i^n-\rho_{i-1}^n)/\Delta x = 0$, therefore

$$\rho_i^{n+1} = \rho_i^n - \frac{u\Delta t}{\Delta x}(\rho_i^n - \rho_{i-1}^n).$$

The first-order upwind scheme with $u>0$ is known to be consistent and stable [3], which means that the density remains bounded, if $\Delta t \leqslant \Delta x/u$. Then the scheme converges to the solution of the linear advection equation (E.1) by the Lax theorem [4].

The first-order upwind scheme can be expressed independently of the sign of the wave propagation speed u:

$$\rho_i^{n+1} = \rho_i^n - K_x^+(\rho_i^n - \rho_{i-1}^n) - K_x^-(\rho_{i+1}^n - \rho_i^n)$$

$$\text{with } K_x^+ := \frac{u^+\Delta t}{\Delta x} \text{ and } u^+ = max(u,0) = \frac{1}{2}(u+|u|)$$

$$K_x^- := \frac{u^-\Delta t}{\Delta x} \text{ and } u^- = min(u,0) = \frac{1}{2}(u-|u|).$$

The stability condition, also known as Courant-Friedrichs-Lewy (CFL) condition, of the first-order upwind scheme for any wave propagation speed $u\neq 0$ is then: $\Delta t \leqslant \Delta x/|u|$.

In the implementation of the scheme, we cannot use the expression $\Delta t = T/N$ for any number of time steps N. We must choose the time step size to satisfy the stability condition, so we set $\Delta t = 0.9\Delta x/|u|$ and we deduce the number of time steps N from the time step size Δt.

C. Pseudo-Code of the 1D Upwind Scheme

Algorithm 1 Pseudo code of 1D upwind scheme

```
$K_x \leftarrow u\frac{\Delta t}{\Delta x}$

// initialization of $\rho^0[i]$
$t \leftarrow 0$
while $t < T$ do
    $t \leftarrow t + \Delta t$
    for each mesh point $i$ do
        $\rho^{n+1}[i] \leftarrow \rho^n[i] - max(0,K_x) * (\rho^n[i] - \rho^n[i-1])$
                    $-min(K_x,0) * (\rho^n[i+1] - \rho^n[i])$
    end for
end while
```

The implementation of the upwind scheme is given in Algorithm 1. Note that a first floating-point problem may appear in the loop condition. Indeed, if Δt is very small compared to t, then the loop may never stop [15]. Nevertheless, this can be easily checked beforehand. An assertion before the loop checking that $T \oplus \Delta t \neq T$ is sufficient to guarantee the termination of the program. Indeed, if $T\oplus\Delta t \neq T$, it means that all the $t\oplus\Delta t$ will be greater than t.

Of course, we would have preferred the program to loop over an integer n and to compute $t = n\otimes\Delta t$ as it makes more sense from a computer arithmetic point of view. But we decided to ensure the program is correct as it is written for varying time steps, and not as we would have like it to be written.

III. Bound on the Round-Off Error of the 1D Upwind Scheme

Now let us focus on the round-off error of the 1D upwind scheme described above. Section III-A describes the working hypotheses. Section III-B bounds the local error (made during one iteration). Section III-C bounds the global error.

In the following, we will consider $u\geqslant 0$ and therefore $K_x^- = 0$ and $K_x^+ = K_x \geqslant 0$. It is easier to handle this case first. The negative case is studied in Section III-D.

We only consider *absolute* error bounds. It may seem silly as relative error bounds (such as the standard model *à la* Higham) provide better bounds. Nevertheless, we want to bound the global error of the scheme and we were only able to handle the iteration from local to global errors with an absolute bound. As we obtain a tight (linear) error bound at the end, we have kept going with absolute bounds.

A. *Hypotheses and Notations*

As explained, we first study the case $u \geqslant 0$, that is to say the following scheme: $\rho_i^{n+1} = \rho_i^n - K_x \times (\rho_i^n - \rho_{i-1}^n)$.

With floating-point arithmetic operations, it becomes:

$$\widehat{\rho}_i^{n+1} = \widehat{\rho}_i^n \ominus \widehat{K_x} \odot (\widehat{\rho}_i^n \ominus \widehat{\rho}_{i-1}^n)$$
$$\text{with } \widehat{K_x} = u \odot \Delta t \oslash \Delta x.$$

We assume a generic floating-point format with p bits of precision. For exceptional behavior, we consider a minimal exponent $e_{\min}$ and a maximal exponent $e_{\max}$ as in the IEEE-754 standard [1]. In particular, the smallest positive floating-point number is the subnormal $2^{e_{\min}-p+1}$, the smallest positive normal is $2^{e_{\min}}$, and the largest floating-point number is $2^{e_{\max}}\left(1-2^{1-p}\right)$. We assume correct rounding for all basic operators, such as $\ominus$ and $\oslash$. For the sake of readability, we denote approximated values with a hat such as in $\widehat{\rho}_i^n$.

First, we assume usual mathematics hypotheses related to this scheme. In particular the stability condition, so $K_x \leqslant 1$.

But $\widehat{K_x}$ is computed from previous values, so it may be slightly off. Nevertheless, if $\widehat{K_x} > 1$, we will have numeric problems, so we assume:

$$\widehat{K_x} \leqslant 1. \qquad \text{(H.2)}$$

Note this can also be tested at the beginning of the program. Still about $\widehat{K_x}$, we need a notation to bound its round-off error:

$$|\widehat{K_x} - K_x| \leqslant C_k \times 2^{-p}.$$

We will then assume

$$C_k \leqslant 9. \qquad \text{(H.3)}$$

The value 9 is needed in the proof of Theorem 1 in Section III-C. In practice, we compute K_x by: $\Delta x \leftarrow L/M$, then $\Delta t \leftarrow 0.9 * \Delta x/|u|$, and then $K_x \leftarrow u\Delta t/\Delta x$. We deduce that $C_k \leqslant 8$ (even for a more generic value than 0.9). But having a generic C_k may encompass other algorithms.

The global round-off error ε_i^{n+1} of ρ_i^{n+1} is defined by:

$$\varepsilon_i^{n+1} = \widehat{\rho}_i^{n+1} - \rho_i^{n+1}.$$

Its bounding is addressed in Section III-C.

For that, we first bound in Section III-B, the local round-off error of ρ_i^{n+1}, denoted by δ_i^{n+1}, and defined by:

$$\delta_i^{n+1} = \widehat{\rho}_i^{n+1} - \left(\widehat{\rho}_i^n - K_x \times (\widehat{\rho}_i^n - \widehat{\rho}_{i-1}^n)\right).$$

As explained before, the bound we consider in this article is an *absolute* bound, as it proved more convenient to handle in the iteration from local to global bounds of Section III-C. We therefore need an absolute bound on the maximal value of ρ, in order to have an absolute bound on the maximal value of $\widehat{\rho}$. Both will be roughly bounded by 2^e. That is to say, from the input values and initializations, we choose e such that the values of interest will be bounded by 2^e. Of course, we may choose a very large e but the error would then be loosely bounded. So e is to be chosen as small as possible.

Our goal is that $|\widehat{\rho}_i^n| < 2^e$ holds at any time and space step. For that, we will bound the exact $|\rho_i^n|$ by a value near but smaller than 2^e and prove by induction that the round-off error does never make it greater than 2^e. We have a kind of loose bound on the round-off error, that will be guaranteed step by step. More precisely, it is an absolute error bound of $2^{e-\alpha}$, which corresponds to a relative error bound of $2^{-\alpha}$ for the largest value, with α a constant defined later. For instance, we may take $\alpha = 10$ in *binary64* to ensure a relative error of 2^{-10} on the largest values of the scheme. Therefore, we need the exact value ρ to be small enough:

$$\forall i, n, \quad |\rho_i^n| \leqslant 2^e(1-2^{-\alpha}). \qquad \text{(H.4)}$$

In order to handle exceptional behaviors (underflow and overflow), we require e to be reasonable:

$$e_{\min} \leqslant e \leqslant e_{\max} - 2 \qquad \text{(H.5)}$$

Moreover, we require the initial error ε_i^0 to be bounded by a constant that does not depend on i:

$$\forall i, \quad |\varepsilon_i^0| \leqslant 2^{e-\alpha_0} \leqslant 2^{e-\alpha-2}. \qquad \text{(H.6)}$$

For instance, α_0 is 24 in our experiments in *binary32*. A bound on the maximal number of iterations N will be required later.

B. *Bound on the Local Round-Off Error*

Let us first focus on δ. There is no novelty here, but plain forward error analysis. For an analysis only in *binary32*, we could have relied on the Gappa tool [16]. As both p and e are variables, we are left with pen and paper proofs.

For the sake of readability, we use the notations of Table I for intermediate variables a_1, a_2, a_3 (both computed and exact).

Using these notations, it remains to bound the local round-off error, that is also equal to: $\delta_i^{n+1} = \widehat{\rho}_i^{n+1} - a_3$.

To bound δ, we add an additional hypothesis only in this section: $|\widehat{\rho}_i^n| < 2^e$. This is assumed here, but it will be proved by induction in Theorem 1 using Hypothesis H.4.

a) About a_1: First, let us bound $a_1 = \widehat{\rho}_i^n - \widehat{\rho}_{i-1}^n$. As $|\widehat{\rho}_i^n| < 2^e$ is an FP number, it means that $|\widehat{\rho}_i^n| \leqslant \mathrm{pred}(2^e)$, and similarly for $\widehat{\rho}_{i-1}^n$. Therefore $|a_1| \leqslant 2\,\mathrm{pred}(2^e) = \mathrm{pred}(2^{e+1})$, so we have $|\widehat{a_1}| \leqslant \mathrm{pred}(2^{e+1}) < 2^{e+1}$.

Last, we easily bound the error between a_1 and $\widehat{a_1}$:

$$\begin{aligned} |\widehat{a_1} - a_1| &= |\widehat{\rho}_i^n \ominus \widehat{\rho}_{i-1}^n - (\widehat{\rho}_i^n - \widehat{\rho}_{i-1}^n)| \\ &\leqslant \frac{1}{2}\mathrm{ulp}(a_1) \leqslant 2^{e-p}. \end{aligned}$$

b) About a_2: As $|\widehat{K_x} \times \widehat{a_1}| \leqslant |\widehat{a_1}| \leqslant \mathrm{pred}(2^{e+1})$, we also have that $|\widehat{a_2}| = |\widehat{K_x} \odot \widehat{a_1}| \leqslant \circ\left(\mathrm{pred}(2^{e+1})\right) = \mathrm{pred}(2^{e+1})$ by monotony of rounding.

TABLE I
NOTATIONS FOR THE 1D UPWIND SCHEME LOCAL ERROR.

	a_1	a_2	a_3
a_i	$\widehat{\rho}_i^n - \widehat{\rho}_{i-1}^n$	$K_x \times a_1 =$ $K_x \times (\widehat{\rho}_i^n - \widehat{\rho}_{i-1}^n)$	$\widehat{\rho}_i^n - a_2 =$ $\widehat{\rho}_i^n - K_x \times (\widehat{\rho}_i^n - \widehat{\rho}_{i-1}^n)$
$\widehat{a_i}$	$\widehat{\rho}_i^n \ominus \widehat{\rho}_{i-1}^n$	$\widehat{K_x} \odot \widehat{a_1} =$ $\widehat{K_x} \odot (\widehat{\rho}_i^n \ominus \widehat{\rho}_{i-1}^n)$	$\widehat{a_3} = \widehat{\rho}_i^{n+1}$
$\lvert\widehat{a_i}\rvert \leqslant$	$\mathrm{pred}(2^{e+1})$	$\leqslant \mathrm{pred}(2^{e+1})$	
$\lvert a_i - \widehat{a_i}\rvert \leqslant$	2^{e-p}	$2^{e-p} \times (1 + 2C_k + K_x)$	

Using the results on $\widehat{a_1}$, we then bound the error on $\widehat{a_2}$:

$$\begin{aligned} |\widehat{a_2} - a_2| &= |\widehat{K_x} \odot \widehat{a_1} - K_x \times a_1| \\ &\leqslant |\widehat{K_x} \odot \widehat{a_1} - \widehat{K_x} \times \widehat{a_1}| + |\widehat{K_x} \times \widehat{a_1} - K_x \times \widehat{a_1}| \\ &\quad + |K_x \times \widehat{a_1} - K_x \times a_1| \\ &\leqslant \frac{1}{2}\mathrm{ulp}(\widehat{K_x} \times \widehat{a_1}) + |\widehat{K_x} - K_x| \times |\widehat{a_1}| \\ &\quad + K_x \times |\widehat{a_1} - a_1|. \end{aligned}$$

As $|\widehat{K_x} \times \widehat{a_1}| \leqslant |\widehat{a_1}| \leqslant \mathrm{pred}(2^{e+1})$, we can finally bound the error between a_2 and $\widehat{a_2}$:

$$\begin{aligned} |\widehat{a_2} - a_2| &\leqslant 2^{e-p} + C_k \times 2^{-p} \times 2^{e+1} + K_x \times 2^{e-p} \\ &\leqslant 2^{e-p} \times (1 + 2C_k + K_x). \end{aligned}$$

c) About a_3 and the local error: Let us now bound $\delta_i^{n+1} = \widehat{\rho}_i^{n+1} - a_3$ using the previous bounds:

$$\begin{aligned} |\delta_k^{n+1}| &= |\widehat{\rho}_i^n \ominus \widehat{a_2} + \widehat{a_2} - \widehat{a_2} - (\widehat{\rho}_i^n - a_2)| \\ &\leqslant |\widehat{\rho}_i^n \ominus \widehat{a_2} - (\widehat{\rho}_i^n - \widehat{a_2})| + |\widehat{a_2} - a_2| \\ &\leqslant \frac{1}{2}\mathrm{ulp}(\widehat{\rho}_i^n - \widehat{a_2}) + 2^{e-p} \times (1 + 2C_k + K_x). \end{aligned}$$

We use: $|\widehat{\rho}_i^n - \widehat{a_2}| \leqslant |\widehat{\rho}_i^n| + |\widehat{a_2}| \leqslant 2^e + 2^{e+1} = 3 \times 2^e$.
We finally bound the local round-off error:

$$\begin{aligned} |\delta_i^{n+1}| &\leqslant \frac{1}{2}\mathrm{ulp}(3 \times 2^e) + 2^{e-p} \times (1 + 2C_k + K_x) \\ &\leqslant 2^{e-p+1} + 2^{e-p} \times (1 + 2C_k + K_x) \\ &\leqslant 2^{e-p} \times (3 + 2C_k + K_x). \end{aligned}$$

Note that this bound does not depend upon either n or i. It is a simple constant depending on the previously-assumed bounds on the values and the input errors. We also remind the reader that this bound holds assuming $|\widehat{\rho}_i^n| < 2^e$ for all i.

Note that the use of Gappa [16] for *binary32* gives us a similar bound under the same hypotheses: 22×2^{-24} while ours is $2^{e-p} \times (3 + 2C_k + K_x) \leqslant 22 \times 2^{-24}$.

The preceding proof seems to ignore exceptional behaviors, but they are automatically handled by Hypothesis H.5. As $e_{\min} \leqslant e$, all the values bounding the a_is are normal floating-point numbers, hence the correctness of the various bounds on the ulp. Even if subnormal values are involved, the round-off errors are still bounded as stated above. From the algorithm and Table I, the maximal involved value is 3×2^e. Therefore, the assumption $e \leqslant e_{\max} - 2$ is enough to guarantee that no overflow occurs. No infinity will be generated by both the computation of $\widehat{\rho}_i^n$ and all intermediate computations.

C. Bound on the Global Round-Off Error

Now let us focus on the difficult part of the proof: going from the local to the global error. We will handle the bound on the $\widehat{\rho}$ at the same time.

Theorem 1. *Let us assume Hypotheses H.2, H.3, H.4, H.5, H.6 and $K_x \geqslant 0$. We also assume*

$$N < 2^{p-\alpha-5}. \qquad \text{(H.7)}$$

Then

$$|\varepsilon_i^n| \leqslant 2^{e-\alpha_0} + n \times 2^{e-p} \times (3 + 2C_k + K_x)$$

and

$$|\widehat{\rho}_i^n| < 2^e.$$

Proof. The proof is done by induction.

For $n = 0$, we initially assumed $|\varepsilon_i^0| \leqslant 2^{e-\alpha_0} \leqslant 2^{e-\alpha-2}$ in Hypothesis H.6. Moreover, $|\widehat{\rho}_i^0| \leqslant |\rho_i^0| + 2^{e-\alpha_0} \leqslant 2^e(1 - 2^{-\alpha}) + 2^{e-\alpha-2} < 2^e$.

We assume the various hypotheses and the induction that provides bounds on $|\varepsilon_i^n|$ and $|\widehat{\rho}_i^n|$ and we will prove those bounds at time step $n+1$.

Let us first express ε_i^{n+1} from the previous ε_j^n. As $\widehat{\rho}_i^{n+1} = \widehat{\rho}_i^n \ominus \widehat{K_x} \odot (\widehat{\rho}_i^n \ominus \widehat{\rho}_{i-1}^n)$, we easily get

$$\varepsilon_i^{n+1} = (1 - K_x) \times \varepsilon_i^n + K_x \times \varepsilon_{i-1}^n + \delta_i^{n+1}.$$

From the induction hypothesis, we have the bound on $|\widehat{\rho}_i^n|$, so the results of Section III-B about the local round-off error hold. Therefore,

$$\begin{aligned} |\varepsilon_i^{n+1}| &\leqslant |1 - K_x| \times |\varepsilon_i^n| + |K_x| \times |\varepsilon_{i-1}^n| + |\delta_i^{n+1}| \\ &\leqslant (1 - K_x) \times (2^{e-\alpha_0} + n\,2^{e-p}(3 + 2C_k + K_x)) \\ &\quad + K_x \times (2^{e-\alpha_0} + n\,2^{e-p}(3 + 2C_k + K_x)) \\ &\quad + 2^{e-p} \times (3 + 2C_k + K_x) \\ &\leqslant 2^{e-\alpha_0} + n \times 2^{e-p} \times (3 + 2C_k + K_x) \\ &\quad + 2^{e-p} \times (3 + 2C_k + K_x) \\ &\leqslant 2^{e-\alpha_0} + 2^{e-p} \times (n+1) \times (3 + 2C_k + K_x). \end{aligned}$$

There are two things to note in this sequence of inequalities. First, we heavily depend on the sign of K_x and $1 - K_x$ to remove the absolute values. If not, it would not have canceled. Second, the bound on ε_i^n does not depend upon i and we use this fact too. The two facts make the proof work out and we are able to prove a bound for the global round-off error proportional to N. Given the fact that this is an iterative algorithm with N steps, this is quite a tight result.

Last but not least, it remains to prove that $|\widehat{\rho}_i^{n+1}| < 2^e$. As $\widehat{\rho}_i^{n+1} = \varepsilon_i^{n+1} + \rho_i^{n+1}$ by definition and $|\rho_i^{n+1}| \leqslant 2^e(1-2^{-\alpha})$ by Hypothesis H.4, there is only to prove that

$$|\varepsilon_i^{n+1}| < 2^{e-\alpha}.$$

So let us bound again (less tightly than before) the error $|\varepsilon_i^{n+1}|$ using Hypotheses H.7 and H.3:

$$\begin{aligned} |\varepsilon_i^{n+1}| &\leqslant 2^{e-\alpha_0} + 2^{e-p} \times (n+1) \times (3 + 2C_k + K_x) \\ &\leqslant 2^{e-\alpha-2} + 2^{e-p} \times 2^{p-\alpha-5} \times (3 + 2 \times 9 + 1) \\ &= 2^{e-\alpha-2} + 22 \times 2^{e-\alpha-5} = 30 \times 2^{e-\alpha-5} \\ &< 32 \times 2^{e-\alpha-5} = 2^{e-\alpha}. \end{aligned}$$

Note that the bound 9 on C_k is enough to have the inequality and holds in our case. There is a slight trade-off between the bounds on C_k, on α, and on N. Nevertheless, the provided assumptions are sufficient and reasonable in typical cases.

For instance, in *binary64*, we may assume $\alpha = 10$ and $\alpha_0 \geqslant 12$. That gives us a maximal number of iterations of $2^{38} - 1$. Another example corresponding to the experimental example below in *binary32* is $\alpha = 6$ and $\alpha_0 = 24$. It gives us a maximal number of iterations of 2^{13} for the theorem to hold, even if the bound seems to hold longer, see Section IV.

D. Negative Velocity

We considered $u \geqslant 0$ in Sections III-A, III-B, and III-C. Let us now focus on the case $u \leqslant 0$ and then $K_x \leqslant 0$. Here, the density and its floating-point counterpart are:

$$\begin{aligned} \rho_i^{n+1} &= \rho_i^n - K_x \times (\rho_{i+1}^n - \rho_i^n), \\ \widehat{\rho}_i^{n+1} &= \widehat{\rho}_i^n \ominus \widehat{K_x} \odot (\widehat{\rho}_{i+1}^n \ominus \widehat{\rho}_i^n). \end{aligned}$$

The global and local round-off errors are defined as before:

$$\begin{aligned} \varepsilon_i^{n+1} &= \widehat{\rho}_i^{n+1} - \rho_i^{n+1}, \\ \delta_i^{n+1} &= \widehat{\rho}_i^{n+1} - (\widehat{\rho}_i^n - K_x \times (\widehat{\rho}_{i+1}^n - \widehat{\rho}_i^n)). \end{aligned}$$

Using δ_i^{n+1}, we can express ε_i^{n+1} as:

$$\varepsilon_i^{n+1} = (1 - |K_x|) \times \varepsilon_i^n + |K_x| \times \varepsilon_{i+1}^n + \delta_i^{n+1}.$$

We can bound δ_i^{n+1} using the same techniques and hypotheses as in Section III-B by introducing the hypothesis $|\widehat{K_x}| \leqslant 1$ and by keeping absolute values on K_x :

$$|\delta_i^{n+1}| \leqslant 2^{e-p} \times (3 + 2C_k + |K_x|).$$

We have the same shape for δ_i^{n+1} and ε_i^{n+1} as in the case $u \geqslant 0$, so we deduce the same bound on the global round-off error with $u \leqslant 0$:

$$|\varepsilon_i^{n+1}| \leqslant 2^{e-\alpha_0} + 2^{e-p} \times (n+1) \times (3 + 2C_k + |K_x|).$$

IV. Experimentation

In order to illustrate Theorem 1, we show here a numerical experimentation to give an insight on the quality of our bound.

We simulated the advection of a density defined on the domain $[0, L] \times [0, T] = [0, 1] \times [0, 4]$ having a gaussian shape, defined by $\rho^0[i] \leftarrow 0.875 * e^{-100(x_i - 0.5)^2} + 0.1$. We used periodic boundary conditions and a constant wave propagation speed $u = 1$. The number of mesh points is $1/\Delta x = 0.9N/(T|u|)$ and is then proportional to N.

Studying the error between the exact solution and the computed solution produced by solving a linear advection equation is quite easy because the solution of this problem is known. Given our domain and speed, the solution of the problem of Equation (E.1) is exactly the initial density, that is to say the gaussian shape.

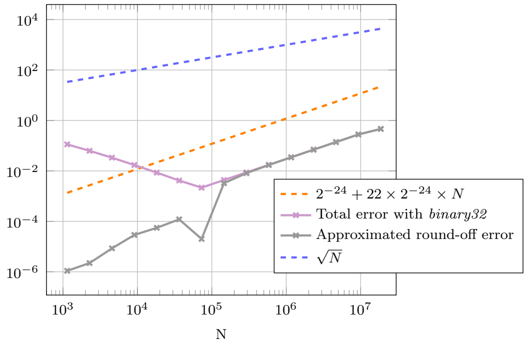
Plotting exactly the round-off error is a difficult task that requires multi-precision that may be slow. We are more interested in the variation of the round-off error, and in particular if it increases linearly (proportional to N) as stated by Theorem 1 or proportional to $\sqrt{N}$ as in Brouwer's law. Experimenting with $N \approx 10^6$ was however out of question with a large p. So we choose to consider the *binary64* computation as "correct" and we plot an approximate round-off error as the difference between the *binary64* and the *binary32* computations.

We used the following parameters: $p = 24$, $e = 0$ since the maximum of gaussian density of our experiment is lower than 1 by definition, $K_x = 0.9$, $C_k = 8$, and $\alpha = 6$ to respect H.4. Moreover the exact initial density studied is the gaussian computed in *binary64*. The error is then only the cast of this value to *binary32*, hence $\alpha_0 = 24$.

With those parameters, our results are true only if the number of iterations satisfied H.7, then if $N < 8192$. But we can see that the bound of the round-off error $2^{e-\alpha_0} + 2^{e-p} \times N \times (3 + 2C_k + K_x)$ still seems correct after 8192 iterations. This is because our theorem overestimates the round-off error. Indeed, the assumption on N is here to ensure that the round-off error is smaller than $2^{e-\alpha} = 2^{-6}$ (so that the computed value is smaller than 1) and this still holds when $N \geqslant 8192$.

We are interested in the error using L^∞ norm in these experimentations to get the maximal round-off error obtained in practice. We simulate the advection of this density using different space step sizes in order to plot the round-off error

Fig. 1. Total error and absolute approximated round-off error of the advection of a gaussian density using the 1D upwind scheme with *binary32*, and the bounds on round-off error.



curve as a function of the number of iterations N. Figure 1 shows both the total error (compared to the mathematical exact solution) and the approximated round-off error (as explained), and it also shows our proved bound of Theorem 1.

As expected, the discretization error is preponderant for small N but gets negligible compared to the round-off error when N grows. There is a sweet spot around $N \approx 10^5$ where the total error is minimal.

The most important point of Figure 1 is that the increase of the round-off error seems linear. Our bound is of course much larger than the "real" error but our formula seems to fit the growth of the round-off error.

V. The 2D Upwind Scheme and its Round-Off Error Bound

Let us now focus on the same scheme, but extended to dimension two. We will follow the same methodology and prove the same kind of bound. Section V-A presents the partial differential equation, the scheme, and the program. Section V-B presents the notations for local and global error, and some preliminary properties. Section V-C bounds the local error and Section V-D gives the final theorem bounding the global error. Section V-E generalizes to other signs as we assumed a non-negative velocity.

A. About the 2D Upwind Scheme

With u_x and u_y the components of the wave propagation speed vector, the Initial Boundary Value Problem in two dimensions is defined on the domain $[0, L_x] \times [0, L_y] \times [0, T]$ and is written:

$$\begin{cases} \dfrac{\partial \rho}{\partial t}(x,y,t) + u_x \dfrac{\partial \rho}{\partial x}(x,y,t) + u_y \dfrac{\partial \rho}{\partial y}(x,y,t) = 0 \\ \rho(x,y,0) = \rho_0(x,y) \\ \rho(0,y,t) = \rho_{l_x}(y,t) \text{ , } \rho(L_x,y,t) = \rho_{r_x}(y,t) \\ \rho(x,0,t) = \rho_{l_y}(x,t) \text{ , } \rho(x,L_y,t) = \rho_{r_y}(x,t) \end{cases} \tag{8}$$

As in Section II-A, we discretized each interval of the domain with evenly distributed points: the space intervals $[0, L_x]$ and $[0, L_y]$ with respectively M and P points using $\Delta x = L_x/M$ and $\Delta y = L_y/P$ as space step size. Mesh points are defined by: $(x_i, y_j, t^n) := (i\Delta x, j\Delta y, n\Delta t)$ with $i, j, n \geqslant 0,$ and $i \leqslant M, j \leqslant P,$ and $n \leqslant N$.

We denote $\rho^n_{i,j} \equiv \rho(x_i, y_j, t^n)$ the discrete value of the density ρ at (x_i, y_j, t^n). If the solution of Equation (8) is known at time t^n, solving the problem means finding the solution at time t^{n+1}.

The expression of the first-order upwind scheme in 2D is:

$$\begin{aligned} \rho^{n+1}_{i,j} = \rho^n_{i,j} &- K^+_x(\rho^n_{i,j} - \rho^n_{i-1,j}) - K^-_x(\rho^n_{i+1,j} - \rho^n_{i,j}) \\ &- K^+_y(\rho^n_{i,j} - \rho^n_{i,j-1}) - K^-_y(\rho^n_{i,j+1} - \rho^n_{i,j}) \end{aligned}$$

with $K^+_x := (u^+_x \Delta t)/\Delta x$ and $K^-_x := (u^-_x \Delta t)/\Delta x$, and similar definitions for K^+_y and K^-_y (with u_y and Δy).

The stability condition is similar as the one dimensional one, so we deduce:

$$\left| \frac{u_x \Delta t}{\Delta x} \right| + \left| \frac{u_y \Delta t}{\Delta y} \right| = \Delta t \frac{|u_x|\Delta y + |u_y|\Delta x}{\Delta x \Delta y} \leqslant 1.$$

In order to respect the stability condition, we choose to use in our implementation:

$$\Delta t = 0.9 \frac{\Delta x \Delta y}{|u_x|\Delta y + |u_y|\Delta x}.$$

The pseudo code in 2D is quite similar to Algorithm 1:

Algorithm 2 Pseudo code of 2D upwind scheme

```
$K_x, K_y \leftarrow u_x \frac{\Delta t}{\Delta x}, u_y \frac{\Delta t}{\Delta y}$

// initialization of $\rho^0[i][j]$
$t \leftarrow 0$
while $t < T$ do
  $t \leftarrow t + \Delta t$
  for each mesh point $i$,$j$ do
    $\rho^{n+1}[i][j] \leftarrow \rho^n[i][j]$
                  $-max(0, K_x) * (\rho^n[i][j] - \rho^n[i-1][j])$
                  $-min(K_x, 0) * (\rho^n[i+1][j] - \rho^n[i][j])$
                  $-max(0, K_y) * (\rho^n[i][j] - \rho^n[i][j-1])$
                  $-min(K_y, 0) * (\rho^n[i][j+1] - \rho^n[i][j])$
  end for
end while
```

B. Hypotheses and Notations

As before, we begin by choosing the signs of $K_x, K_y \geqslant 0$. The other cases will be similar and treated in Section V-E.

The scheme is then:

$$\rho^{n+1}_{i,j} = \rho^n_{i,j} - K_x \times (\rho^n_{i,j} - \rho^n_{i-1,j}) - K_y \times (\rho^n_{i,j} - \rho^n_{i,j-1}).$$

And its floating-point counterpart is:

$$\widehat{\rho}^{n+1}_{i,j} = \widehat{\rho}^n_{i,j} \ominus \widehat{K_x} \odot (\widehat{\rho}^n_{i,j} \ominus \widehat{\rho}^n_{i-1,j}) \ominus K_y \odot (\widehat{\rho}^n_{i,j} \ominus \widehat{\rho}^n_{i,j-1}).$$

The global and local round-off errors are respectively defined as in one dimension:

$$\begin{aligned} \varepsilon^{n+1}_{i,j} &= \widehat{\rho}^{n+1}_{i,j} - \rho^{n+1}_{i,j}, \\ \delta^{n+1}_{i,j} &= \widehat{\rho}^{n+1}_{i,j} - (\widehat{\rho}^n_{i,j} - K_x \times (\widehat{\rho}^n_{i,j} - \widehat{\rho}^n_{i-1,j}) \\ &\qquad - K_y \times (\widehat{\rho}^n_{i,j} - \widehat{\rho}^n_{i,j-1})). \end{aligned}$$

By algebraic equalities, we end up with

$$\varepsilon^{n+1}_{i,j} = \delta^{n+1}_{i,j} + \varepsilon^n_{i,j} \times (1 - K_x - K_y) + K_x \varepsilon^n_{i-1,j} + K_y \varepsilon^n_{i,j-1}.$$

There is one term in this equality that may seem frightening: $\varepsilon^n_{i,j} \times (1 - K_x - K_y)$. In the 1D case, the preceding error was multiplied by K_x and $1 - K_x$, both smaller than 1. Here in two-dimension, the value $1 - K_x - K_y$ does not obviously seem to be in $[0, 1]$. But mathematical properties ensure this. Assuming $u_x, u_y \geqslant 0$, we have

$$\begin{aligned} K_x &= u_x \frac{\Delta t}{\Delta x} = 0.9 \frac{u_x \Delta y}{|u_x|\Delta y + |u_y|\Delta x}, \\ K_y &= u_y \frac{\Delta t}{\Delta y} = 0.9 \frac{u_y \Delta x}{|u_x|\Delta y + |u_y|\Delta x}, \\ \text{then } K_x + K_y &= 0.9 \frac{u_x \Delta y + u_y \Delta x}{|u_x|\Delta y + |u_y|\Delta x} = 0.9. \end{aligned}$$

TABLE II
NOTATIONS 2D

	a_1	a_2	a_3	a_4
a_i	$\widehat{\rho}^n_{i,j} - \widehat{\rho}^n_{i-1,j}$	$K_x \times a_1$	$\widehat{\rho}^n_{i,j} - a_2$	$K_y \times (\widehat{\rho}^n_{i,j} - \widehat{\rho}^n_{i,j-1})$
$\widehat{a_i}$	$\widehat{\rho}^n_{i,j} \ominus \widehat{\rho}^n_{i-1,j}$	$\widehat{K_x} \odot \widehat{a_1}$	$\widehat{\rho}^n_{i,j} \ominus \widehat{a_2}$	$\widehat{K_y} \odot (\widehat{\rho}^n_{i,j} \ominus \widehat{\rho}^n_{i,j-1})$
$\lvert\widehat{a_i}\rvert \leqslant$	2^{e+1}	2^{e+1}	3×2^e	2^{e+1}
$\lvert a_i - \widehat{a_i}\rvert \leqslant$	2^{e-p}	$2^{e-p} \times (1 + 2C_{k_x} + K_x)$	$2^{e-p} \times (3 + 2C_{k_x} + K_x)$	$2^{e-p} \times (1 + 2C_{k_y} + K_y)$

We assume that $\widehat{K_x}$ and $\widehat{K_y}$ are in $[0, 1]$, as in 1D:

$$\begin{aligned} 0 \leqslant \widehat{K_x} \leqslant 1, \\ 0 \leqslant \widehat{K_y} \leqslant 1. \end{aligned} \tag{H.9}$$

Again, this can easily be checked *a priori*.

We also need to bound the errors on $\widehat{K_x}$ and $\widehat{K_y}$:

$$\begin{aligned} |\widehat{K_x} - K_x| \leqslant \quad & C_{k_x} \times 2^{-p} \qquad \text{with} \quad C_{k_x} \leqslant 9, \\ |\widehat{K_y} - K_y| \leqslant \quad & C_{k_y} \times 2^{-p} \qquad \text{with} \quad C_{k_y} \leqslant 9. \end{aligned} \tag{H.10}$$

The same algorithm as in 1D is used for computing $\widehat{K_x}$ and $\widehat{K_y}$, hence the similar bound.

In order to handle exceptional behaviors (underflow and overflow), we require the same assumption on e as in 1D:

$$e_{\min} \leqslant e \leqslant e_{\max} - 2 \tag{H.11}$$

C. Bound on the Local Round-Off Error

As before, we will ensure that $|\widehat{\rho}^n_{i,j}| < 2^e$ for a given e. For that, we have the same kind of hypotheses as before on the exact values taken by ρ:

$$\forall i, j, n, \quad |\rho^n_{i,j}| \leqslant 2^e(1 - 2^{-\alpha}) \tag{H.12}$$

for a given α.

Also as before, we list the notations for the local error in Table II. With those, we can rewrite the local round-off error:

$$\delta^{n+1}_{i,j} = \widehat{\rho}^{n+1}_{i j} - (a_3 - a_4) = (\widehat{a_3} \ominus \widehat{a_4}) - (a_3 - a_4).$$

The error bounds are partly the same as in Section III-B. Using the same proofs, we deduce that $|\widehat{a_1} - a_1| \leqslant 2^{e-p}$, that $|\widehat{a_2} - a_2| \leqslant 2^{e-p} \times (1 + 2C_{k_x} + K_x)$, and that $|\widehat{a_3} - a_3| \leqslant 2^{e-p} \times (3 + 2C_{k_x} + K_x)$.

We can now study a_4 and $\widehat{a_4}$ using the same techniques, and we deduce that $|(\widehat{\rho}^n_{i,j} \ominus \widehat{\rho}^n_{i,j-1}) - (\widehat{\rho}^n_{i,j} - \widehat{\rho}^n_{i,j-1})| \leqslant 2^{e-p}$ and that $|\widehat{a_4} - a_4| \leqslant 2^{e-p} \times (1 + 2C_{k_y} + K_y)$.

Finally we can bound the local round-off error:

$$\begin{aligned} |\delta^{n+1}_{i,j}| \quad &= \quad |(\widehat{a_3} \ominus \widehat{a_4}) - (a_3 - a_4)| \\ &= \quad |(\widehat{a_3} \ominus \widehat{a_4}) - (\widehat{a_3} - \widehat{a_4}) + (\widehat{a_3} - \widehat{a_4}) - (a_3 - a_4)| \\ &\leqslant \quad |(\widehat{a_3} \ominus \widehat{a_4}) - (\widehat{a_3} - \widehat{a_4})| + |\widehat{a_3} - a_3| + |\widehat{a_4} - a_4| \\ &\leqslant \quad \frac{1}{2}\mathrm{ulp}(\widehat{a_3} - \widehat{a_4}) + 2^{e-p} \times (3 + 2C_{k_x} + K_x) \\ &\quad\quad + 2^{e-p} \times (1 + 2C_{k_y} + K_y). \end{aligned}$$

To bound $\frac{1}{2}\mathrm{ulp}(\widehat{a_3} - \widehat{a_4})$, we used: $|a_3| \leqslant |\widehat{\rho}^n_{i,j}| + |a_2| \leqslant 2^e + 2^{e+1} = 3 \times 2^e$ and then $|\widehat{a_3}| \leqslant 3 \times 2^e$. And similarly we can bound $\widehat{a_4}$ as $\widehat{a_2}$, which gives $|\widehat{a_4}| \leqslant 2^{e+1}$. Thus:

$$|\widehat{a_3} - \widehat{a_4}| \leqslant 3 \times 2^e + 2^{e+1} = 5 \times 2^e,$$

$$\frac{1}{2}\mathrm{ulp}(\widehat{a_3} - \widehat{a_4}) \leqslant \frac{1}{2}\mathrm{ulp}(5 \times 2^e) \leqslant 2^{e-p+2}.$$

Finally:

$$\begin{aligned} |\delta^{n+1}_{i,j}| \quad &\leqslant \quad 2^{e-p+2} + 2^{e-p} \times (4 + 2(C_{k_x} + C_{k_y}) + K_x + K_y) \\ &\leqslant \quad 2^{e-p} \times (8 + 2(C_{k_x} + C_{k_y}) + K_x + K_y). \end{aligned}$$

This bound does not depend upon either n, i, or j. We also remind the reader that this bound holds assuming $|\widehat{\rho}^n_{i,j}| < 2^e$ for all i and j. As in 1D, the use of Gappa [16] for *binary32* gives us a similar bound.

As before, the exceptional behaviors are automatically handled by Hypothesis H.11. First, the values bounding the a_is are normal floating-point numbers. Second, as seen by the algorithm and Table II, the maximal involved value is again 3×2^e. Therefore, the assumption $e \leqslant e_{\max} - 2$ is enough to guarantee that no overflow occurs.

D. Bound on the Global Round-Off Error

We then prove by induction the bound on ε:

Theorem 2. *Let us assume Hypotheses H.9, H.10, H.11, and H.12. We also assume $N < 2^{p-\alpha-6}$ and $\forall i, j \quad |\varepsilon^0_{i,j}| \leqslant 2^{e-\alpha_0} \leqslant 2^{e-\alpha-2}$. Then*

$$|\varepsilon^n_{i,j}| \leqslant 2^{e-\alpha_0} + n \times 2^{e-p} \times (8 + 2(C_{k_x} + C_{k_y}) + K_x + K_y)$$

and

$$|\widehat{\rho}^n_i| < 2^e.$$

Proof. The proof is done by induction and is trivial for $n = 0$.

The bound on the error is given by the previous equality $\varepsilon^{n+1}_{i,j} = \delta^{n+1}_{i,j} + \varepsilon^n_{i,j} \times (1 - K_x - K_y) + K_x \varepsilon^n_{i-1,j} + K_y \varepsilon^n_{i,j-1}$, the bound on the local round-off error, and the fact that both $1 - K_x - K_y$, K_x, and K_y are non-negative.

The hardest part is, again, to show the bound on $|\widehat{\rho}^{n+1}_i|$ given the various hypotheses. Given Hypothesis (H.12), it is enough to prove that $|\varepsilon^{n+1}_{i,j}| < 2^{e-\alpha}$.

$$\begin{aligned} |\varepsilon^{n+1}_{i,j}| \quad &\leqslant \quad 2^{e-\alpha_0} + n\, 2^{e-p}\,(8 + 2(C_{k_x} + C_{k_y}) + K_x + K_y) \\ &< \quad 2^{e-\alpha-2} + 2^{p-\alpha-6} 2^{e-p} (8 + 2 \times 18 + 2) \\ &= \quad 2^{e-\alpha-2} + 46 \times 2^{e-\alpha-6} = 62 \times 2^{e-\alpha-6} \\ &< \quad 64 \times 2^{e-\alpha-6} = 2^{e-\alpha}. \end{aligned}$$

Compared to 1D, we had to decrease the bound on N as the bound on the local round-off error is slightly larger. For instance, in *binary64*, we may assume $\alpha = 10$ and $\alpha_0 \geqslant 12$. That gives us a maximal number of iterations of $2^{37} - 1$.

E. Other signs for K_x and K_y

If K_x and K_y are negative, the global round-off error has the following form:

$$\varepsilon_{i,j}^{n+1} = \delta_{i,j}^{n+1} + \varepsilon_{i,j}^{n} \times (1 + K_x + K_y) - K_x \times \varepsilon_{i+1,j}^{n} - K_y \times \varepsilon_{i,j+1}^{n}.$$

To be consistent with the bound determined in Section V-D, we need to have $1 + K_x + K_y \geqslant 0$. We can easily verify this hypothesis and then guarantee to find the same bound on the global round-off error:

$$\begin{aligned} K_x + K_y &= 0.9 \frac{u_x \Delta y + u_y \Delta x}{|u_x| \Delta y + |u_y| \Delta x} \\ &= 0.9 \frac{u_x \Delta y + u_y \Delta x}{-u_x \Delta y - u_y \Delta x} = -0.9. \end{aligned}$$

Then $1 + K_x + K_y = 1 - 0.9 = 0.1 \geqslant 0$.

We can verify the same type of hypothesis using K_x and K_y with opposite signs. For example if $K_x \geqslant 0$ and $K_y \leqslant 0$, the global round-off error is :

$$\varepsilon_{i,j}^{n+1} = \delta_{i,j}^{n+1} + \varepsilon_{i,j}^{n} \times (1 - K_x + K_y) + K_x \times \varepsilon_{i-1,j}^{n} - K_y \times \varepsilon_{i,j+1}^{n}.$$

We also check that $1 - K_x + K_y$ is positive:

$$\begin{aligned} K_x - K_y &= 0.9 \frac{u_x \Delta y - u_y \Delta x}{|u_x| \Delta y + |u_y| \Delta x} \\ &= 0.9 \frac{u_x \Delta y - u_y \Delta x}{u_x \Delta y - u_y \Delta x} = 0.9 \leqslant 1. \end{aligned}$$

Except that mathematical lemma that needed its own proof, the FP part is exactly similar and the proofs end up with the same bound on the round-off error of the 2D upwind scheme:

$$|\varepsilon_{i,j}^{n}| \leqslant 2^{e-\alpha_0} + n \times 2^{e-p} \times (8 + 2(C_{k_x} + C_{k_y}) + |K_x| + |K_y|).$$

VI. Conclusion and Perspectives

This article carefully studies a particular scheme for a particular differential equation. It provides a tight bound on the round-off error of the corresponding program, that is proportional to the number of time steps (and not exponential). This was achieved by taking into account dependencies between errors. The idea is to define a local error δ, easily boundable by forward error analysis and handle the recurrence on ε while taking advantage of the mathematical properties. For instance in 1D with $u \geqslant 0$, our definitions are such that: $\varepsilon_i^{n+1} = \delta_i^{n+1} + (1 - K_x) \times \varepsilon_i^n + K_x \times \varepsilon_{i-1}^n$, with a δ easily bounded. As both $K_x \in [0, 1]$ and the bound does not depend upon the chosen space step i, we simplify $(1 - K_x) + K_x$ and get a linear bound. The same idea holds in 2D, taking advantage of similar mathematical inequalities on K_x and K_y. This was the key in order not to get an exponential error bound.

As we have an absolute error bound, exceptional behaviors were easily handled. It is enough to bound e by reasonable values. On the one hand, it implies that there is no overflow (therefore no infinity is generated). On the other hand, our error bound still holds even in case of underflow (at any time or space step). We do not prevent underflow contrary to overflow.

These encouraging results on the advection equation allow us to consider applying this method to the hydrodynamic equations which are made up of three coupled advection equations, one of which is non-linear, and a closing relation which most of the time hides a square root. A tight bound on the round-off error could enable simulations with reduced precision, which would shorten restitution times while guaranteeing the quality of the results; two major points to impact the numerical simulation industry. Nevertheless, to bound this type of system, one must adapt the method to deal with the coupling of the quantities, irregular meshes, non-constant velocities and discontinuous solutions that appear in shocks, and determine bounds on additional routines needed for higher order schemes such as limiters, time integration (Runge-Kutta), Riemann solvers (for Godunov type schemes), etc.

Another perspective is to investigate probabilistic round-off error analysis *à la* Higham and Mary [17]. It is unsure we will get an error proportional to $\sqrt{n}$ as seen by our experiments of Section IV, but the idea is attractive.

Acknowledgments

We are thankful to Guillaume Perrin for his helpful comments and suggestions.

References

[1] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[2] P. D. Dueben, N. Wedi, S. Saarinen, and C. Zeman, "Global simulations of the atmosphere at 1.45 km grid-spacing with the integrated forecasting system," *Journal of the Meteorological Society of Japan. Ser. II*, vol. 98, no. 3, pp. 551–572, 2020.

[3] R. J. Leveque, *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[4] E. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer Berlin Heidelberg, 2009.

[5] M. A. Stadtherr, "High performance computing: Are we just getting wrong answer faster?" *CAST division awards banquet, Miami Beach, Florida*, 1998.

[6] R. Courant, E. Isaacson, and M. Rees, "On the solution of nonlinear hyperbolic differential equations by finite differences," *Communications on Pure and Applied Mathematics*, vol. 5, no. 3, pp. 243–255, 1952.

[7] D. Brouwer, "On the accumulation of errors in numerical integration," *The Astronomical Journal*, vol. 46, pp. 149–153, 1937.

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

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

[10] J. D. Lambert, *Numerical Method for Ordinary Differential Systems*. Wiley, 1991.

[11] "IEEE standard for interval arithmetic," *IEEE Std 1788-2015*, 2015.

[12] J. Alexandre dit Sandretto and A. Chapoutot, "Validated Simulation of Differential Algebraic Equations with Runge-Kutta Methods," *Reliable Computing electronic edition*, vol. 22, Jul. 2016.

[13] W. Weens, "Toward a predictive model to monitor the balance between discretization and rounding errors in hydrodynamic simulations," 2020, siam Conference on Parallel Processing for Scientific Computing.

[14] A. Dawson and P. D. Düben, "rpe v5: an emulator for reduced floating-point precision in large numerical simulations," *Geoscientific Model Development*, vol. 10, no. 6, pp. 2221–2230, 2017.

[15] W. M. Gentleman and S. B. Marovich, "More on algorithms that reveal properties of floating point arithmetic units," *Communications of the ACM*, vol. 17, no. 5, pp. 276–277, 1974.

[16] F. de Dinechin, C. Lauter, and G. Melquiond, "Certifying the floating-point implementation of an elementary function using Gappa," *Transactions on Computers*, vol. 60, no. 2, pp. 242–253, 2011.

[17] N. J. Higham and T. Mary, "A New Approach to Probabilistic Rounding Error Analysis," *SIAM Journal on Scientific Computing*, vol. 41, no. 5, pp. A2815–A2835, 2019.