Journal of Computational Physics **175,** 454–486 (2002)
doi:10.1006/jcph.2001.6952, available online at http://www.idealibrary.com on IDEAL®

High-Order Monotonicity-Preserving Compact Schemes for Linear Scalar Advection on 2-D Irregular Meshes

Quang Huy Tran* and Bruno Scheurer†

*Institut Français du Pétrole, Division Informatique Scientifique et Mathématiques Appliquées, 1 et 4 avenue de Bois Préau, 92852 Rueil-Malmaison Cedex, France; and †Commissariat à l'Energie Atomique, DIF, B.P. 12, 91680 Bruyères-le-Châtel, France
E-mail: Q-Huy.Tran@ifp.fr

Received November 21, 2000; revised October 1, 2001

This paper is concerned with the numerical solution for linear scalar advection problems, the velocity field of which may be uniform or a given function of the space variable. We would like to propose the following: (1) a new family of 1-D compact explicit schemes, which preserve monotonicity while maintaining high-order accuracy in smooth regions; and (2) an extension to the 2-D case of this family of schemes, which ensures good accuracy and isotropy of the computed solution even for very distorted meshes. A few theoretical results are proven, while abundant numerical tests are shown in order to illustrate the quality of the schemes at issue. © 2002 Elsevier Science (USA)

INTRODUCTION

The linear scalar advection equation is usually the starting point for students or research scientists who work in the area of numerical schemes for hyperbolic conservation laws. In the industrial context, it has a critical rôle in fluid mechanics ALE codes such as KIVA [1, 21]. Although many properties are known about this apparently simple topic, the "perfect" scheme for multidimensional advection remains yet to be discovered.

Let us examine the 1-D case. High-resolution upwind schemes based on nonlinear limiting for the equation

$$u_t + au_x = 0 \tag{1}$$

yield excellent results and have matured into a quite robust technological standard. The reader is referred to several papers [2, 13, 37, 41] and review works [4, 16, 38]. However, these schemes generally use a 5-point stencil, which makes it difficult to extend them to the 2-D case. Of course, dimensional splitting could be used, and actually it is rather popular, but this is not we are looking for. Other types of schemes have been suggested for the 1-D advection, for instance, ENO and UNO schemes [19, 20], box schemes [6, 36], discontinuous Galerkin [7], and Iserles schemes [15, 23, 31]. All of these schemes are high order but not necessarily TVD, unless some special treatments are applied.

The situation for the 2-D case is more complex. The advection equation reads

$$u_t + au_x + bu_y = 0, \tag{2}$$

and what we are looking for is a "genuinely multidimensional scheme," that is, a scheme (i) which does not rely on dimensional splitting and (ii) which provides isotropic error distributions. Such a quest is justified by the heavy use of advection steps on irregular meshes in many industrial applications. In such contexts, one is led to the observation that the grid's orientation has a tremendous effect on the quality of the results. As for dimensional splitting, although it remains "the method of choice" for many practioners, it is not convenient for computations on unstructured grids.

The quest may have started when Colella [9] generalized the idea of Godunov's scheme. His scheme, called corner transport upwind (CTU), can be made second order using slope-limitation. However, it does not always ensure high accuracy when the mesh is irregular. An alternative philosophy for securing high order is advocated by Cockburn *et al.* [8, 7] via discontinous Galerkin methods. Unfortunately, these methods turn out to be very expensive. On the other hand, there exist more "compact" stencils, such as in N-schemes [30, 35] or in residual distribution schemes [11, 12, 28]. Nevertheless, all of these schemes have been mostly designed for the steady-state case and so are only of first order for evolution problems.

In this paper, we wish to indicate how 1-D Iserles schemes [23, 31] can be modified to become monotonicity-preserving. Then, we derive a new way of extending these corrected Iserles schemes to the 2-D case. The final product is a family of nonlinear compact stencils, which leads to very satisfactory results for linear advection problems over space-dependent velocity fields.

1. THE 1-D CASE

1.1. Uniform Velocity

Let us consider Eq. (1) in which the velocity a is assumed to be positive and uniform. The initial condition is $u(t = 0; x) = u_0(x)$. The domain $\mathbb{R}$ is divided into intervals of length Δx. First, we concentrate on two Iserles schemes, namely,

$$\texttt{(I1)} \quad u_{i+1}^{n+1} = u_i^{n-1} + (1 - 2\lambda)\left(u_{i+1}^n - u_i^n\right) \tag{3}$$

$$\texttt{(I2)} \quad u_{i+1}^{n+1} = u_i^{n-2} + 2(1 - 3\lambda)\left(u_{i+1}^n - u_i^{n-1}\right) + \frac{(1 - 3\lambda)(1 - 2\lambda)}{1 + \lambda}\left(u_i^n - u_{i+1}^{n-1}\right), \tag{4}$$

with $\lambda = a\Delta t/\Delta x$. Fourier analysis [31] shows that: (i) scheme `(I1)` is second order and is stable for $0 \le \lambda \le 1$; (ii) scheme `(I2)` is fourth order and is stable for $0 \le \lambda \le \frac{1}{2}$. The schemes `(I1)` and `(I2)` are known to be *dissipation-free*. The price to be paid for such a property is a huge amount of *dispersion*.

There is a nice and useful way of interpreting formulae (3) and (4). In the (x, t)-plane of Fig. 1, let us draw the characteristic lines $\mathcal{C}(j, m)$ associated with the *stencil points*

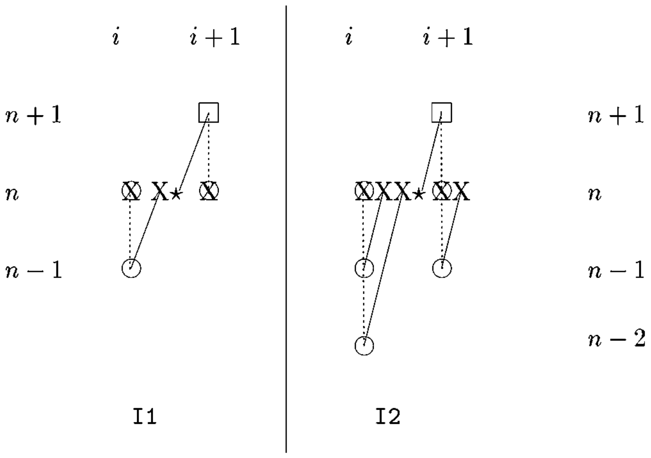

FIG. 1. Iserles stencils.

$\bigcirc = (j, m)$, where $j \in \{i, i+1\}$ is a space-like index and $m \in \{n-2, n-1, n, n+1\}$ is a time-like index. The intersection of these characteristic lines and the axis $t = n\Delta t$ are denoted by $\times = X(j, m)$. The $\times$'s are referred to as *characteristic points*, whether or not they coincide with some stencil points. Of course, u is constant along any characteristic line. A special symbol $\star$ is attributed to $X(i+1, n+1)$, referred to as *spot*, since $(i+1, n+1)$ is the point to be computed. Then, the value of u at the spot $\star$ can be proven to be the outcome of a polynomial interpolation process from the data available at the characteristic points $\times$. The interpolation is quadratic for `I1` and quartic for `I2`. Insofar as the interpolation process does not respect the maximum principle, we can hence understand the origin of dispersion.

Screening out the oscillations is attempted by forcing the new value of u at the spot not to exceed those at the characteristic points $\times$ that are "closest" to the spot $\star$. For instance, in Fig. 1, the characteristic points that "enclose" the spot $\star$ are: $X(i, n-1)$ and $X(i+1, n)$ for the left scheme (`I1`), and $X(i, n-2)$ and $X(i+1, n)$ for the right scheme (`I2`). To better explain how the schemes can be made monotonicity-preserving, let us introduce some more notations.

For two real numbers v and w, regardless of whether $v \leq w$ or $v > w$, we designate by $|v, w|$ the closed interval $[\min(v, w), \max(v, w)]$. Thus, $|v, w| = |w, v|$. If $u \in \mathbb{R}$, we call $\Pi_{|v,w|}(u)$ the *projection* of u onto $|v, w|$, namely,

$$\Pi_{|v,w|}(u) = \begin{cases} \min(v, w) & \text{if } u < \min(v, w) \\ u & \text{if } u \in |v, w| \\ \max(v, w) & \text{if } u > \max(v, w). \end{cases} \tag{5}$$

Intuitively, we can understand the meaning of Π by mentally representing a continuous function $x \mapsto u(x)$. As shown in Fig. 2, if some value $\square$ of u happens to lie outside interval $|v, w|$, we just have to "pull it back." Such a simple-minded limitation is motivated by our desire to avoid creating any new local extremum. In practice, we proceed in two stages: first, we apply the original Iserles formulae to update the grid point, and next, we resort to the projection.

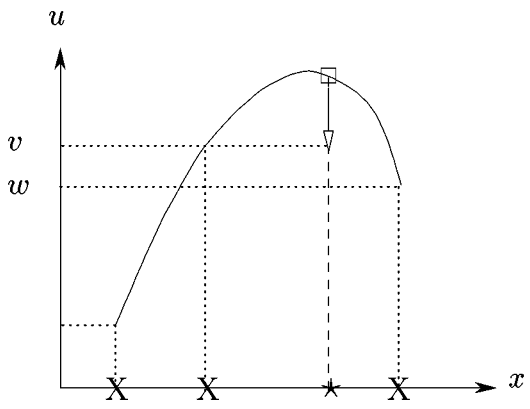

FIG. 2. The idea of projection.

The heart of the matter is to decide on the u-interval over which projection must be carried out right on. This u-interval is $|u(X_l), u(X_r)|$, where $[X_l, X_r]$ is the *bounding box* that encloses the spot $\star$. In the upcoming figures, the bounding box $[X_l, X_r]$ is represented by the rectangle $\boxed{\text{X X}}$. Although the formulae may look exceedingly complicated, the underlying idea is very simple: always search for the characteristic lines which are closest to the spot. This search, of course, will depend on the Courant number λ.

More precisely, the projection process is described as follows:

• For `(I1)`, as depicted in Fig. 3,

$$u_{i+1}^{n+1} := \begin{cases} \Pi_{|u_i^{n-1}, u_{i+1}^n|}\left(u_{i+1}^{n+1}\right) & \text{if } 0 < \lambda < \frac{1}{2} \quad \text{(a)} \\ \Pi_{|u_i^n, u_i^{n-1}|}\left(u_{i+1}^{n+1}\right) & \text{if } \frac{1}{2} < \lambda < 1 \quad \text{(b)}. \end{cases} \tag{6}$$

If $\lambda = \frac{1}{2}$, no projection is required, since $u_{i+1}^{n+1} = u_i^{n-1}$ according to (3).

• For `(I2)`, as depicted in Fig. 4,

$$u_{i+1}^{n+1} := \begin{cases} \Pi_{|u_i^{n-2}, u_{i+1}^n|}\left(u_{i+1}^{n+1}\right) & \text{if } 0 < \lambda < \frac{1}{3} \quad \text{(a)} \\ \Pi_{|u_i^{n-1}, u_i^{n-2}|}\left(u_{i+1}^{n+1}\right) & \text{if } \frac{1}{3} < \lambda < \frac{1}{2} \quad \text{(b)}. \end{cases} \tag{7}$$

For $\lambda = \frac{1}{3}$, no projection is required, since $u_{i+1}^{n+1} = u_i^{n-2}$ according to (4).

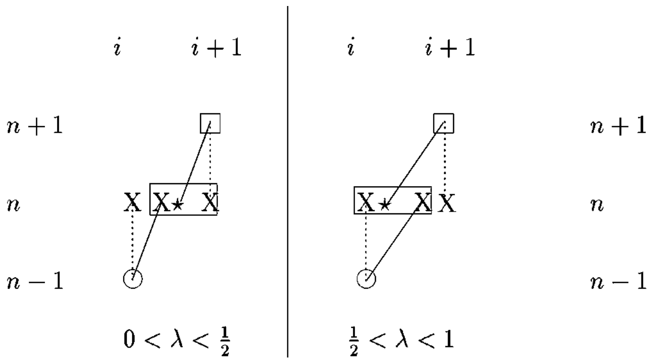

FIG. 3. Choice of the bounding box for `I1`.

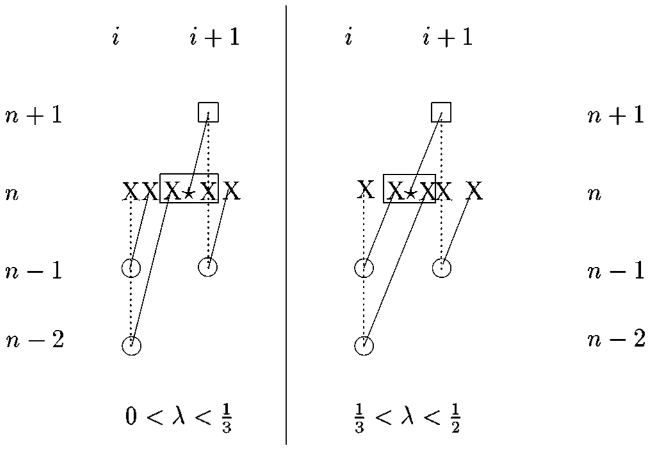

FIG. 4. Choice of the bounding box for `I2`.

The combination of the original schemes `(I1)` and `(I2)` and the projection process (6) and (7) gives pretty good results, as will be shown in the next section. Unfortunately, their quality worsens as λ gets closer to 0. The following observation accounts for this phenomenon, mainly due to the interpolation process. In Fig. 5, it is easy to check that

- if $\lambda < \frac{1}{3}$, then the characteristic point $\diamond = X(i+1, n-1)$ is closer to the spot $\star = X(i+1, n+1)$ than $Y = X(i, n)$;
- if $\lambda < \frac{1}{4}$, then the characteristic point $\diamond = X(i+1, n-2)$ is closer to the spot $\star = X(i+1, n+1)$ than $Y = X(i, n)$.

Henceforth, when $\lambda < \frac{1}{3}$, it is more "interesting" to get $\diamond = X(i+1, n-1)$ involved in the `(I1)` stencil. Thus, to retain an accurate quadratic interpolation, $Y = X(i, n)$ has to be left out. The rationale for this switch in the stencil is to always make use of the "closest" points to the target for interpolating. The corresponding formula is

$$u_{i+1}^{n+1} = \frac{2\lambda^2}{1-\lambda} u_i^{n-1} + \frac{2(1-2\lambda)}{1-\lambda} u_{i+1}^{n} - (1-2\lambda) u_{i+1}^{n-1}. \tag{8}$$

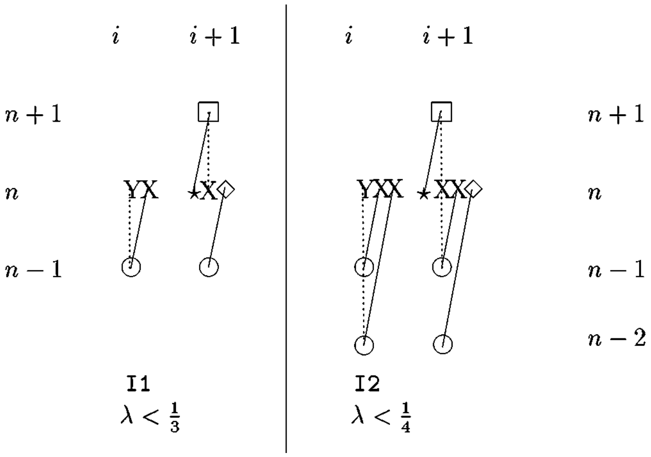

FIG. 5. Modified Iserles schemes for small λ.

Similarly, when $\lambda < \frac{1}{4}$, it is more "accurate" to get $\diamond = X(i+1, n-2)$ involved in the `(I2)` stencil. Again, to keep an accurate quartic interpolation, the point $Y = X(i, n)$ has to be taken out, as has been previously explained and will be evidenced by numerical tests. Thus,

$$u_{i+1}^{n+1} = -\frac{6\lambda^2(1-3\lambda)}{1-\lambda^2}u_i^{n-1} + \frac{6\lambda^2}{1-\lambda}u_i^{n-2} + \frac{3(1-3\lambda)}{1-\lambda}u_{i+1}^{n} - \frac{3(1-3\lambda)(1-2\lambda)}{1-\lambda}u_{i+1}^{n-1} + \frac{(1-2\lambda)(1-3\lambda)}{1+\lambda}u_{i+1}^{n-2}. \tag{9}$$

In summary, we propose two corrected schemes,

$$\begin{aligned}(\texttt{I1C}) &= \{\text{interpolation (3) or (8)}\} + \{\text{projection (6)}\}\\ (\texttt{I2C}) &= \{\text{interpolation (4) or (9)}\} + \{\text{projection (7)}\},\end{aligned}$$

where the interpolation formula depends on the value of λ as indicated above. For $\lambda = \frac{1}{3}$ in `(I1C)` and $\lambda = \frac{1}{4}$ in `(I2C)`, we keep the original stencils, i.e., (3) and (4). Although the switch from (3) to (8) and from (4) to (9) is, in principle, *discontinuous* with respect to λ, the numerical results prove to be very acceptable. In what follows, (8) and (9) will be termed *proximity correction* for the Iserles schemes. It is interesting to note—and easy to prove (see Appendix A)—that both schemes `(I1C)` and `(I2C)` are *monotonicity-preserving*.

To our knowledges, such ideas about (i) projection and (ii) modification of the Iserles schemes have never been put forward. We shall see that these ideas are easily extendable to the 2-D case.

1.2. Variable Velocity

It is relatively straightforward to upgrade the previous schemes to the case of a variable velocity field $a(x)$. However, some basic hypotheses on a will be needed:

(H0) The velocity field a is *continuous* with respect to x. This hypothesis is necessary for existence and uniqueness of the advection problem.

(H1) The values of a are discretized at the nodes i of the grid.

(H2) Every, if any, *source* point of u, i.e., every abscissa s for which $u(s) = 0$ together with $a(s-\epsilon) < 0$ and $a(s+\epsilon) > 0$ for small enough $\epsilon > 0$, must coincide with a node.

Hypothesis (H2) may sound a little too constraining, but the physical explanation is that additional data on u are required at any source point.

We are now in a position to describe our strategy for the variable velocity case. If $a_i > 0$ and $a_{i+1} > 0$, we define an average velocity

$$a := a_{i+1/2} = \frac{1}{2}(a_i + a_{i+1}), \tag{10}$$

and apply `(I1C)` or `(I2C)` with $\lambda = a\Delta t/\Delta x$, along with the appropriate projection step, over the interval $[i, i+1]$ in order to compute u_{i+1}^{n+1}. If $a_i < 0$ and $a_{i+1} < 0$, we exchange i and $i+1$ in the formulae (to take into account the direction of propagation) in order to compute u_i^{n+1}.

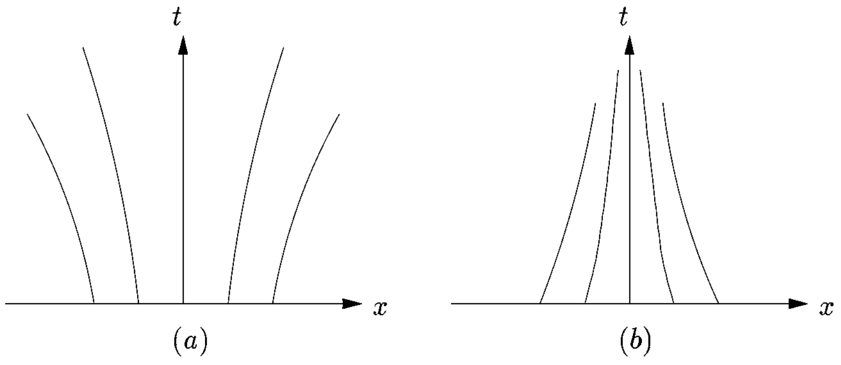

FIG. 6. The two types of sonic points for a continuous velocity field.

It could be thought that for a variable velocity field, more accuracy would be achieved via an affine representation of a. The characteristic curves $\mathcal{C}(j, m)$ are no longer straight lines. Hence, the position of characteristic points $X(j, m)$ must be determined with a more sophisticated procedure before any interpolation process is carried out. We will see from numerical results that the gain in quality is not worth the extra price to be paid.

Furthermore, special attention must be devoted to the treatment of sonic nodes, i.e., any node i where $a_i = 0$. Since $a_i = 0$, the line $x = i\Delta x$ is a characteristic curve. Additionally, since the velocity field a is assumed to be continuous, the other charactistic curves in the neighborhood of i do not cross the line $x = i\Delta x$. Let us distinguish two cases:

- If the sonic point i is a source (Fig. 6a), the value of u_i^{n+1} must be *imposed*[1] as data of the problem. Then, we compute u_{i+1}^{n+1} by using (I1C) or (I2C) with positive velocity over $[i, i+1]$, and u_{i-1}^{n+1} by using the negative-velocity version of the scheme over $[i-1, i]$.
- If the sonic point i is a *sink* (Fig. 6b), we naturally have $u_i^{n+1} = u_i^n$. The value of u_{i+1}^{n+1} is updated by a modified Iserles scheme with negative velocity over $[i+1, i+2]$, whereas the value of u_{i-1}^{n+1} is updated by a modified Iserles scheme with positive velocity over $[i-2, i-1]$.

In both cases, the formulae are used with the average velocity over the interval at issue. Note that, in this paper, only space-dependency of a is considered. The case of a velocity field $a(x, t)$ which depends on space and time is currently under study.

1.3. Numerical Results

Let us start with a uniform velocity field $a = 1$. The discontinuous data SQUARE, defined by

$$u_0(x) = \begin{cases} 1 & \text{if } 0.5 < x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

is advected during a lapse of time $T = 3$. The numerical solution, diplayed for $x \in [3, 4.5]$, is compared to the analytical solution, as well as to a numerical solution computed by a classical MUSCL scheme [41] with the Ultrabee limitation procedure [13]. We recall that Ultrabee makes use of the actual value λ of the CFL ratio. Therefore, it is more compressive

[1]Although this may not be always possible in a "real-life" computation.

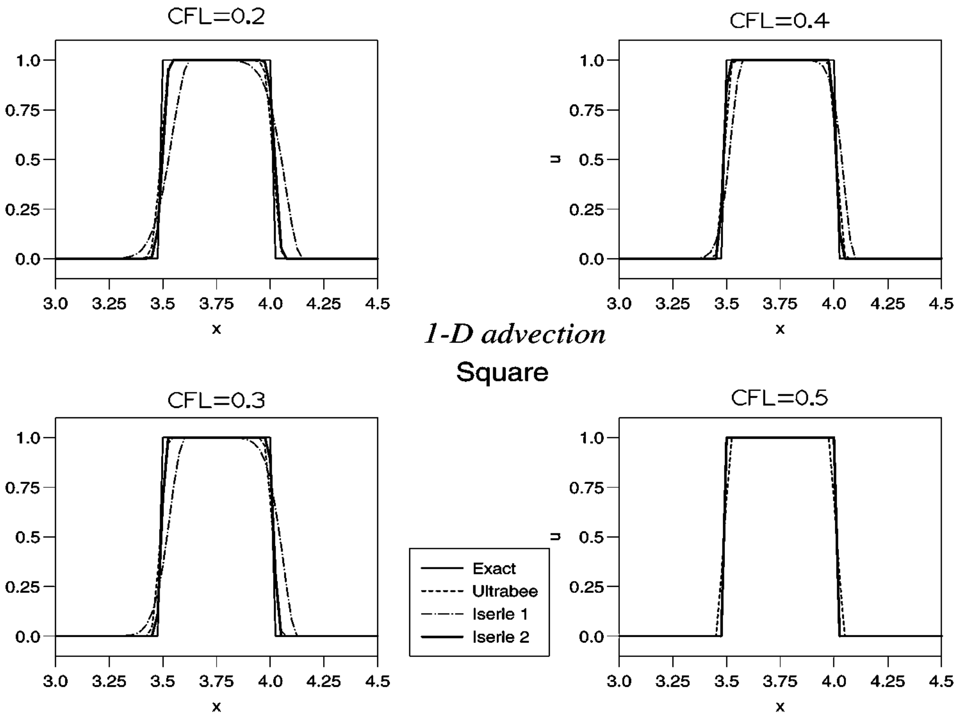

FIG. 7. Advection of SQUARE for large λ, without proximity correction.

than Superbee. For the runs in Figs. 7–9, the mesh spacing is $\Delta x = 0.025$. The first time step is always performed with a standard upwind scheme using no reconstruction. As for the second time step, it is always performed with (I1C).

In Fig. 7, we can see the results corresponding to four different values of λ, ranging from 0.2 to 0.5. Projections (6) and (7) are of course applied, but for the moment, there is no proximity correction introduced in (8) and (9). The shape of the solution obtained with (I1) is unsymmetric and, anyway, too diffusive. Obviously, only (I2) is able to compete with MUSCL. Note that for $\lambda = 0.5$, the two Iserles schemes are exact.

In Fig. 8, the results are associated to smaller values of λ, ranging from 0.05 to 0.3. Although no new local extremum does appear, there is a staircase effect which looks "inaesthetical." If the proximity correction is turned on, the results are much better as shown in Fig. 9, especially for (I2C). This testifies, at least numerically, to the fact that the proximity correction is a necessary step within the whole scheme. This correction will have a fundamental role in variable velocity simulations, where the local CFL may be very close to 0.

Let us quantify the orders of convergence by performing a logarithmic least-squares regression on the relative L^1-errors, computed as functions of the mesh spacing $\Delta x = h$ ranging through {0.05, 0.025, 0.0125, 0.00625}. For this convergence study, in addition to the SQUARE data, we use two other types of initial data, namely,

- a WAVELET

$$u_0(x) = \begin{cases} \cos^2[2\pi(x - 0.75)] & \text{if } 0.5 < x < 1 \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

which is C^1-smooth (but not C^2) and has a peak at $x = 0.75$; and

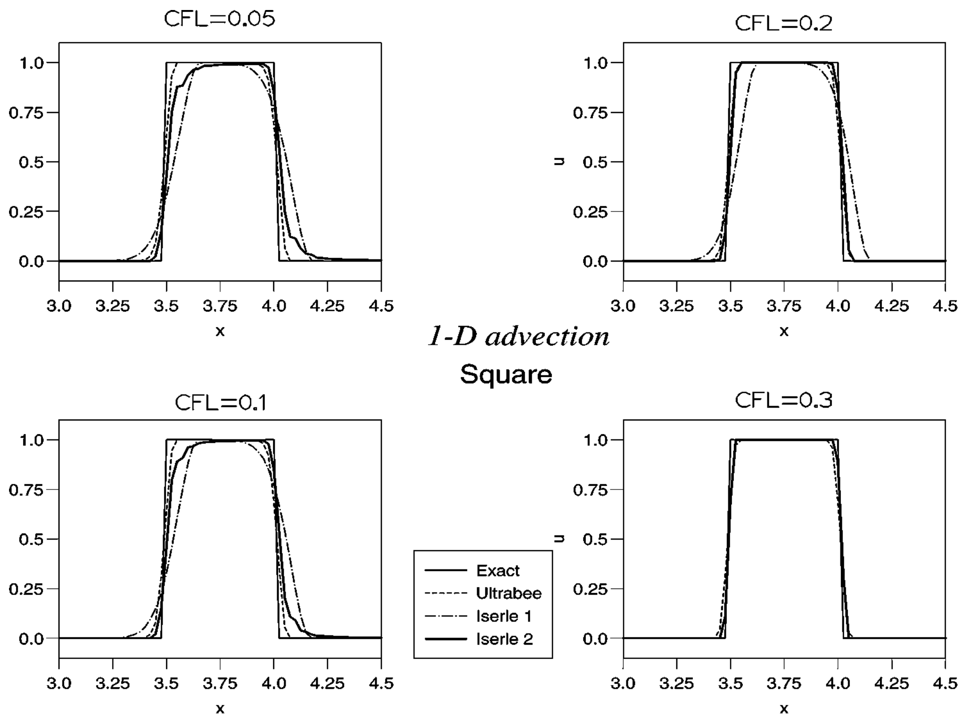

FIG. 8. Advection of SQUARE for small λ, without proximity correction.

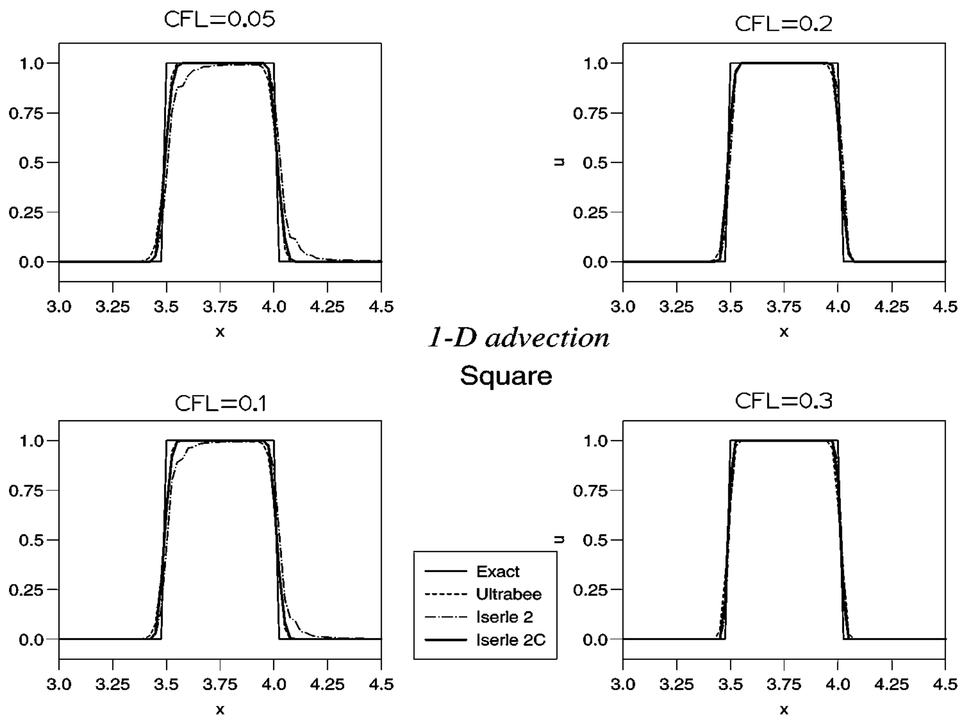

FIG. 9. Advection of SQUARE for small λ, with proximity correction.

- a HYPETAN (standing for HYPErbolic TANgent)

$$u_0(x) = \begin{cases} 0.5\left\{1 + \tanh\left[\frac{x-0.75}{0.25-4(x-0.75)^2}\right]\right\} & \text{if } 0.5 < x < 1 \\ 0 & \text{if } x \leq 0.5 \\ 1 & \text{otherwise,} \end{cases} \tag{13}$$

which is C^∞-smooth. Moreover, it is a monotone-increasing function.

In Fig. 10 and Table I, the word "MUSCL" designates the standard second-order upwind scheme, the limitor being either Ultrabee (for SQUARE) or Van Leer (for WAVELET and HYPETAN). At the right of each entry in Table I, the number in the parentheses corresponds to the limitation-free high-order version of the schemes, i.e., the "LW" (Lax–Wendroff)

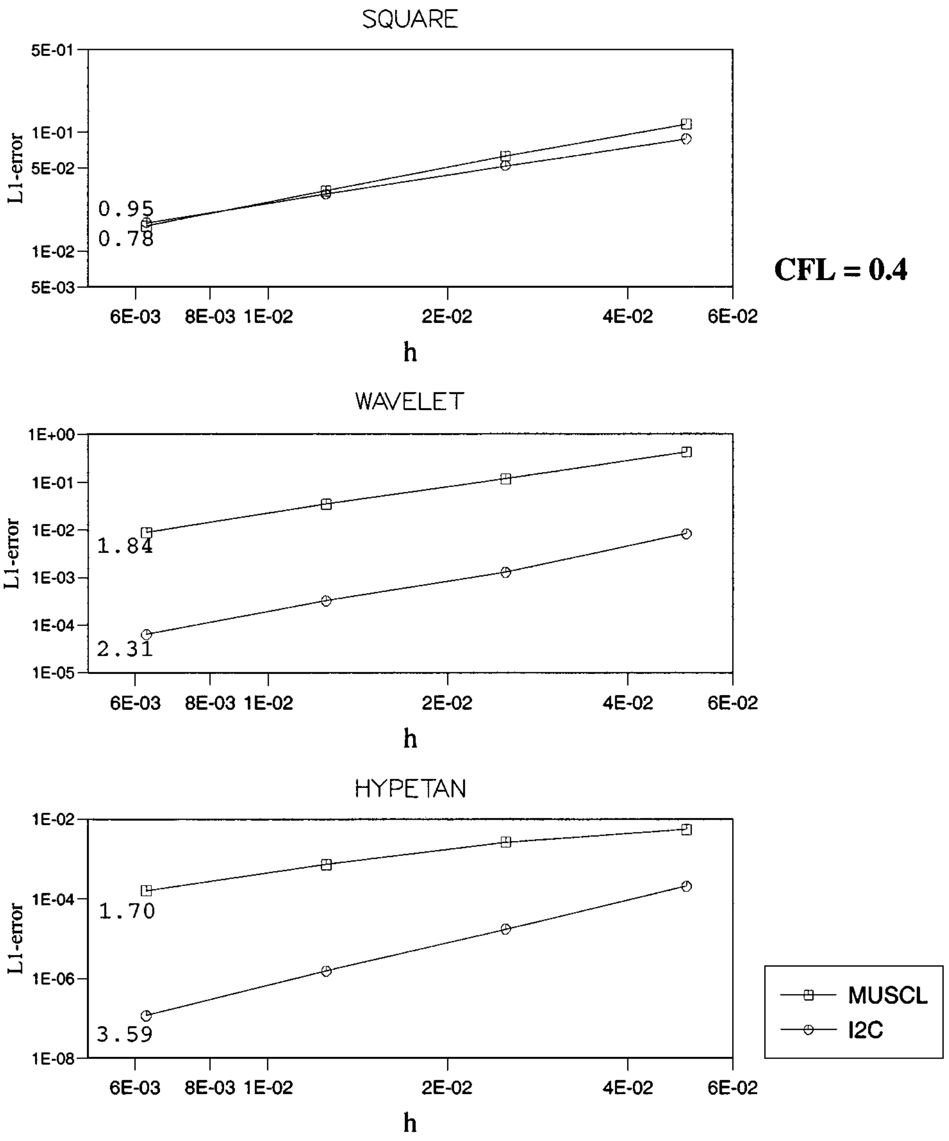

FIG. 10. Convergence of MUSCL and I2C schemes on a logarithmic scale for a 1-D advection experiment.

TABLE I
Orders of Convergence for the 1-D Translation Experiment

	SQUARE		WAVELET		HYPETAN	
CFL	MUSCL (LW)	I2C (I2)	MUSCL (LW)	I2C (I2)	MUSCL (LW)	I2C (I2)
0.8	0.90 (0.74)	N.A.	1.79 (1.73)	N.A.	1.83 (1.87)	N.A.
0.5	0.96 (0.68)	N.A.	1.89 (1.66)	N.A.	1.76 (1.80)	N.A.
0.4	0.95 (0.67)	0.78 (0.50)	1.84 (1.66)	2.31 (2.35)	1.70 (1.79)	3.59 (3.72)
0.2	0.95 (0.64)	0.76 (0.47)	1.69 (1.67)	2.40 (2.24)	1.45 (1.75)	3.16 (3.36)
0.1	0.94 (0.65)	0.77 (0.42)	1.60 (1.70)	2.38 (2.23)	1.30 (1.72)	3.02 (3.17)

second-order scheme and the original fourth–order “I2” (Iserles). This number is supplied simply in order to highlight the effect of the monotonizing mechanisms involved in various schemes. In practical applications of interest for us, LW and I2 cannot be used because of the spurious oscillations generated.

From Table I, we see that for the discontinuous data `SQUARE`, I2C does not achieve a faster convergence than MUSCL. However, the smoother the initial data, the more I2C outdoes MUSCL, both in terms of order of convergence and magnitude of error. The reason why the C^1 data `WAVELET` gives rise to orders of convergence lesser than 2.5 is the presence of a local extremum. For the C^∞ monotone increasing data `HYPETAN`, I2C yields orders of convergence close to 4.

We are now concerned with two variable velocity cases: an *expansion* field $a(x) = x$, and a *compression* field $a(x) = -x$. We have just seen that `(I2C)`is much better than `(I1C)`. Therefore, we will concentrate on `(I2C)`. In order to get a thorough insight of its behavior, we will work with two types of initial data, namely `SQUARE` and `WAVELET` (see below). On the other hand, we have implemented a variant called `(I2C’)`, inwhich the velocity a is represented by a locally affine function and the characteristic points $X(j, m)$ are computed accordingly.

For all the runs, the mesh spacing is $\Delta x = 0.025$, while the time step Δt is set so that $\lambda_{\max} = |a|_{\max}\Delta t/\Delta x = 0.4$, where $|a|_{\max} = \max_x |a(x)|$. The computations were performed for the three types of initial condition `SQUARE`, `WAVELET`, and `HYPETAN`. For the sake of concision, only the results associated to `SQUARE` are shown in this paper.

Figure 11a displays the results for the expansion case. The initial data `SQUARE`,

$$u_0(x) = \begin{cases} 1 & \text{if } 0.05 < x < 0.15 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

being transported over a lapse $T = 3$ along the characteristic curves

$$x(x_0; t) = x_0 \exp(t) \tag{15}$$

associated to the velocity field $a(x) = x$, has been stretched by a factor of $\exp(3) \approx 20$. The display window is therefore $[0, 4]$. Comparison with the analytical solution and the MUSCL (Ultrabee) scheme shows that there is no visible difference between `(I2C)` and `(I2C’)`. The overcompressive behavior of Ultrabee can be regarded as a slight disadvantage. On the other hand, it can be noticed that, for all schemes, the numerical diffusion is much more important invariable velocity than in uniform velocity.

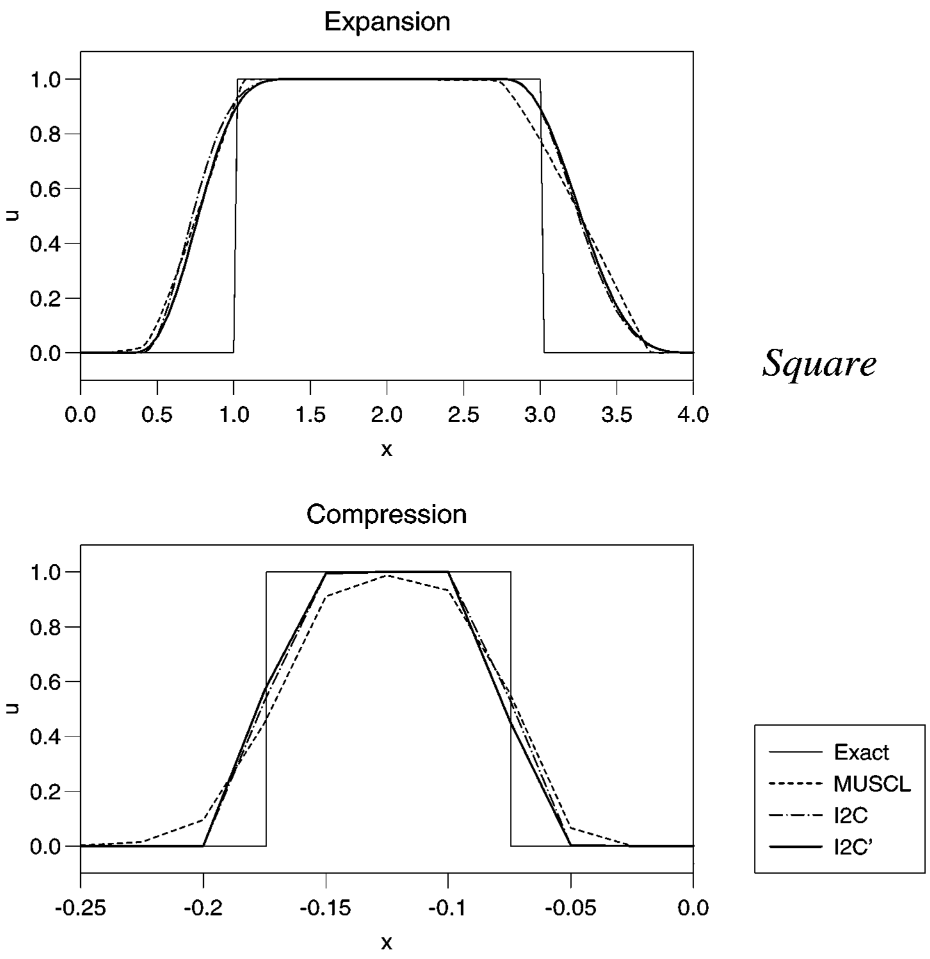

FIG. 11. Expansion and compression of SQUARE by Iserles schemes with proximity correction.

Figure 11b exhibits the results for the compression case. The initial SQUARE,

$$u_0(x) = \begin{cases} 1 & \text{if } -3 < x < -1 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

being transported over a lapse $T = 3$ along the characteristic curves

$$x(x_0; t) = x_0 \exp(-t) \tag{17}$$

associated to the velocity field $a(x) = -x$, has been squeezed by a factor $\exp(-3) \approx 0.05$. Consequently, the display window is now $[-0.25, 0]$.

2. THE 2-D CASE

For the general 2-D advection equation

$$u_t + a(x, y)u_x + b(x, y)u_y = 0, \tag{18}$$

it is usual to introduce the notion of *flowline* in the (x, y)-plane. A flowline is defined as a curve whose tangent is parallel to velocity field $\mathbf{c} = (a, b)$. More accurately, it is the set $\mathcal{L}(s_0; x_0, y_0)$ of points $(x(s), y(s))$, parameterized by s, that are solution of the differential system

$$\frac{\mathrm{d}x}{\mathrm{d}s} = \frac{a}{c} \quad \text{and} \quad \frac{\mathrm{d}y}{\mathrm{d}s} = \frac{b}{c} \tag{19}$$

along with the Cauchy conditions $x(s_0) = x_0$ and $y(s_0) = y_0$. Here, the normalizing factor

$$c = \|\mathbf{c}\| = \sqrt{a^2 + b^2} \tag{20}$$

is designed to make s the Euclidean curvilinear coordinate. One shall not confuse a flowline with a *bicharacteristic curve*, which is defined in the space (x, y, t) as the set $\mathcal{C}(t_0; x_0, y_0)$ of points $(x(t), y(t), t)$ such as its projection on the (x, y)-plane coincide with the flowline $\mathcal{L}(t_0, x_0, y_0)$.

Let σ be orthogonal coordinate associated to s the (x, y)-plane, as depicted in Fig. 12. Intuitively, σ is a continuous “tag” that allows us to identify each of the flowlines in the (x, y)-plane by an equation such as $\sigma =$ constant. If the mapping $(x, y) \leftrightarrow (\sigma, \tau)$ is a local diffeomorphism, then the couple (s, σ) can be considered as a coordinate system, at least locally, and we have

$$au_x + bu_y = (as_x + bs_y)u_s + (a\sigma_x + b\sigma_y)u_\sigma. \tag{21}$$

This equality involves partial derivatives of the new coordinates (s, σ) with respect to the old coordinates (x, y). Notice, however, that $a\sigma_x + b\sigma_y = c(\sigma_x x_s + \sigma_y y_s)$ is a multiple of the derivative of σ along a flowline. Since σ remains constant along a flowline, we have $a\sigma_x + b\sigma_y = 0$, which cancels out the second term on the right-hand side of (21). As for the remaining term, we have $as_x + bs_y = c(x_s s_x + y_s s_y) = cs_s = c$, after (19).

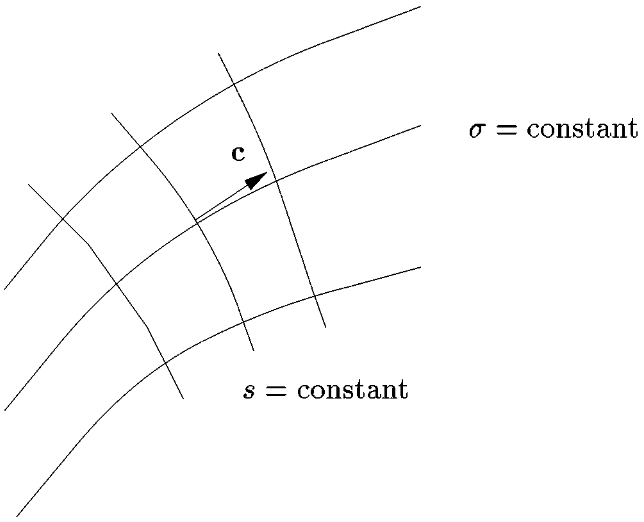

FIG. 12. Local coordinates (s, σ).

The advection equation (18) can finally be expressed as

$$u_t + c(s, \sigma)u_s = 0, \tag{22}$$

where u is to be thought of as a function of (t, s, σ). Thus, we are somehow brought back to the 1-D case. The main idea is therefore to apply the previously modified Iserles schemes along the flowline. Below are the details.

2.1. Uniform Velocity

For sake of simplicity, let us first work with a uniform velocity field. The flowlines are therefore straight lines. To further simplify the presentation, we assume that we are endowed with a structured mesh, but this hypothesis does not restrict the validity of the method. In Fig. 13, (i, j) is the node to be updated. Let us draw the flowline $\mathcal{L}$ going through (i, j) and let us suppose that $\mathcal{L}$ passes through the quadrilateral with nodes (i, j), $(i-1, j)$, $(i-1, j-1)$, and $(i, j-1)$. The cell which contains the backward flowline from node (i, j) is referred to as the *influence cell.*

Let $\mathcal{R}$ be the union of the two edges that do not contain (i, j), and define

$$p(i, j) = \mathcal{L} \cap \mathcal{R} \tag{23}$$

to be the *parent* of (i, j). We wish to apply the modified Iserles scheme (I2C) between $p(i, j)$ and (i, j). Of course, if $p(i, j)$ happens to be one of the existing nodes, for instance $(i-1, j-1)$ as in Fig. 13a, there is no problem at all. The problem arises when $p(i, j)$ is in a "general" position, as in Fig. 13b: how can we attribute a value for u to the parent point?

Unsurprisingly, the answer is interpolation. At first sight, it seems natural to try a linear interpolation to assess $u_{p(i,j)}$ from $u_{i-1,j-1}$ and $u_{i,j-1}$. However, as will be shown later, the results are hopelessly bad. We need higher order interpolation, while avoiding the use of neighboring nodes, which would destroy the compacity of the scheme. The solution to such a dilemma is to add extra unknowns defined at *additional points* over each edge.

In Fig. 14a, we add one additional point per edge, which ensures a quadratic interpolation. In Fig. 14b, we add two additional points by edge, which yields a cubic interpolation. At these extra points $\times$, the value of u has to be stored and updated in the same way as the

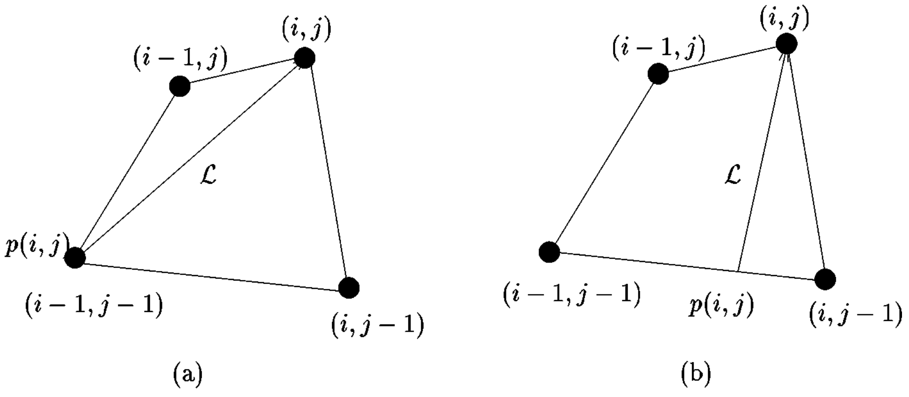

FIG. 13. Basic idea of 2-D Iserles schemes.

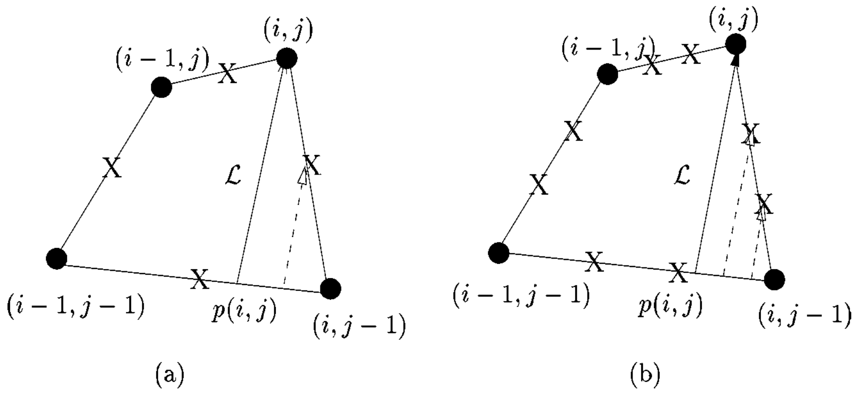

FIG. 14. Additional points for interpolation.

value of u at a node: each point $\times$ has a parent point, and we need to apply `(I2C)` between the parent and the point to be updated.

Note that the words "quadratic" and "cubic" describe the 1-D interpolation along the edges, but do not suggest any 2-D interpolation over the cell. Of course, the more accurate the interpolation, the more expansive the CPU time will be, since the number of points to be updated grows quite fast. Our experience is that the quadratic interpolation, using just one auxiliary point by edge, is a very acceptable trade-off.

It could be argued that, since the additional points along edges become extra unknowns which must be updated in the same manner as the original vertex unknowns, the efective stencil of the scheme increases and its compactness is therefore compromised. The point we wish to make is that the stencil remains compact in space, insofar as the stencil points all belong to the same cell. Of course, their number has increased in a manner not unlike the larger stencils used by ENO or other reconstruction schemes. But compactness with respect to space remains the key advantage: it makes it easier for us to work out high-order approximation formulae, regardless of how much distorted the mesh may be (ENO or other reconstruction schemes are uneasy to deal with over irregular meshes).

To help practitioners who would like to implement the full scheme, below are the interpolation formulae. Let us map the edge to interval $[-\frac{1}{2}, \frac{1}{2}]$. The midpoint of the edge is then 0, while the third and two-third points are at $-\frac{1}{6}$ and $\frac{1}{6}$. Let $\alpha \in [-\frac{1}{2}, \frac{1}{2}]$ be the *normalized abscissa* of a parent point.

- For linear interpolation, we are given $u_{-1/2}$ and $u_{1/2}$. Then,

$$u_\alpha \approx \frac{1}{2}(1-2\alpha)u_{-1/2} + \frac{1}{2}(1+2\alpha)u_{1/2}. \tag{24}$$

- For quadratic interpolation, we are given $u_{-1/2}$, u_0, and $u_{1/2}$. Then,

$$u_\alpha \approx -\alpha(1-2\alpha)u_{-1/2} + (1-4\alpha^2)u_0 + \alpha(1+2\alpha)u_{1/2}. \tag{25}$$

- For cubic interpolation, we are given $u_{-1/2}$, $u_{-1/6}$, $u_{1/6}$, and $u_{1/2}$. Then,

$$\begin{aligned} u_\alpha \approx &-\frac{1}{16}(1-2\alpha)(1-36\alpha^2)u_{-1/2} + \frac{9}{16}(1-4\alpha^2)(1-6\alpha)u_{-1/6} \\ &+ \frac{9}{16}(1-4\alpha^2)(1+6\alpha)u_{1/6} - \frac{1}{16}(1+2\alpha)(1-36\alpha^2)u_{1/2}. \end{aligned} \tag{26}$$

Once interpolation is done, a *projection step* has to be followed, in order to preserve monotonicity.[2] Analogously to the projection in the Iserles scheme, the newly computed value of u has to be "controlled" by the values of u at the closest points. For example, in the case of Fig. 14a, we have to assign

$$u_{p(i,j)} := \Pi_{|u_{i-1/2,j-1},u_{i,j-1}|}\left(u_{p(i,j)}\right), \tag{27}$$

where $(i-1/2, j-1)$ denotes the extra point on the edge containing $p(i, j)$. In the case of Fig. 14b, it is appropriate to request

$$u_{p(i,j)} := \Pi_{|u_{i-1/3,j-1},u_{i,j-1}|}\left(u_{p(i,j)}\right), \tag{28}$$

where $(i-2/3, j-1)$ and $(i-1/3, j-1)$ are the extra points.

Figure 15 is a 3-D sketch that summarizes the whole idea of the scheme. Once interpolation and projection steps are completed, the value $u_{i,j}^{n+1}$ is obtained by using the Iserles scheme with proximity correction in the plane P.

2.2. Variable Velocity

As in the 1-D case, we first require that **c**(x, y) is continuous. Furthermore, we need to assume

(H3) The (vector) values of **c** are given at the vertices (i, j) of the mesh.

(H4) Every, if any, *source* point of **c**, i.e., every postion (x, y) for which $\mathbf{c}(x, y) = \mathbf{0}$, and $\mu a(x+\epsilon\mu, y+\epsilon\nu) + \nu b(x+\epsilon\mu, y+\epsilon\nu) > 0$ for all unit vector $\mathbf{n} = (\mu, \nu)$ and for all $\epsilon > 0$ small enough, coincide with a vertex.

For the same reasons explained in the previous section, the evolution of u at a source point has to be given as part of the data of the problem. At a *sonic* point $[\mathbf{c}(x, y) = \mathbf{0}]$ that is not a source, we are justified in writing $u^{n+1} = u^n$.

Let us now explain how to update a *generic* point, i.e., either a grid point or an extra point. We proceed in four stages:

1. Determine the influence cell of the point to be updated. Once this is found, compute the average velocity over the cell.
2. Determine the parent of the point to be updated, using the average velocity to approximate the flowline by a straight line.
3. Determine the value of u at the parent point by interpolation and projection, as explained above.
4. Apply the 1-D Iserles scheme between the point to be updated and its parent.

Since **c** does not depend on t, steps 1 and 2 can be done once and for all before entering the time loop.

2.3. Reference Schemes

Our objective is to compare the new method to the second-order versions of: (i) the Donor scheme, based on 1-D approximation of fluxes across edges; and (ii) Colella's CTU scheme that we extended to the case of irregular meshes. Let us briefly recall these two schemes.

[2] Except for linear interpolation.

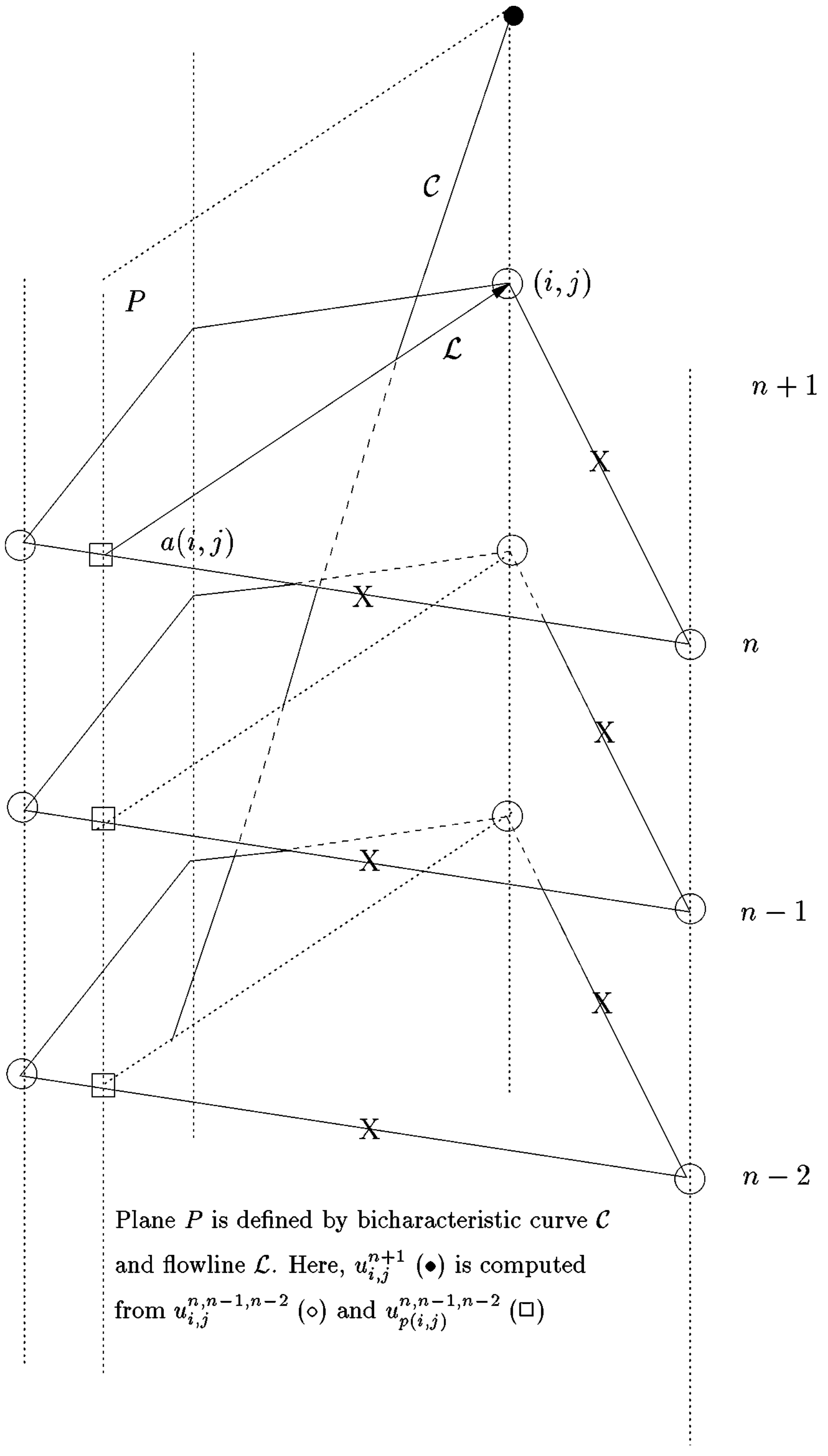
$\mathcal{C}$
(i,j)
P
$\mathcal{L}$
$n+1$
X
$a(i,j)$
X
n
X
X
$n-1$
X
X
$n-2$
Plane P is defined by bicharacteristic curve $\mathcal{C}$ and flowline $\mathcal{L}$. Here, $u_{i,j}^{n+1}$ ($\bullet$) is computed from $u_{i,j}^{n,n-1,n-2}$ ($\circ$) and $u_{p(i,j)}^{n,n-1,n-2}$ ($\square$)

FIG. 15. Modified Iserles scheme in 2-D with quadratic interpolation.

Always for the sake of simplicity, we present them for structured grids. In both Donor and CTU methods, the unknowns $u_{i,j}$ are located at the centers of the cells. Consider the quadrilateral cell

$$Q_{i,j} = \left\{N_{i-1/2,j-1/2}, N_{i+1/2,j-1/2}, N_{i+1/2,j+1/2}, N_{i-1/2,j+1/2}\right\} \tag{29}$$

at the mass-center $G_{i,j}$ of which $u_{i,j}$ is defined. Over $Q_{i,j}$, u is seen as an affine function of (x, y), i.e.,

$$u_{i,j}(x, y) = u_{i,j} + p_{i,j}\left(x - x_{G_{i,j}}\right) + q_{i,j}\left(y - y_{G_{i,j}}\right). \tag{30}$$

The way the gradient $(p_{i,j}, q_{i,j})$ is constructed and limited in order to achieve second-order accuracy, while preserving monotonicity, will be explained later. The average value of u over $Q_{i,j}$ is exactly $u_{i,j}$.

First, we put the advection equation (18) in the *conservative* form, which yields

$$u_t + (au)_x + (bu)_y = (a_x + b_y)u. \tag{31}$$

By the principle of finite-volume methods, we have

$$\begin{aligned} u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{1}{\mathcal{A}(Q_{i,j})}\int_0^{\Delta t} d\tau \oint_{\partial Q_{i,j}} u^\star(\ell, \tau)\mathbf{c}(\ell)\cdot\mathbf{n}(\ell)\,d\ell \\ + \frac{1}{\mathcal{A}(Q_{i,j})}\int_0^{\Delta t} d\tau \int_{Q_{i,j}} u \operatorname{div}\mathbf{c}\,dx\,dy, \end{aligned} \tag{32}$$

where $\mathcal{A}$ denotes the area, ∂ the boundary, $\mathbf{n}$ the outward normal unit vector, $\cdot$ the dot product, and $u^\star$ the "physical" solution along the edge, shifted in time. Since $\partial Q_{i,j}$ consists of four edges, we write

$$\oint_{\partial Q_{i,j}} \cdots = \int_{N_{i+1/2,j-1/2}}^{N_{i+1/2,j+1/2}} \cdots + \int_{N_{i+1/2,j+1/2}}^{N_{i-1/2,j+1/2}} \cdots + \int_{N_{i-1/2,j-1/2}}^{N_{i-1/2,j-1/2}} \cdots + \int_{N_{i-1/2,j-1/2}}^{N_{i+1/2,j-1/2}} \cdots \tag{33}$$

The Donor and CTU schemes differ in how the elementary integrals on the right-hand side are approximated.

2.3.1. *Donor*

In the Donor method, there is one velocity c for each edge. Thus, we have, for instance,

$$\int_{N_{i+1/2,j-1/2}}^{N_{i+1/2,j+1/2}} u^\star(\ell, \tau)[\mathbf{c}\cdot\mathbf{n}](\ell)\,d\ell = \left(\mathbf{c}_{i+1/2,j}\cdot\mathbf{n}_{i+1/2,j}\right)\int_{N_{i+1/2,j-1/2}}^{N_{i+1/2,j+1/2}} u^\star(\ell, \tau)\,d\ell. \tag{34}$$

Introduce the "apparent" velocity $v_{i+1/2,j} = \mathbf{c}_{i+1/2,j}\cdot n_{i+1/2,j}$. Then, the "flux" term $u^\star$ is evaluated as

$$u^\star(\ell, \tau) = \begin{cases} u_{i,j}^n\left(M_\ell - \tau v_{i+1/2,j}\mathbf{n}_{i+1/2,j}\right) & \text{if } v_{i+1/2,j} \geq 0 \\ u_{i+1,j}^n\left(M_\ell - \tau v_{i+1/2,j}\mathbf{n}_{i+1/2,j}\right) & \text{if } v_{i+1/2,j} < 0, \end{cases} \tag{35}$$

where M_ℓ is the point whose abscissa along $N_{i+1/2,j-1/2}N_{i+1/2,j+1/2}$ is ℓ. Of course, the function $u^n_{i,j}$ is given by (30). In a nutshell, this amounts to the consideration that locally, at the level of this edge, we have a 1-D avection problem governed by velocity $v_{i+1/2,j}$. The formulae for the other edges are similar. To this stage, the first term in the right-hand side of (32) is available.

On the other hand, it is possible to assign a mean value for div $\mathbf{c}$ over $Q_{i,j}$ using Green's formula. More precisely,

$$(\text{div } \mathbf{c})_{i,j} = \frac{1}{\mathcal{A}(Q_{i,j})} \sum_{e \in \text{edges of } Q_{i,j}} \mathbf{c}_e \cdot \mathbf{n}_e |e|. \tag{36}$$

This allows us to discretize the second term in the right-hand side of (32) by $\Delta t\,(\text{div}\,\mathbf{c})_{i,j} u^n_{i,j}$ in order to obtain an explicit scheme. Note, however, that most of the velocity fields we will be working with are divergence-free (div $\mathbf{c} = 0$), so we do not have to worry about this term.

2.3.2. *CTU*

In the CTU method, the velocity is given at the vertices of the cells. At each vertex, we define a local Riemann problem as the advection associated to the local (constant) velocity value. Let $E_{i+1/2,j}$ be the midpoint of $N_{i+1/2,j-1/2}N_{i+1/2,j+1/2}$. Then, we can write

$$\begin{aligned} \int_{N_{i+1/2,j-1/2}}^{N_{i+1/2,j+1/2}} u^\star(\ell,\tau)[\mathbf{c}\cdot\mathbf{n}](\ell)\,d\ell &= \left[\mathbf{c}_{i+1/2,j-1/2}\cdot\mathbf{n}_{i+1/2,j}\right] \int_{N_{i+1/2,j-1/2}}^{E_{i+1/2,j}} u^\star(\ell,\tau)\,d\ell \\ &\quad + \left[\mathbf{c}_{i+1/2,j+1/2}\cdot\mathbf{n}_{i+1/2,j}\right] \int_{E_{i+1/2,j}}^{N_{i+1/2,j+1/2}} u^\star(\ell,\tau)\,d\ell, \end{aligned} \tag{37}$$

while $u^\star$ is defined as

$$u^\star(\ell,\tau) = \begin{cases} u^n\left(M_\ell - \tau\mathbf{c}_{i+1/2,j-1/2}\right) & \text{if } \ell \in \left[N_{i+1/2,j-1/2}E_{i+1/2,j}\right] \\ u^n\left(M_\ell - \tau\mathbf{c}_{i+1/2,j+1/2}\right) & \text{if } \ell \in \left[E_{i+1/2,j}N_{i+1/2,j+1/2}\right]. \end{cases} \tag{38}$$

The point $M_{\ell-\tau}\mathbf{c}\ldots$ may fall into a couple of cells around edge $N_{i+1/2,j-1/2}N_{i+1/2,j+1/2}$. This is why we have not written down space indices as in (38). In practice, we have to deal with various types of intersection between a segment and several half-lines originating from each vertex. Over each cell, u^n is an affine function as specified in (30).

Under some geometrical CFL conditions, it is possible to compute exactly the integrals involved in this method. The details are given in [39]. As for the term containing div $\mathbf{c}$, it can be coped with similarly to what was done for the Donor method.

2.3.3. *Gradient Reconstruction*

On non-Cartesian meshes, it is difficult to generalize 1-D slope limiters such as Van Leer or Superbee. Inspired by Dukowicz and Kodis [14], we take the following approach. We first compute a "candidate" gradient $(p_{i,j}, q_{i,j})$ by solving the least-squares minimization problem

$$\min_{p_{i,j},q_{i,j}} \sum_{c \in \text{neighbor cells}} \left| u_{G_c} - \left\langle u_{G_{i,j}} + p_{i,j}\left(x_{G_c} - x_{G_{i,j}}\right) + q_{i,j}\left(y_{G_c} - y_{G_{i,j}}\right)\right\rangle\right|^2. \tag{39}$$

We may take either four or eight neighboring cells. The solution is achieved by solving a linear system.

Next, we search for the extremal values

$$u_{i,j}^{\downarrow} = \min_{c \in \text{neighbor cells}} u_{G_c} \quad \text{et} \quad u_{i,j}^{\uparrow} = \max_{c \in \text{neigbor cells}} u_{G_c} \tag{40}$$

of u around $Q_{i,j}$. Compare these to the extremal values over cell $Q_{i,j}$, namely,

$$\begin{aligned} u_{i,j}^{-} &= u_{G_{i,j}} + \min_{v \in \text{vertices of } Q_{i,j}} p_{i,j}\left(x_v - x_{G_{i,j}}\right) + q_{i,j}\left(y_v - y_{G_{i,j}}\right) \\ u_{i,j}^{+} &= u_{G_{i,j}} + \max_{v \in \text{vertices of } Q_{i,j}} p_{i,j}\left(x_v - x_{G_{i,j}}\right) + q_{i,j}\left(y_v - y_{G_{i,j}}\right) \end{aligned} \tag{41}$$

in order to define the ratios

$$\sigma_{i,j}^{-} = \max\left(0, \frac{u_{i,j}^{\downarrow} - u_{G_{i,j}}}{u_{i,j}^{-} - u_{G_{i,j}}}\right) \quad \text{and} \quad \sigma_{i,j}^{+} = \max\left(0, \frac{u_{i,j}^{\uparrow} - u_{G_{i,j}}}{u_{i,j}^{+} - u_{G_{i,j}}}\right). \tag{42}$$

Now, set

$$\sigma_{i,j} = \min(1, \sigma_{i,j}^{+}, \sigma_{i,j}^{-}), \tag{43}$$

then change the candidate gradient as

$$p_{i,j} := \sigma_{i,j} p_{i,j} \quad \text{and} \quad q_{i,j} := \sigma_{i,j} q_{i,j}. \tag{44}$$

This limitation procedure prevents new extremal values of u from arising.

2.4. Numerical Results

Extensive numerical simulations have been carried out [39, 40] in order to test our method. In this paper, we present six of them: two for a regular Cartesian mesh and four for two deformed meshes. The initial data that will be used throughout the rest of the section are the 2-D Cartesian products of their 1-D counterparts `SQUARE`, `WAVELET`, and `HYPETAN`. The first two of them are compact-supported and their initial support is denoted by S. As for `HYPETAN`, it is not compact-supported, and its relevancy may be questionable in view of practical applications. However, it is interesting to know whether or not this kind of data still gives rise to a very high order of convergence. In most simulations, for the sake of concision, we show the results corresponding to `SQUARE` alone. Unless otherwise stated, the comments for `WAVELET` and `HYPETAN` are similar.

2.4.1. *Regular Meshes*

Uniform translation. Inside the domain $\overline{\Omega} = [0, 3] \times [0, 6]$, the initial square

$$S = [x_c - 0.25, x_c + 0.25] \times [y_c - 0.25, y_c + 0.25] \tag{45}$$

is located at $(x_c, y_c) = (0.75, 5.25)$. The uniform velocity vector is $(a, b) = (1, -3)$, so that after a lapse of time $T = 1.5$, the center of the square has moved to $(2.25, 0.75)$. For

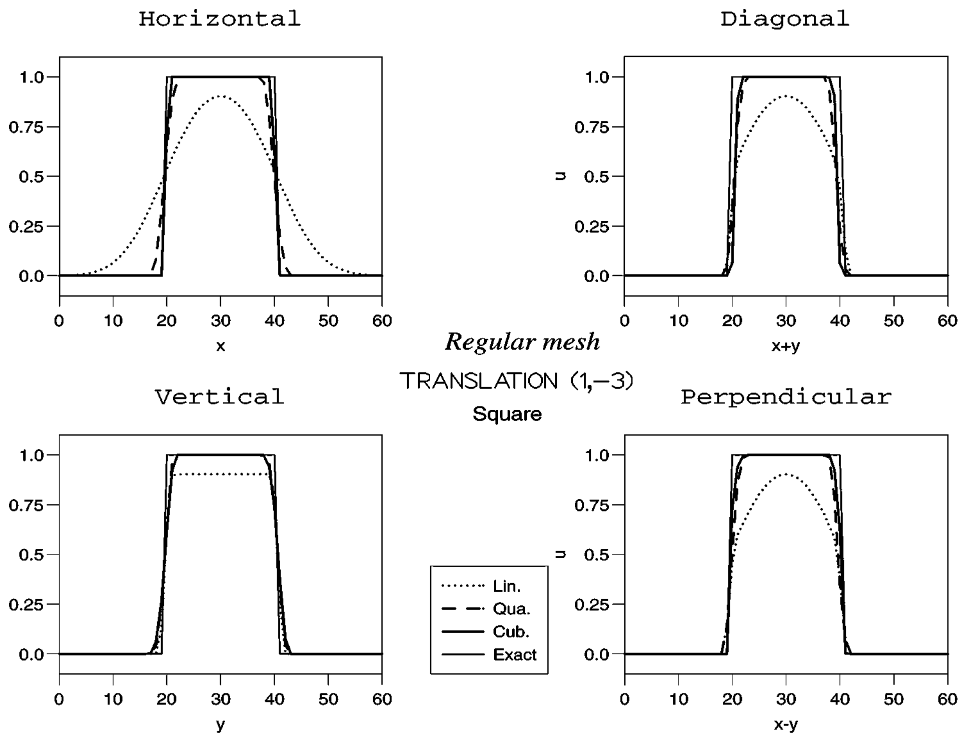

FIG. 16. Translation (1, −3) of SQUARE by modified Iserles schemes.

the runs in Fig. 16 and 17, the grid is made up of squares of size $h = 0.025$. The CFL ratio is 0.4.

Over the bottom right corner square $[1.5, 3] \times [0, 1.5]$, we record the output and proceed to various 1-D cuts along four directions: horizontal, vertical, diagonal SW-NE, and diagonal SE-NW (perpendicular). The results are shown in the following pages. The reader should be aware of the fact that the numbers in the abscissa axis simply refer to sample tags, and do not represent an actual position.

First, in Fig. 16, we compare different interpolation methods for the modified Iserles scheme. It is seen that linear interpolation is too diffusive. Cubic interpolation is slightly better than quadratic interpolation, but cannot always be afforded in view of the CPU time. Note that in this experiment, since $(a, b) = (1, -3)$, the cubic interpolation turns out be exact for some of the parent points.

Figure 16 also evidences the fact that we need at least one auxiliary point on each edge. Otherwise, the simple-minded linear interpolation destroys everything! On the other hand, simulations of translation with other velocities such as $(1, -4)$ show that the cubic interpolation (2 extra points per edge) does not bring about a tremendous improvement. Therefore, the quadratic interpolation (1 extra point per edge) appears to be good trade-off between accuracy and computational cost.

Next, we compare the Iserles scheme with quadratic interpolation to the Donor and CTU schemes that were previously described. This is depicted in Fig. 17. Obviously, the modified Iserles scheme is closer to the exact solution.

Let us investigate now the respective orders of convergence. These are numerically computed by logarithmic least-squares regression over the relative L^1-errors measured for the

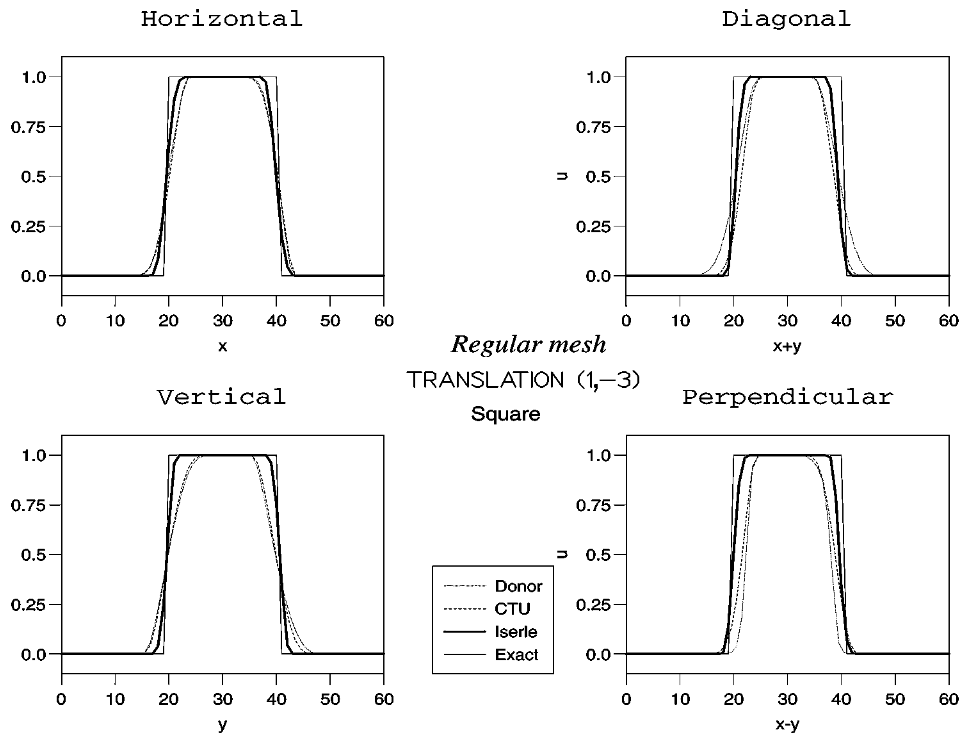

FIG. 17. Translation (1, −3) of SQUARE on regular mesh.

mesh spacing range $h \in \{0.05, 0.025, 0.0125\}$. In Table II, which summarizes the study of convergence for the $(1, -3)$-translation experiment, the word "DON" stands for Donor, while "ISE" denotes the modified Iserles scheme with quadratic interpolation (one extra-point per edge). Several remarks can be pointed out:

- In a way that is much more dramatic than the 1-D case (Table I), the orders of convergence of DON, CTU, and ISE are quite sensitive to the value of the CFL ratio. This remark applies especially to DON. The orders of DON and ISE decrease with the CFL, while that of CTU increases with the CFL.
- The order of ISE is not always higher than that of CTU, except for the data HYPETAN. Nevertheless, the magnitude of the relative L^1-error due to ISE is systematically much lower than that due to CTU. For instance, in the middle panel of Fig. 18 that corresponds to the WAVELET data, the error associated to ISE with $h = 0.05$ is about 10 times less than that

TABLE II
Orders of Convergence for the 2-D Translation Experiment

	SQUARE			WAVELET			HYPETAN		
CFL	DON	CTU	ISE	DON	CTU	ISE	DON	CTU	ISE
0.5	0.23	0.75	0.65	0.40	2.15	1.92	0.46	1.47	1.87
0.4	0.37	0.74	0.67	0.56	2.09	2.00	0.70	1.29	1.93
0.2	0.59	0.73	0.69	0.75	1.82	2.11	0.84	1.17	1.97
0.1	0.71	0.72	0.70	0.95	1.68	2.17	0.98	1.08	1.99

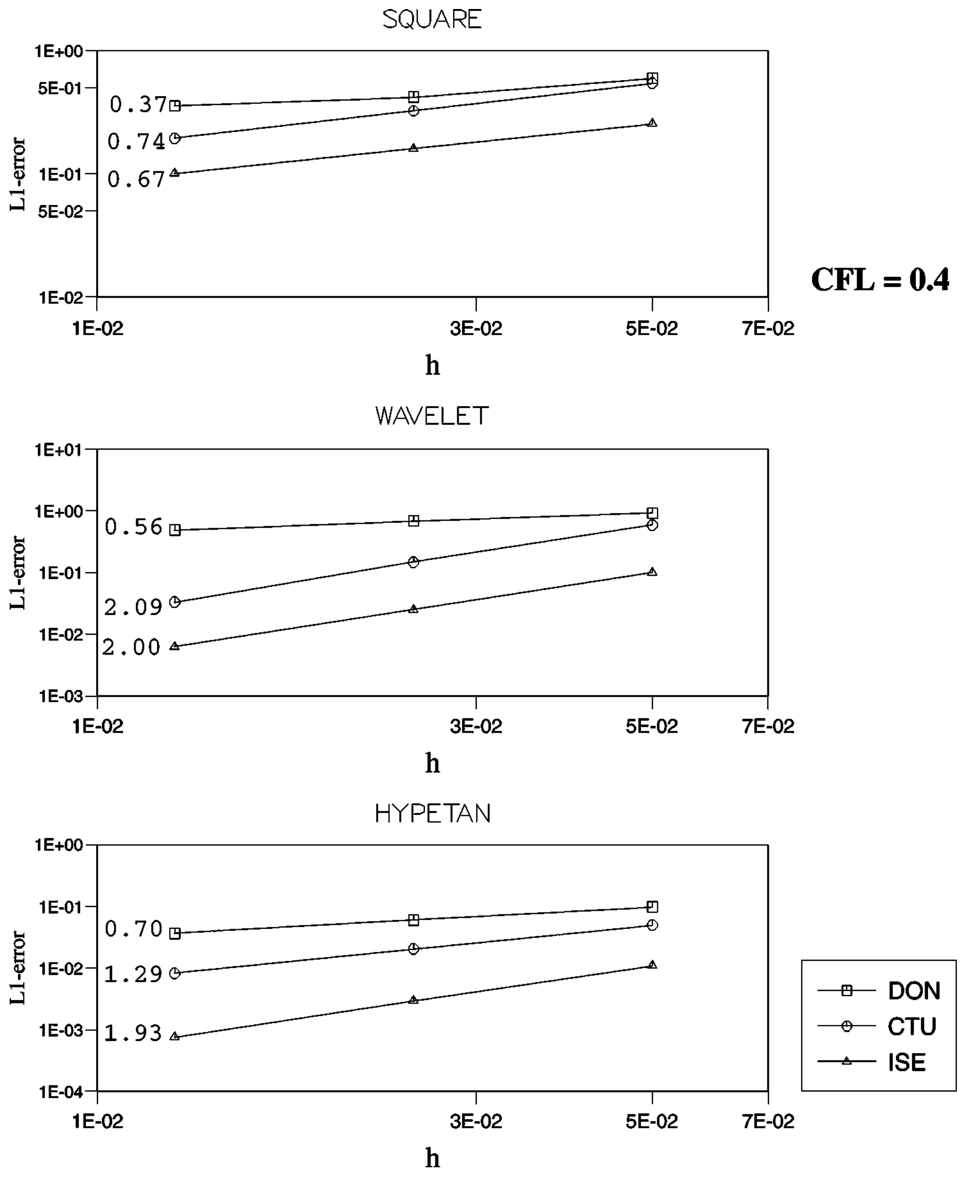

FIG. 18. Convergence of various schemes on logarithmic scale for a 2-D advection experiment.

associated to CTU with the same space step. It is even less than the error due to CTU with $h = 0.025$, a space step twice smaller! This implies that for a fixed threshold of relative L^1-error, it is possible to use the ISE method with a space step twice as big as that of the CTU method (at least for the range of space steps of practical use for applications), so that the auxiliary point on each edge in the ISE method does not penalize the computer's memory.

• For the very smooth data HYPETAN, the ISE method does not achieve orders higher than 2. This stems from the fact that we are using quadratic interpolation over each edge. Had we used cubic interpolation (two auxiliary points per edge), the results would be better.

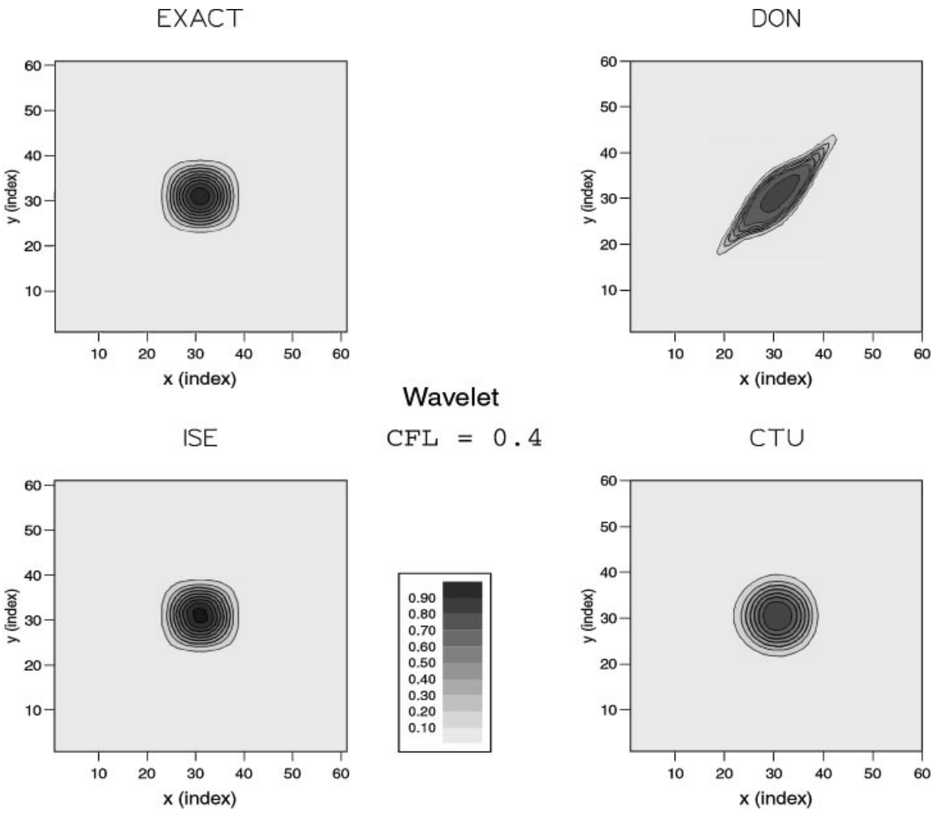

FIG. 19. Final snapshots of the translation experiment for the `WAVELET` data.

Another important aspect of the comparison among different 2-D schemes is how "isotropic" they are. Indeed, some schemes may yield a small error, as well as a good order of convergence, but may distort the solution in one direction. We have not devised a quantitative study of anisotropy for the schemes. Instead, we have systematically looked at the isolines on the snapshots of the solutions. With this visual scrutiny, the ISE scheme has always turned out to be the one that best preserves the shape of the initial data. As an example, Fig. 19 contains four snapshots representing the final solutions of the $(1, -3)$-translation experiment for the `WAVELET` data. We see that DON is utterly uncompetitive, while CTU is slightly dissymmetrical. Only ISE is close enough to the exact solution.

Circular vortex. At the center $(x_c, y_c) = (0.75, 0.75)$ of the square $\overline{\Omega} = [0, 1.5] \times [0, 1.5]$, initial data are set. The velocity field is

$$a(x, y) = -2\pi(y - y_c) \quad \text{and} \quad b(x, y) = 2\pi(x - x_c), \tag{46}$$

so that after at $t = 10$, the initial data has made 10 turns. The results are displayed in Figs. 20 and 21 under the form of 1-D cuts. As before, we refer to the schemes as DON, CTU, and ISE. For `SQUARE`, the ISE scheme yields the least diffusive curves. As for `WAVELET`, it has to be noticed that since the "origin" of the vortex is a sonic point which is not a source, the value of u at (x_c, y_c) remains constant in time, as was previously explained. This accounts for the seemingly "perfect" peak of the wavelet. Note that it does not make sense to perform this vortex experiment for the `HYPETAN` data.

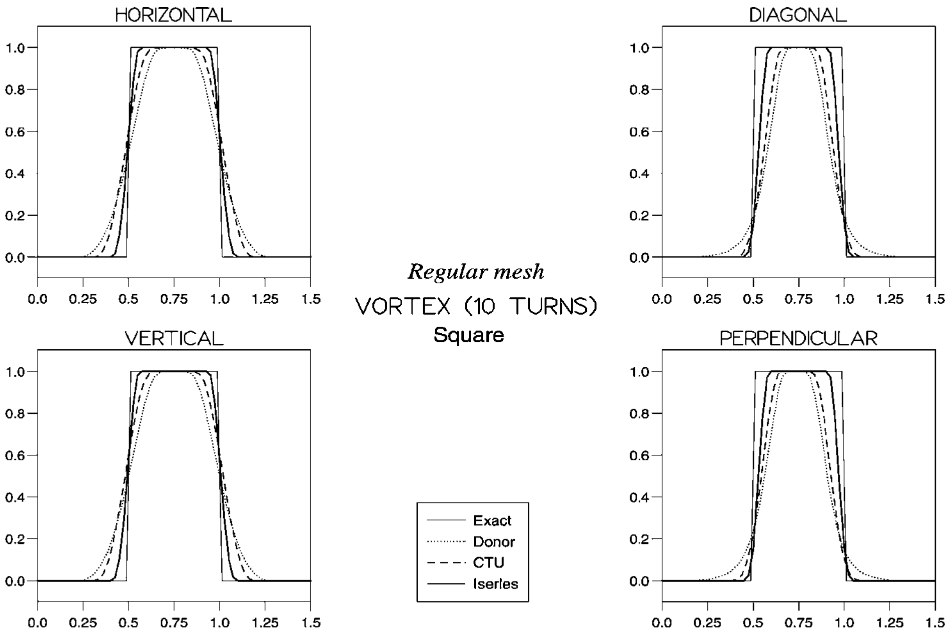

FIG. 20. Vortex of SQUARE over a regular mesh.

2.4.2. *Deformed Meshes*

Several kinds of irregular meshes have been tested. We will show the results for two of them: a moderately deformed mesh, called *trapezoid mesh*, and a highly deformed one, called *Kershaw mesh*.

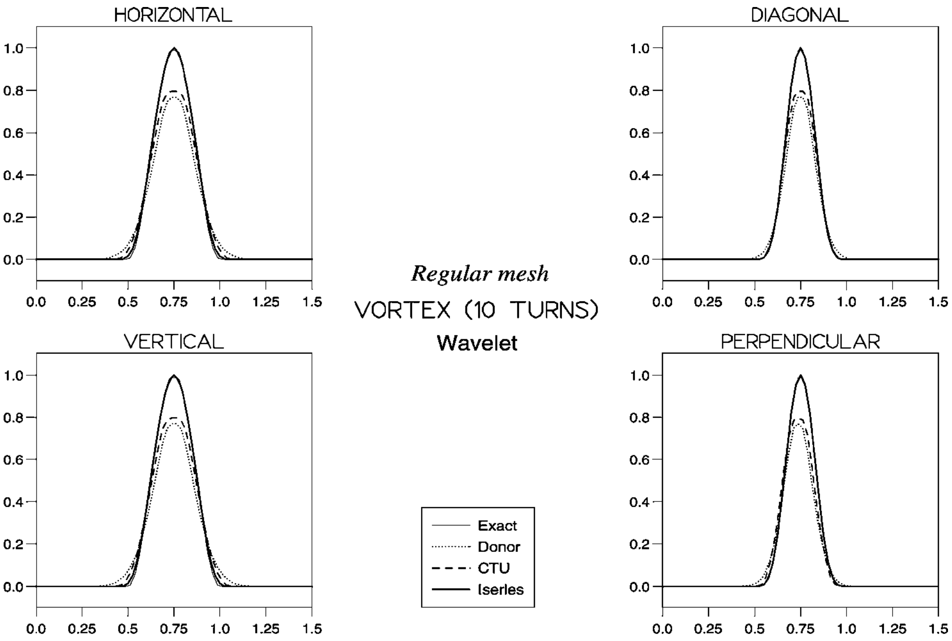

FIG. 21. Vortex of WAVELET over a regular mesh.

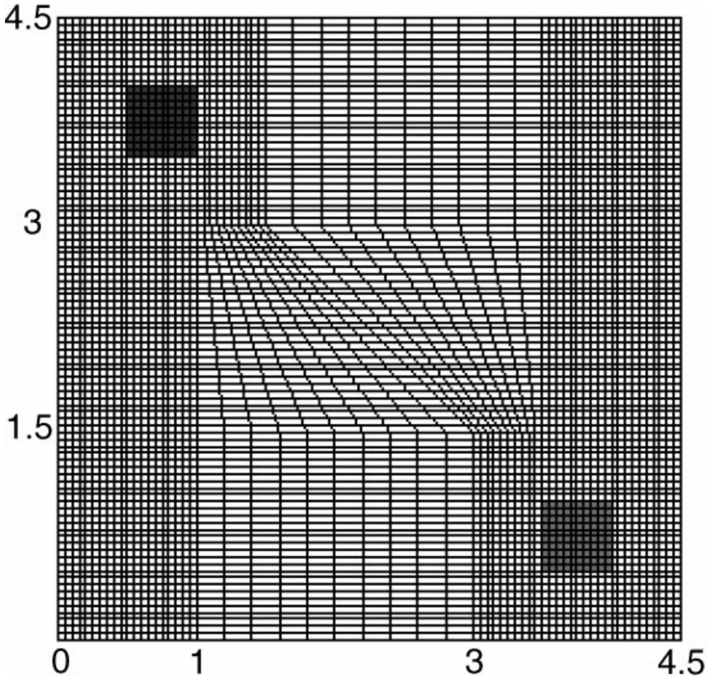

FIG. 22. Trapezoid mesh.

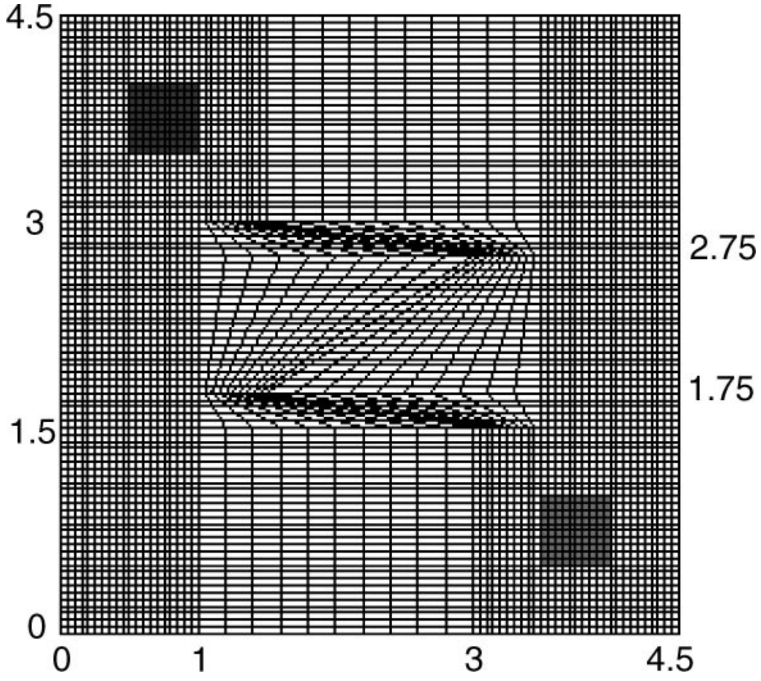

FIG. 23. Kershaw’s mesh.

The trapezoid mesh is depicted in Fig. 22. The Kershaw mesh, inspired from [26], is in Fig. 23. Both of them represent the domain $\bar{\Omega} = [0, 4.5] \times [0, 4.5]$. This domain is discretized by 120×180 cells. Most of these are rectangles, the smallest of which are squares of size 0.025×0.025. In regions $1 \leq x \leq 3.5$ and $1.5 \leq y \leq 1.75$ or $2.75 \leq y \leq 3$, the cells are skewed trapezoids.

In order to move the initial data, supported by $S = [0.5, 1] \times [3.5, 4]$, to its final position $S' = [3.5, 4] \times [0.5, 1]$, we perform either the uniform translation of velocity $(a, b) = (1, -1)$ or a rotation by a quarter of turn counterclockwise, the center of which is located at $(x_o, y_o) = (3.75, 3.75)$. Intuitively, the initial data has to cross a "turbulence" zone, in the middle of the mesh. Therefore, we want to measure the extent of "damages" caused by each method.

Again, in all runs, the CFL ratio is set to 0.4. We recall that the 1-D cuts that will be shown to have been executed over the "arrival" square S' along the four main directions: horizontal, vertical, first, and second diagonals. As for the scheme itself, it is the modified Iserles scheme with quadratic interpolation over the edges. Since the comments are quite similar for `SQUARE`, `WAVELET`, and `HYPETAN`, we only show the results corresponding to the first data, which is the most difficult case.

Trapezoid mesh. The results associated with the translation experiment are shown in Fig. 24. Those corresponding to the rotation experiment are in Fig. 25. The quality of the

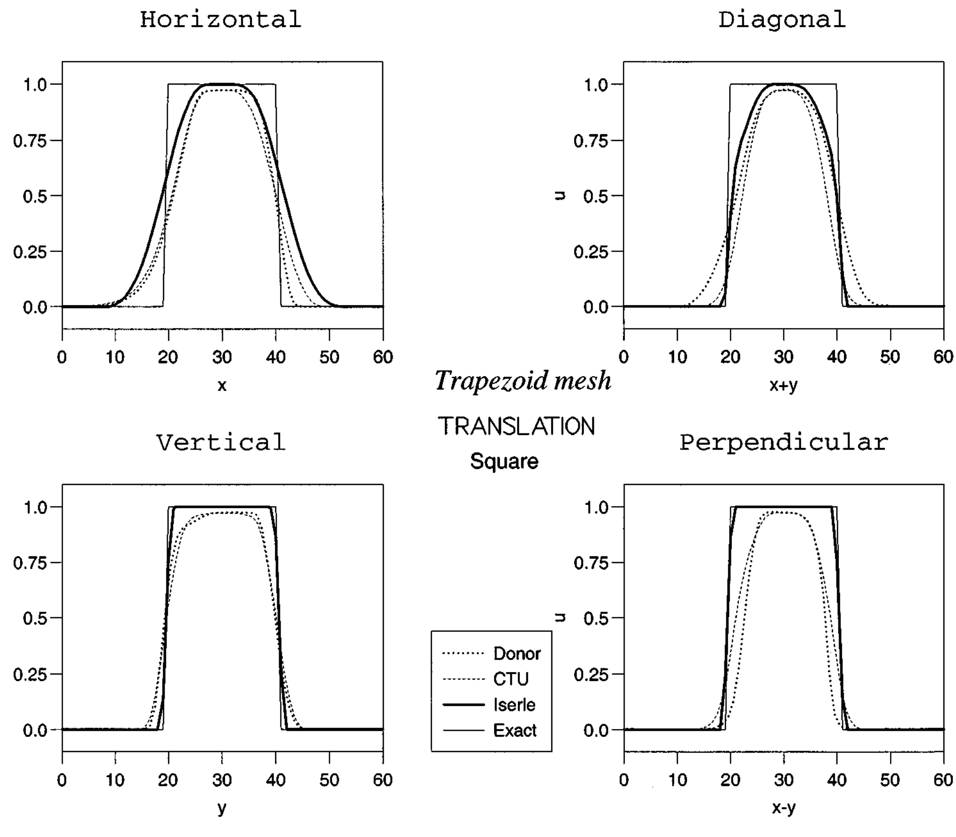

FIG. 24. Translation of `SQUARE` on the trapezoid mesh.

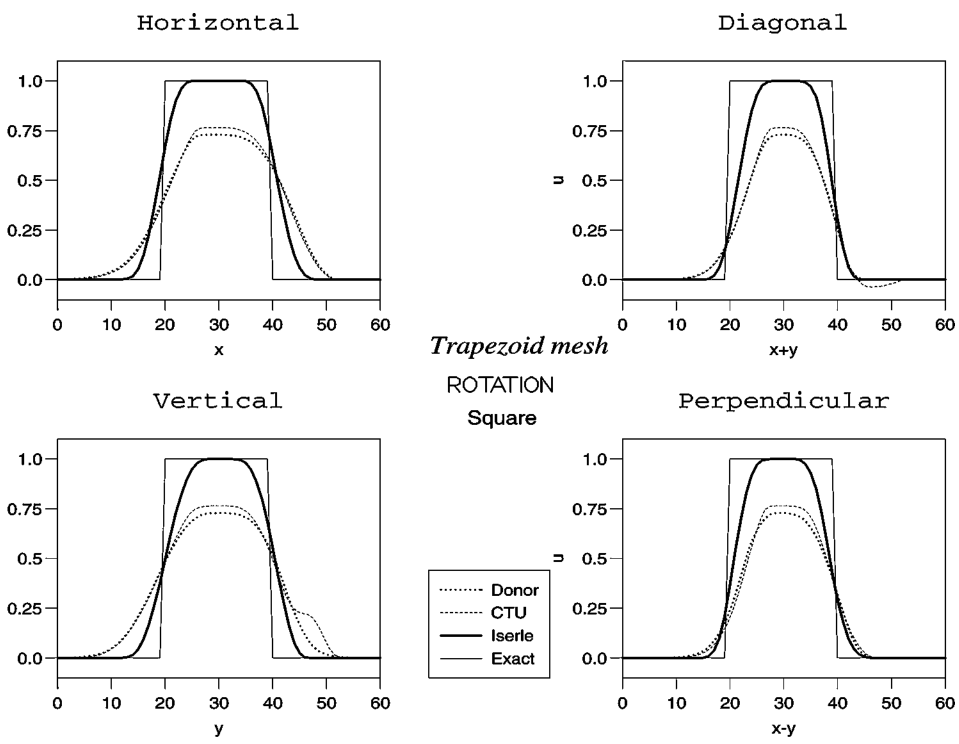

FIG. 25. Rotation of SQUARE on the trapezoid mesh.

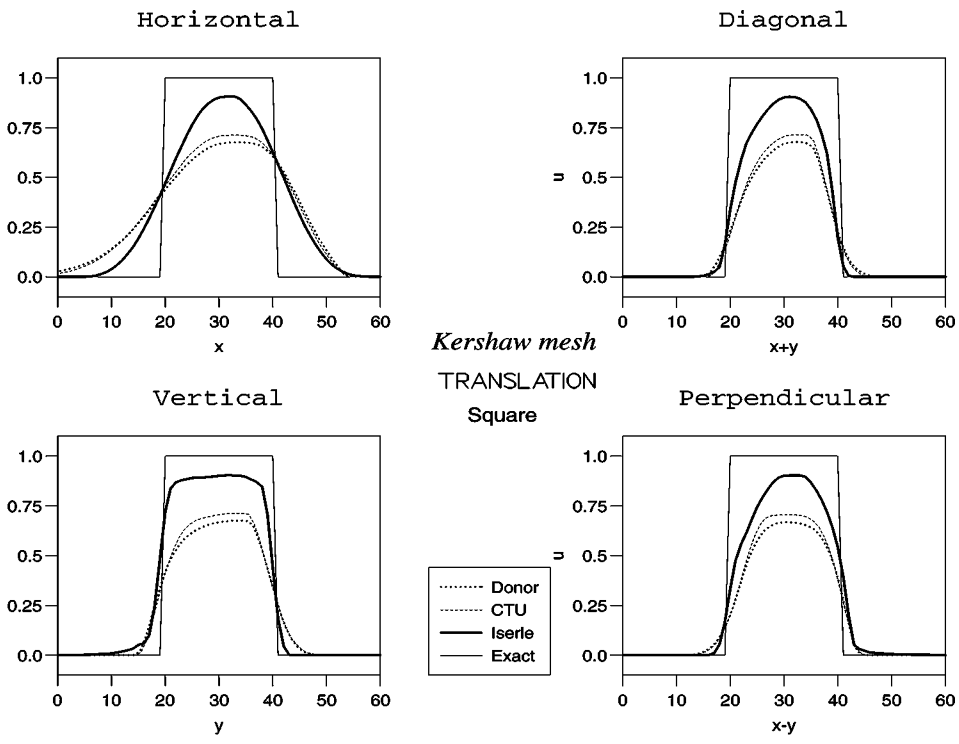

FIG. 26. Translation of SQUARE on Kershaw's mesh.

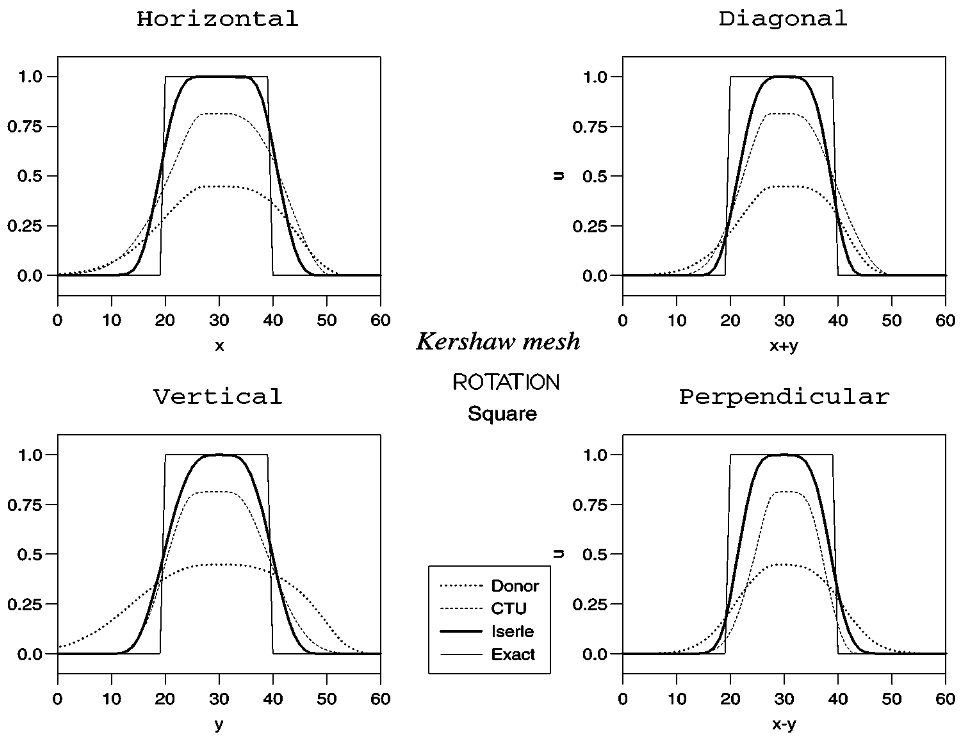

FIG. 27. Rotation of `SQUARE` on Kershaw's mesh.

numerical results is not too bad. The comments are pretty similar to the case of regular mesh.

Kershaw mesh. The results associated with the translation experiment are shown in Fig. 26. Those corresponding to the rotation experiment are in Fig. 27. The quality of the numerical results is much deteriorated on this very stiff example, but the comments are pretty similar to the case of regular mesh.

CONCLUSION

Our primary purpose was to look for genuinely multidimensional schemes for advection problems. We ended up with a "1-D" compact scheme by nature, since it has to be applied along the "appropriate" direction of the flowline. This answer is, ultimately, very natural. Unlike general conservation laws in multiple dimensions, advection is purely a "1-D" issue, at least locally.

In theory, the idea of the proposed scheme is quite simple. In practice, we need to introduce several tricks: projection, proximity correction, and monotone interpolation at parent points. To our knowledge, this approach seems to be new. We insist on the fact that modified Iserles schemes can be used over any kind of mesh, no matter of whether they are structured, unstructured, or hybrid. In this paper, we have explained the ideas of the scheme for structured mesh only because the notations and the computer implementations are much easier.

Because of the many tricks involved, the method may look rather complex at first sight, especially in multiple dimensions. In reality, the programming efort incurred is not as substantial as other methods, e.g., discontinuous Galerkin schemes. This claim can be supported by a quick counting of the number of variables required in each method.

Further numerical experiments—not shown in this paper—demonstrate that, actually, the Iserles schemes over a coarse mesh are more accurate than traditional "cheaper" schemes over refined meshes (i.e., with an increased grid resolution). For instance, over a Cartesian mesh with $\Delta x = \Delta y = h$, the Iserles scheme with quadratic interpolation (one extra point per edge) is far more accurate than the CTU scheme over a Cartesian mesh with $\Delta x = \Delta y = h/2$. Over the same h-mesh, the Iserles scheme with cubic interpolation (two extra points per edge) remains slightly better than the CTU scheme over a $h/3$-mesh. This feature can prove to be very useful, since in practical applications, we mostly have to work with imposed deformed meshes, which are not convenient to refine.

Over regular 2-D meshes, the actual number of computations can be optimized. In such a case, the CPU times are in the ratio of about 1 : 2 : 4 (DON, CTU, ISE). Over deformed 2-D meshes, computations are more expensive. The CPU times are now in the ratio of 1 : 3 : 5. It would be risky, and dishonest, for us to claim that the benefits of the new scheme outweigh other factors such as cost or programming complexity. As previously said, our initial objective was to search for an accurate as possible advection scheme, assuming that we are willing to pay the price. We are happy enough to have devised the modified Iserles schemes, because these are the best answer we could have expected. The reason why we discounted the discontinuous Galerkin method is not so much because it is expensive, but because fundamentally it is not a genuinely multidimensional scheme: to evaluate the flux across each edge, we have to consider local 1-D Riemann problems in the normal direction.

The next step will be to extend these modified Iserles schemes to a 2-D time-dependent velocity field $\mathbf{c}(x, y, t)$. Then, the last step will be to generalize everything to 3-D meshes.

APPENDIX

Assume a constant velocity field a in 1-D. Let us proceed to show

THEOREM 2.1. *If the first time step is the upwind scheme, then the scheme* `I1C` *preserves monotonicity. Similarly, if the first time step is the upwind scheme and if the second time step is the* `I1C` *scheme, then the scheme* `I2C` *preserves monotonicity.*

Proof. We prove only the first part of the claim, the proof of the second one being exactly similar. To fix ideas, assume that $a > 0$ and that the initial data u^0 is increasing, so that $u_i^0 \leq u_{i+1}^0$ for all i. After the first time step, which is a mere upwind scheme, we have $u_i^0 \leq u_{i+1}^1 \leq u_{i+1}^0$ for all i. Let us inspect the second time step.

If $\lambda \leq \frac{1}{2}$, we have $u_i^2 \in |u_{i-1}^0, u_i^1| = [u_{i-1}^0, u_i^1]$ because of the projection step (6), hence,

$$u_{i-1}^1 \leq u_{i-1}^0 \leq u_i^2 \leq u_{\mathrm{i}}^1. \tag{47}$$

The same inequality holds for the index $i + 1$, i.e.,

$$u_i^1 \leq u_i^0 \leq u_{i+1}^2 \leq u_{i+1}^1, \tag{48}$$

so that we can infer

$$u_i^2 \leq u_i^1 \leq u_i^0 \leq u_{i+1}^2. \tag{49}$$

In other terms, u^2 is still increasing. Thanks to the intermediate inequalities in (49), we can go on by induction on the time step and the result follows.

If $\lambda > \frac{1}{2}$, we have $u_i^2 \in |u_{i-1}^1, u_{i-1}^0| = [u_{i-1}^1, u_{i-1}^0]$ because of the projection step (6), hence,

$$u_{i-1}^1 \leq u_i^2 \leq u_{i-1}^0 \leq u_i^1. \tag{50}$$

The same inequality holds for the index $i+1$, i.e.,

$$u_i^1 \leq u_{i+1}^2 \leq u_i^0 \leq u_{i+1}^1, \tag{51}$$

so that we can infer

$$u_i^2 \leq u_{i-1}^0 \leq u_i^1 \leq u_{i+1}^2, \tag{52}$$

and we can carry on by induction.

ACKNOWLEDGMENTS

The authors thank Arnaud Torrès, Marc Zolver, and Guillaume Gautier for their interest in this work. We are also grateful to the two anonymous reviewers whose valuable comments helped us to improve this paper.

REFERENCES

1. A. A. Amsden, P. J. O'Rourke, and T. D. Butler, *KIVA-II: A Computer Program for Chemically Reactive Flows with Sprays*, Report LA-11560-MS (Los Alamos National Laboratory, 1989).
2. M. Arora and P. L. Roe, A well-behaved TVD limiter for high-resolution calculations of unsteady flow, *J. Comput. Phys.* **132**, 3 (1997).
3. J. B. Bell, C. N. Dawson, and G. R. Shubin, An unsplit, higher order Godunov method for scalar conservation laws in multiple dimensions, *J. Comput. Phys.* **74**, 1 (1998).
4. D. J. Benson, Momentum advection on a staggered mesh, *J. Comput. Phys.* **100**, 143 (1992).
5. S. J. Billett and E. F. Toro, On WAF-type schemes for multidimensional hyperbolic conservation laws, *J. Comput. Phys.* **130**, 1 (1997).
6. J. J. Chattot and S. Malet, A box-scheme for the Euler equations, in *Non-Linear Hyperbolic Problems*, edited by C. Carasso, P. A. Raviart, and D. Serre, Lecture Notes in Mathematics (Springer-Verlag, Paris, 1987), Vol. 1270, pp. 82–102.
7. B. Cockburn and C. W. Shu, TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws II: General framework, *Math. Comp.* **52**, 411 (1989).
8. B. Cockburn, S. Hou, and C. W. Shu, The Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case, *Math. Comp.* **54**, 545 (1990).
9. P. Colella, Multidimensional upwind methods for hyperbolic conservation laws, *J. Comput. Phys.* **87**, 171 (1990).
10. P. Colella and P. R. Woodward, The piecewise parabolic method for gas-dynamical simulations, *J. Comput. Phys.* **54**, 174 (1984).
11. H. Deconinck, R. Struijs, and P. L. Roe, Fluctuation splitting for multidimensional convection problem: An alternative to finite volume and finite element methods, in *VKI Lecture Series 1990-03 on CFD* (von Karman Institute, Belgique, 1991).
12. H. Deconinck, R. Struijs, and G. Bourgois, Compact Advection Schemes on Unstructured Grids, in *VKI Lecture Series 1993-04 on CFD* (von Karman Institute, Belgium, 1993).

13. B. Després and F. Lagoutière, Un schéma non-linéaire anti-dissipatif pour l'équation d'advection linéaire, *C. R. Acad. Sci. Paris* **238**, I, 939 (1999).
14. J. K. Dukowicz and J. W. Kodis, Accurate conservative remapping for arbitrary Lagrangian–Eulerian computations, *SIAM J. Sci. Stat. Comp* **8**, 305 (1987).
15. D. Fauvin, *Étude de Schémas Non-Dissipatifs pour Certaines Équations Hyperboliques*, CEA/DAM/CELV Technical Report (1995).
16. E. Godlewski and P. A. Raviart, *Hyperbolic Systems of Conservation Laws*, Mathématiques et Applications (SMAI, Ellipses, Paris, 1991).
17. E. Godlewski and P. A. Raviart, *Numerical Approximation of Hyperbolic Systems of Conservation Laws*, Applied Mathematical Sciences (Springer-Verlag, New York, 1996).
18. A. Harten, P. D. Lax, and B. Van Leer, On upstream differencing and Godunov-type schemes for hyperbolic conservation laws, *SIAM Review* **25**, 35 (1983).
19. A. Harten, ENO schemes with subcell resolution, *J. Comput. Phys.* **83**, 148 (1989).
20. A. Harten and S. Osher, Uniformly High-Order Accurate Nonoscillatory Schemes I, *SIAM J. Numer. Anal.* **24**, 279 (1987).
21. M. J. Holst, *Notes on the KIVA-II Software and Chemically Reactive Fluid Mechanics*, Report UCRL-ID-112019 (Lawrence Livermore National Laboratory, Livermore, CA, 1992).
22. M. E. Hubbard and P. L. Roe, Compact high-resolution algorithms for time-dependent advection on unstructured grids, preprint.
23. A. Iserles, Generalised leapfrog schemes, *IMA J. Numer. Anal.* **6**, 381 (1986).
24. R. J. LeVeque, *Numerical Methods for Conservation Laws*, Lectures in Mathematics (ETH Zürich, Birkhäuser Verlag, Berlin, 1992).
25. R. J. LeVeque, Nonlinear conservation laws and finite volume methods for astrophysical fluid flow, in *Computational Methods for Astrophysical Fluid Flow, with lectures by* D. Mihalas, E. Dorfi, and E. Mueller, edited by O. Steiner and A. Gautschy, 27th Saas-Fee Advanced Course Lecture Notes (Springer-Verlag, Berlin/ New York, 1998).
26. J. E. Morel, J. E. Dendy, Jr., M. L. Hall, and S. W. White, A cell-centered Lagrangian–Mesh diffusion differencing scheme, *J. Comput. Phys.* **103**, 286 (1992).
27. P. J. O'Rourke and M. S. Sahota, A variable explicit/implicit numerical method for calculating advection on unstructured Meshes, *J. Comput. Phys.* **143**, 312 (1998).
28. H. Paillère, J. Boxho, G. Degrez, and H. Deconinck, Multidimensional upwind residual distribution schemes for the convection–diffusion equation, *Int. J. Numer. Meth. Fluids* **23**, 923 (1996).
29. B. Perthame and Y. Qiu, A variant of Van Leer's method for multidimensional systems of conservation laws, *J. Comput. Phys.* **112**, 370 (1994).
30. P. L. Roe and D. Sidilkover, Optimum positive linear schemes for advection in two and three dimensions, *SIAM J. Numer. Anal.* **29**, 1542 (1992).
31. P. L. Roe, Linear bicharacteristic schemes without dissipation, *SIAM J. Sci. Comput.* **19**, 1405 (1998).
32. J. Saltzman, An unsplit 3-D upwind method for hyperbolic conservation laws, *J. Comput. Phys.* **115**, 153 (1994).
33. D. Serre, *Systèmes de Lois de Conservation I & II* (Diderot Éditeur Arts et Sciences, Paris, 1996).
34. C. W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, II *J. Comput. Phys.* **83**, 32 (1989).
35. D. Sidilkover and P. L. Roe, *Unification of Some Advection Schemes in Two Dimensions*, ICASE Report 95-10 (Indiana Council of Administrators of Special Education, Indianapolis, 1995).
36. D. Sidilkover, *A New Time-Space Accurate Scheme for Hyperbolic Problems. I: Quasi-Explicit Case*, ICASE Report 98-25 (Indiana Council of Administrators of Special Education, Indianapolis, 1998).
37. P. K. Sweby, High resolution schemes using flux limiters for hyperbolic conservation laws, *SIAM J. Numer. Anal.* **21**, 995 (1984).
38. E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics* (Springer-Verlag, Berlin, 1997).

39. Q. H. Tran, *Schémas de Type Multidimensionnel en Maillage Déformé Structuré pour l'Advection Scalaire Linéaire. I: CTU et Galerkin Discontinu*, IFP Technical Report 45163 (Institut Français du Pétrole, Rueil-Malmaison, France, 1998).

40. Q. H. Tran, *Schémas de Type Multidimensionnel en Maillage Déformé Structuré pour l'Advection Scalaire Linéaire. II: Schémas compacts*, IFP Technical Report 54078 (Institut Français du Pétrole, Rueil-Malmaison, France, 2000).

41. B. Van Leer, Towards the ultimate conservative difference scheme V: A second-order sequel to Godunov's method, *J. Comput. Phys.* **32**, 101 (1979).

42. B. Van Leer, Multidimensional explicit difference schemes for hyperbolic conservation laws, in *Computing Methods in Applied Sciences and Engineering*, edited by R. Glowinski and J.-L Lions (Elsevier, North-Holland, 1984), Vol. VI, pp. 493–497.

43. H. Wang, R. E. Ewing, G. Qin, S. L. Lyons, M. Al-Lawatia, and S. Man, A family of Eulerian–Lagrangian localized adjoint methods for multi-dimensional advection reaction equations, *J. Comput. Phys.* **152**, 120 (1999).