SIAM J. SCI. COMPUT.
Vol. 18, No. 2, pp. 315–354, March 1997

MULTIRESOLUTION SCHEMES FOR THE NUMERICAL SOLUTION OF 2-D CONSERVATION LAWS I*

BARNA L. BIHARI† AND AMI HARTEN‡

Abstract. A generalization of Harten's multiresolution algorithms to two-dimensional (2-D) hyperbolic conservation laws is presented. Given a Cartesian grid and a discretized function on it, the method computes the local-scale components of the function by recursive diadic coarsening of the grid. Since the function's regularity can be described in terms of its scale or *multiresolution* analysis, the numerical solution of conservation laws becomes more efficient by eliminating flux computations wherever the solution is smooth. Instead, in those locations, the divergence of the solution is interpolated from the next coarser grid level. First, the basic 2-D essentially nonoscillatory (ENO) scheme is presented, then the 2-D multiresolution analysis is developed, and finally the subsequent scheme is tested numerically. The computational results confirm that the efficiency of the numerical scheme can be considerably improved in two dimensions as well.

Key words. multiresolution, ENO schemes, essentially nonoscillatory, regularity analysis, conservation laws

AMS subject classifications. 35A40, 35B65, 35L65, 65M06, 65M99

PII. S1064827594278848

1. Introduction.

Essentially nonoscillatory (ENO) schemes can now be routinely used to obtain high-order, practically oscillation-free solutions to conservation laws in one, two, and three dimensions [14], [13], [16], [12]. The main disadvantage of these schemes is their high cost. For most problems, however, using the expensive adaptive stencil everywhere is not necessary; this is justified only near discontinuities. Furthermore, the solution is also overresolved in much of the domain, so many of the flux computations themselves are redundant. Therefore, one approach for speedup is to automatically detect the regions of smoothness where resolution of the fluxes or divergence on coarser grids and their subsequent interpolation onto the finest grid is sufficient to achieve a prescribed accuracy.

The main idea of multiresolution schemes comes from the observation that the regularity of a function discretized on a set of hierarchical or nested grids can be described by the "difference in information" between two successive grid levels. That is, the difference between the original value and the value obtained by approximating the function discretized on a coarser level is proportional to the derivatives at the given location, and therefore it can be used to measure the function's degree of smoothness. These differences, or errors, are in fact the function's *local-scale components* which, when available, allow us to reconstruct it at finer levels exactly. On the other hand, by truncating sufficiently small components to zero, we can predict the function approximately, but using often significantly fewer components and therefore less storage. More importantly, in the context of numerical solution algorithms, the data compression can be applied to the numerical solution operator itself, thereby solving an approximate, sparse problem. Exactly where the original numerical method is used

*Received by the editors December 16,1994; accepted for publication July 23, 1995. Presented at the SIAM Annual Meeting, San Diego, CA, July 25–29, 1994.
http://www.siam.org/journals/sisc/18-2/27884.html
†Rockwell Science Center, Thousand Oaks, CA (blb@infinity.risc.rockwell.com).
‡The author is deceased. Former address: School of Mathematical Sciences, Tel-Aviv University, Israel.

is determined by the size of the aforementioned scale components computed on each level of resolution. Thus the name *multiresolution schemes*.

After its introduction in [7] and initial applications in [8], [6], and [4], multiresolution analysis was put in a more general framework, and the theoretical foundations for multiresolution representation of data and operators were established [10], [11], [9]. In the application area of hyperbolic conservation laws, various versions of the original multiresolution scheme exist, but the idea of using the scale coefficients to describe the regularity of the solution remains central to the algorithm [4], [6], [8]. Such numerical schemes were typically used in conjunction with the aforementioned ENO reconstruction, and while they proved to be significantly more efficient than conventional ENO schemes, they obtained results of the same high quality. The multiresolution analysis itself is completely independent of the numerical scheme, or the type of equation or problem we are solving. It is also potentially useful in image or data compression (its hierarchical nature has a close resemblance to wavelets) (e.g., [7]), matrix computations (see [15]); and in general in all areas where compression of the data or operators is possible and useful. For the numerical solution of conservation laws, application of multiresolution analysis to the "data compression" of ENO schemes seemed attractive, given the inherently high computational cost of these schemes.

In this work we present extensions of the semidiscrete version of the one-dimensional (1-D) multiresolution scheme [4] to 2-D hyperbolic conservation laws. The key concepts of the original multiresolution analysis introduced in [7] have a straightforward generalization to higher dimensions. Encoding and decoding for cell averages easily carry through from one dimension to two dimensions by using a tensor product approach. However, interpolation of numerical flux point values, as was done in one dimension, cannot be used in a robust and general manner. Instead, as one of the novel approaches introduced here, we interpolate the *numerical divergence*, in the sense of cell averages, from coarser levels. This method of predicting from coarse grid to fine also seems natural for unstructured grids and should easily generalize to three dimensions as well.

The higher dimensionality adds new intricacies to the truncation operation too, in practice as well as analysis. As bounds on the l_1 and l_∞ differences between the multiresolution solution and the original ENO solution can be established, we can be assured that the quality of the solution is not compromised by the multiresolution optimization. The choice of the reconstruction procedure also plays a more important role than in one dimension. We found that even to achieve acceptable data compression of the cell averages, the solution should be kept as oscillation free as possible. We shall briefly describe the 2-D ENO scheme, which can be generalized to be of arbitrary high order of accuracy, and is general enough for eventual use on unstructured grids as well (possibly coupled to the multiresolution representation presented in [1]). Our numerical results seem to support the available theory and analysis.

In section 2 we shall outline the semidiscrete numerical scheme without the multiresolution analysis; however, some of the choices with respect to the reconstruction and time stepping will be made to better fit the multiresolution modules that will be added later. Section 3 will explain the 2-D multiresolution analysis for point values and cell averages, as well as the regularity analysis and the subsequent truncation. We will then take the numerical scheme of section 2 and the multiresolution algorithms of section 3 and combine them into one scheme, a multiresolution scheme. In section 4 we describe the algorithm in detail. Finally, in section 5 numerical results

for first- and second-order reconstructions will be presented for both the linear and the nonlinear equations.

2. The 2-D numerical scheme.

2.1. The overall framework. In this paper we are concerned with solving the initial value problem for the scalar hyperbolic conservation law in two spatial dimensions:

$$u_t + f(u)_x + g(u)_y = 0, \tag{2.1a}$$

$$u(x, y, 0) = u_0(x, y), \tag{2.1b}$$

where f and g are the scalar flux functions. For simplicity, we shall assume that $x, y \in [-1, 1]$ and $u_0(x, y)$ is periodic in both the x and the y directions with period 2, so that we can use periodic boundary conditions at all boundaries.

The numerical scheme used is of the form

$$v^{n+1} = \mathbf{A} \cdot \mathbf{E}(\tau) \cdot \mathbf{R}(\cdot; v^n) \equiv \bar{\mathbf{E}}_h \cdot v^n, \tag{2.2}$$

where v^n are the cell averages (to be defined below), $\mathbf{A}$ represents the cell averaging operation, and $\mathbf{R}$ is the reconstruction operator applied to cell averages of the solution to obtain its point values. $\mathbf{E}$ is the exact time evolution operator so that

$$u(\cdot, t) = \mathbf{E}(t) \cdot u_0. \tag{2.3}$$

Then the finite volume scheme can be written in the conservative semidiscrete form:

$$(v_{j,k}(t))_t = -\frac{1}{h_x}(\bar{f}_{j+\frac{1}{2},k} - \bar{f}_{j-\frac{1}{2},k}) - \frac{1}{h_y}(\bar{g}_{j,k+\frac{1}{2}} - \bar{g}_{j,k-\frac{1}{2}}), \tag{2.4}$$

where h_x and h_y are the size of the (uniform) Cartesian grid cells in the x- and y-directions, respectively. $\bar{f}_{j\pm\frac{1}{2},k}$ are the numerical fluxes across the right and left faces of cell (j, k). Likewise, $\bar{g}_{j,k\pm\frac{1}{2}}$ are the numerical fluxes across the top and bottom cell faces of the cell with index (j, k). On the other hand, $v_{j,k}(t)$ is an approximation to the average of the exact solution $u(x, y, t)$ in the cell $[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}] \times [y_{k-\frac{1}{2}}, y_{k+\frac{1}{2}}], j = 0, ..., M-1$ and $k = 0, ..., N-1$:

$$v_{j,k}(t) \approx \frac{1}{h_x h_y} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \int_{y_{k-\frac{1}{2}}}^{y_{k+\frac{1}{2}}} u(x, y, t) dx dy,$$

with $t_n = n\tau, \tau = \Delta t$ and $h_x = \frac{2}{M}$, $h_y = \frac{2}{N}$, M and N being the number of cells in the x- and y-directions in the domain $[-1, 1] \times [-1, 1]$.

2.2. Numerical flux. A numerical flux F^* is an approximation to the contour integral along a curve segment C of the exact fluxes f and g obtained by solving the Riemann problem. That is,

$$F^* \approx \int_C F^R(u_l(s), u_r(s)) ds, \tag{2.5a}$$

where u_l and u_r are the values of the independent variable u on the left and on the right of C, respectively. $F^R(u_l, u_r)$ (here "R" stands for Riemann) is the solution

to the 1-D Riemann problem given the left and right values of u. For Cartesian coordinates we can write

$$F^R = \begin{cases} f(u^R) = f(u^R(u_l, u_r)), & \text{if } C \perp (0,1), \\ g(u^R) = g(u^R(u_l, u_r)), & \text{if } C \perp (1,0). \end{cases} \tag{2.5b}$$

Typically, for the equations of gas dynamics, an approximate Riemann solver is used. For our scalar prototype equations, there is no additional computational expense in using the exact solution of the Riemann problem, and, for completeness, we include it here.

(a) Linear wave equation (in two dimensions): $f(u)$ and $g(u)$ in (2.1) are

$$f(u) = g(u) = u; \tag{2.6a}$$

the F^R is therefore given by

$$F^R(u_l, u_r) = u_l. \tag{2.6b}$$

(b) Burgers's equation (in two dimensions): $f(u)$ and $g(u)$ in (2.1) are

$$f(u) = g(u) = \frac{1}{2}u^2, \tag{2.7a}$$

and F^R in (2.5a) is computed as in (2.5b), where u^R is defined by

$$u^R(u_l, u_r) = \begin{cases} u_r, & \text{if } c_l \le 0, c_r < 0, \\ u_l, & \text{if } c_l > 0, c_r \ge 0, \\ 0, & \text{if } c_l \le 0 \le c_r. \end{cases} \tag{2.7b}$$

The interpretation of c_l and c_r in this context is that of a "wave speed" immediately to the left and to the right of the discontinuity. In the case of Burgers's equation, the wave speed at a certain location coincides with the speed of a particle there. Information about the wave speed can be used to predict that particle's location at a future time. Waves can be either shocks or rarefaction fans; and for the latter, the entropy condition is used to find the unique solution to the Riemann problem. Hence we have the following definitions for $c_{l,r}$:

$$c_l = \begin{cases} \frac{u_l + u_r}{2}, & \text{if } u_l \ge u_r \text{ (shock)}, \\ u_l, & \text{otherwise (rarefaction)}; \end{cases} \tag{2.8a}$$

$$c_r = \begin{cases} \frac{u_l + u_r}{2}, & \text{if } u_l \ge u_r \text{ (shock)}, \\ u_r, & \text{otherwise (rarefaction)}. \end{cases} \tag{2.8b}$$

For our equally spaced Cartesian grid the contour integral of (2.5) reduces to the integral of either the flux f or the flux g, since the curve C is just the cell interface between two cells.We approximate this integral numerically by using a quadrature for which a Gaussian quadrature seems especially attractive because its accuracy is roughly double the number of evaluation points. Approximation (2.5a), when applied to both f and g as given in (2.5b), will yield

$$\bar{f}_{j\pm\frac{1}{2},k} = \sum_{i=1}^{n_q} \rho_i f(u^R(u_l^{y\pm}, u_r^{y\pm})), \tag{2.9a}$$

$$\bar{g}_{j,k\pm\frac{1}{2}} = \sum_{i=1}^{n_q} \rho_i g(u^R(u_l^{x\pm}, u_r^{x\pm})), \tag{2.9b}$$

where n_q is the number of Gaussian quadrature points used, ρ_i are the quadrature coefficients, and $u_l^{x,y\pm}$, $u_r^{x,y\pm}$ are the reconstructed values evaluated at the appropriate quadrature points using the polynomials to the left and to the right of the cell interface, respectively; here the superscripts x and y indicate the direction of change in the coordinate of the evaluation point. For n_q we have used

$$n_q \geq \left\lfloor \frac{r-1}{2} \right\rfloor + 1, \tag{2.9c}$$

where r is the order of the reconstruction and $\lfloor x \rfloor$ means the integer part of x. Since in this paper we are concerned only with first- and second-order ENO schemes, the more general sums of (2.9a,b) will reduce to only one term, i.e., $n_q = 1$, and we get the midpoint rule.

With these definitions, we can now define the *numerical divergence* $S_{j,k}$ which will also be occasionally referred to as the "right-hand side" (RHS), since it constitutes the RHS of the semidiscrete equation (2.4):

$$S_{j,k} = S_{j,k}(v(t)) = -\frac{1}{h_x}(\bar{f}_{j+\frac{1}{2},k} - \bar{f}_{j-\frac{1}{2},k}) - \frac{1}{h_y}(\bar{g}_{j,k+\frac{1}{2}} - \bar{g}_{j,k-\frac{1}{2}}). \tag{2.10}$$

(As it relates to (2.1a), it is actually $-S$ that approximates divergence, but in this context the above terminology was chosen as a matter of convenience.) Note that for each cell (j,k) the quantity S is a function, in general, of the entire v array.

2.3. ENO reconstruction. We turn now to describe the reconstruction procedure **R** of (2.2). The original ideas came from [12], and the arbitrary high-order implementations are described in [2] and [3]; here we briefly review the scheme specific to the first- and second-order cases.

For each cell we seek a multidimensional, piecewise polynomial, ENO interpolation that is uniformly rth-order accurate throughout the domain and is also conservative. That is, given cell averages $\bar{v} = \{\bar{v}_{j,k}\}$ of $v(x,y)$, we define the piecewise polynomial reconstruction R that satisfies the following three properties.

(a) Accuracy:

$$R(x,y;\bar{v}) = v(x,y) + O(h_x^r) + O(h_y^r). \tag{2.11a}$$

(b) Conservation:

$$A_{j,k}R(x,y;\bar{v}) = \bar{v}_{j,k}, \tag{2.11b}$$

where $A_{j,k}$ denotes the averaging operation

$$A_{j,k}w(x,y) = \frac{1}{h_x h_y}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}}\int_{y_{k-\frac{1}{2}}}^{y_{k+\frac{1}{2}}} w(x,y)dxdy. \tag{2.11c}$$

(c) ENO property: the stencil used for the reconstructed polynomial should correspond to the "smoothest" available (the selected stencil should not include discontinuities).

In other words, for each cell (j,k) there will be a polynomial $P_{j,k}(x,y)$ of degree $r-1$ whose coefficients will be chosen from a well-defined pool so as to guarantee an ENO reconstruction.

For example, we see that in the $r=1$ case the selection procedure is trivial, since then

$$P_{j,k}(x,y) = \bar{v}_{j,k} \qquad \forall\ 0 \le j \le M-1, \quad 0 \le k \le N-1. \tag{2.12}$$

This reconstruction, in fact, guarantees a TVD property in two dimensions as well, as defined in [5], and as such it gives an oscillation-free solution.

For the second-order ENO case each polynomial is a plane in three dimensions whose equation is given by

$$P_{j,k}(x,y) = \alpha^1_{j,k}x + \alpha^2_{j,k}y + \alpha^3_{j,k}. \tag{2.13}$$

For structured Cartesian grids, the coefficients $\alpha^i_{j,k}, i = 1,2,3$ are determined as follows.

(1) For each cell $\Phi_{j,k}$, we compute a least squares fit based on a 3×3 stencil centered at (j,k).

(2) For each cell $\Phi_{j,k}$ we identify the neighborhood $\bar{K}_{j,k}$ of cell neighbors. Then we compute the sum

$$\sigma_{j_0,k_0} = |\alpha^1_{j_0,k_0}| + |\alpha^2_{j_0,k_0}| \tag{2.14}$$

for each of the cells $\Phi_{j_0,k_0} \in \bar{K}_{j,k}$ and select the cell that Φ_{j_b,k_b} for which

$$\sigma_{j_b,k_b} = \min_{\Phi_{j_0,k_0} \in \bar{K}_{j,k}} \sigma_{j_0,k_0}. \tag{2.15}$$

In short, we select the cell whose polynomial's first derivatives have the minimal sum. Note that this method of selecting a least squares polynomial out of nine possible choices can, in fact, include influences from cells that are even four cells away from cell $\Phi_{j,k}$. This large choice of stencils will reduce the possibility of introducing new oscillations, but it will still yield a second-order reconstruction.

(3) The remaining coefficient $\alpha^3_{j,k}$ is computed in accordance to property (2.11b). Conservation will be satisfied if we require

$$\alpha^3_{j,k} = \bar{v}_{j,k} - \alpha^1_{j,k}x^c_{j,k} - \alpha^2_{j,k}y^c_{j,k}, \tag{2.16}$$

where the superscript "c" indicates the coordinate of the centroid.

2.4. Time integration. Using (2.10), we write the semidiscrete form (2.4) as

$$(v_{j,k}(t))_t = S_{j,k}(v(t)). \tag{2.17}$$

To integrate (2.17) in time for each $v_{j,k}$, we used a second-order Runge–Kutta method, subsequently abbreviated by RK2. The following RK2 algorithm corresponds to the midpoint method:

$$\begin{aligned} v^{n+\frac{1}{2}}_{j,k} &= v^n_{j,k} + \tfrac{1}{2}\tau S_{j,k}(v^n), \\ v^{n+1}_{j,k} &= v^n_{j,k} + \tau S_{j,k}(v^{n+\frac{1}{2}}). \end{aligned} \tag{2.18}$$

In the above, the symbolic $n + \frac{1}{2}$ index is used to denote an intermediate value of the cell average $v_{j,k}$, computed for the first stage of the RK2 method. The numerical divergence $S_{j,k}$ is a function of the entire v array, or at least some nontrivial subset thereof (in our notation, dropping the subscripts (j, k) from v means the entire 2-D array v). This semidiscrete formulation not only makes the programming easier but also makes it natural and efficient to perform multiresolution interpolation on the RHS S, as will be shown below.

3. The multiresolution analysis. We turn now to show the 2-D generalization of the multiresolution analysis introduced in [7] for point values and cell averages. Assume we have a uniform 2-D grid

$$G^0 = \{(x^0_{j,k}, y^0_{j,k})\} = \{(x_{j,k}, y_{j,k})\} \tag{3.1a}$$

on the square $[-1, 1] \times [-1, 1]$, where $j = 0, 1, ..., M - 1$ and $k = 0, 1, ..., N - 1$ with $M = M_0 = 2^{m_0}, N = N_0 = 2^{n_0}$. Let $h_x = h_{x,0} = \frac{2}{M_0}$ and $h_y = h_{y,0} = \frac{2}{N_0}$ be the cell width in the x- and y-directions, respectively. We construct a set of nested, diadically coarsened grids in the following recursive manner: given a grid G^{l-1} we obtain G^l by first removing the even k-index grid points along the constant j-lines; and then, on the resulting grid, eliminating the even j-index grid points along the constant k-lines. Then G^l will have grid spacings

$$h_{l_{x,y}} = 2h_{l-1_{x,y}}, \tag{3.2a}$$

where

$$h_{x,l} = \frac{2}{N_l}, h_{y,l} = \frac{2}{M_l} \tag{3.2b}$$

and

$$M_l = \frac{M_0}{2^l}, N_l = \frac{N_0}{2^l}. \tag{3.2c}$$

Therefore the following relationship also holds by construction:

$$G^l = \{(x^l_{j,k}, y^l_{j,k})\} = \{(x^{l-1}_{2j+1,2k+1}, y^{l-1}_{2j+1,2k+1})\}, \tag{3.3}$$

where $0 \le j \le M_l - 1, 0 \le k \le N_l - 1$. Suppose we continue this coarsening process L times until we reach a minimal granularity $M_L \times N_L$ on the grid G^L. Then, on this hierarchical set of grids $\{G^l\}^L_{l=0}$ we can define the 2-D multiresolution analyses for point values and cell averages, concepts we shall later incorporate into the numerical scheme previously defined in order to design a "faster" numerical scheme, which is termed *multiresolution scheme*.

3.1. Multiresolution analysis for 2-D point values. Let $u^0_{j,k} = u^0(x_{j,k}, y_{j,k})$, $0 \le j \le M_0 - 1, 0 \le k \le N_0 - 1$ be a grid function defined by its point values at each grid point on G^0. We take u^0 to be periodic with period 2 in both the x- and y-directions. A construction analogous to the one used for the grid G^0 itself can be done naturally on u^0, too. Given u^l we then have

$$u^l_{j,k} = u^{l-1}_{2j+1,2k+1} \quad \text{for} \quad 0 \le j \le M_l - 1, 0 \le k \le N_l - 1, \tag{3.4}$$

and thus to construct u^l all values of u^{l-1} whose j- or k-index is even are discarded.

Next, we define an interpolation operation $I(x, y; u^l)$ which we shall use to approximate the previously discarded values of u^{l-1} from the sparser u^l array. To achieve maximal accuracy we use a central interpolation; we can also take advantage of the equally spaced grid and thus have constant symmetric coefficients. Hence I will be restricted to even orders of accuracy. The interpolation will take different forms for locations with the following (j, k) index pairs:

(odd,odd): exact

$$u^{l-1}_{2j+1,2k+1} = I_1(x^{l-1}_{2j+1,2k+1}, y^{l-1}_{2j+1,2k+1}; u^l) = u^l_{j,k}, \tag{3.5a}$$

(even,odd): interpolation along j

$$u^{l-1}_{2j,2k+1} \approx I_2(x^{l-1}_{2j,2k+1}, y^{l-1}_{2j,2k+1}; u^l) = \sum_{i=1}^{s} \beta_i (u^l_{j+i-1,k} + u^l_{j-i,k}), \tag{3.5b}$$

(odd,even): interpolation along k

$$u^{l-1}_{2j+1,2k} \approx I_3(x^{l-1}_{2j+1,2k}, y^{l-1}_{2j+1,2k}; u^l) = \sum_{m=1}^{s} \beta_m (u^l_{j,k+m-1} + u^l_{j,k-m}), \tag{3.5c}$$

(even,even): tensor product interpolation in j and k

$$\begin{aligned} u^{l-1}_{2j,2k} \approx & I_4(x^{l-1}_{2j,2k}, y^{l-1}_{2j,2k}; u^l) \\ = & \sum_{i=1}^{s} \beta_i \sum_{m=1}^{s} \beta_m (u^{l-1}_{j+i-1,k+m-1} + u^{l-1}_{j-i,k+m-1} \\ & + u^l_{j+i-1,k-m} + u^l_{j-i,k-m}). \end{aligned} \tag{3.5d}$$

In equations (3.5a–d) the coefficients $\beta_i, i = 1, 2, ..., s$ come from standard interpolation results; $r = 2s$, where r is the order of the interpolation. For a few common values of r we have

$$\begin{aligned} &\text{if } r = 2, \text{ then } \beta_1 = \tfrac{1}{2}; \\ &\text{if } r = 4, \text{ then } \beta_1 = \tfrac{9}{16}, \beta_2 = -\tfrac{1}{16}; \\ &\text{if } r = 6, \text{ then } \beta_1 = \tfrac{150}{256}, \beta_2 = -\tfrac{25}{256}, \beta_3 = \tfrac{3}{256}. \end{aligned} \tag{3.6}$$

We can now define the 2-D equivalents of the *encoding* and *decoding* procedures first presented in [7]. By summarizing equations (3.5a–d) into

$$I(x^{l-1}_{j,k}, y^{l-1}_{j,k}; u^l) = \begin{cases} I_1(x^{l-1}_{j,k}, y^{l-1}_{j,k}; u^l), & \text{if } j = 2j_0 + 1, k = 2k_0 + 1, \\ I_2(x^{l-1}_{j,k}, y^{l-1}_{j,k}; u^l), & \text{if } j = 2j_0\ , k = 2k_0 + 1, \\ I_3(x^{l-1}_{j,k}, y^{l-1}_{j,k}; u^l), & \text{if } j = 2j_0 + 1, k = 2k_0, \\ I_4(x^{l-1}_{j,k}, y^{l-1}_{j,k}; u^l), & \text{if } j = 2j_0\ , k = 2k_0 \end{cases} \tag{3.7}$$

for all $0 \le j \le M_{l-1} - 1,\ 0 \le k \le N_{l-1} - 1$, where $0 \le j_0 \le M_l - 1,\ 0 \le k_0 \le N_l - 1$, the encoding $\mathbf{M}$ and decoding $\mathbf{M}^{-1}$ operations take a rather simple form.

Encoding ($\mathbf{M}$):

DO for $l = 1, 2, ..., L$

DO for $j = 0, 1, ..., M_l - 1; \quad k = 0, 1, ..., N_l - 1$

$$u^l_{j,k} = u^{l-1}_{2j+1,2k+1}$$

ENDDO

DO for $j_0 = 0, 1, ..., M_{l-1} - 1;\ k_0 = 0, 1, ..., N_{l-1} - 1$

$$d^{l-1}_{j_0,k_0} = u^{l-1}_{j_0,k_0} - I(x_{j_0,k_0}, y_{j_0,k_0}; u^l)$$

ENDDO

(3.8) **ENDDO**

Decoding ($\mathbf{M}^{-1}$):

DO for $l = L, L-1, ..., 1$

DO for $j_0 = 0, 1, ..., M_{l-1} - 1; \quad k_0 = 0, 1, ..., N_{l-1} - 1$

$$u^{l-1}_{j_0,k_0} = d^{l-1}_{j_0,k_0} + I(x_{j_0,k_0}, y_{j_0,k_0}; u^l)$$

ENDDO

(3.9) **ENDDO**

The multiresolution analysis of u^0 is then defined as

$$u_{\mathbf{M}} = (d^0, d^1, ..., d^{L-1}, u^L)^T = \mathbf{M}u^0. \tag{3.10}$$

Remark 3.1. Note that operation I in (3.8) includes one simple "copying" for each three interpolations, as indicated by (3.7). Therefore, on the coarsest level L, u^L will have some select exact values of u^0. This coarsest u^L array will be used as the starting point for the reverse order iteration in (3.9), at the end of which we shall indeed recover u^0 exactly. Namely, $\mathbf{M}$ is an invertible operation and its inverse is $\mathbf{M}^{-1}$.

Remark 3.2. $\mathbf{M}$ and $\mathbf{M}^{-1}$ are in fact matrices. Since $d^l_{j,k} = 0$ if $j = 2j_0 + 1, k = 2k_0 + 1$, we can pack only the nonzero d's and all of the u^L array into the array $u_{\mathbf{M}}$ (as defined by (3.10)) so that its size will be $M_0 \times N_0$, which is the size of the original u^0. It can therefore be shown that the multiresolution representation $\mathbf{M}u^0$ requires no additional storage. In the current work, however, we seek to emphasize the advantage offered by multiresolution analysis with respect to operation count and will defer the use of more sophisticated data structures to a future paper.

Remark 3.3. The 1-D counterparts [7] of algorithms (3.8) and (3.9) interpolate from coarser levels at every other grid point and copy at the other half of the locations. In two dimensions, using the previously described hierarchical set of grids $\{G^l\}_{l=0}^L$, the copying is done one out of four times.

3.2. Multiresolution analysis for 2-D cell averages. If we interpret now

$$\bar{u}^0_{j,k} = \frac{1}{h_x h_y} \int_{x_{j,k}}^{x_{j+1,k}} \int_{y_{j,k}}^{y_{j,k+1}} u(x, y) dx dy \tag{3.11}$$

for $0 \le j \le M_0 - 1, 0 \le k \le N_0 - 1$, as cell averages of $u(x, y)$ over grid G^0 of (3.1a), another type of multiresolution analysis can be defined: that of cell averages. At an

arbitrary level l and on a grid G^l of our hierarchical sequence, (3.11) will generalize to

$$\bar{u}^l_{j,k} = \frac{1}{h_{x,l}h_{y,l}} \int_{x^l_{j,k}}^{x^l_{j+1,k}} \int_{y^l_{j,k}}^{y^l_{j,k+1}} u(x,y)dxdy, \tag{3.12}$$

where $0 \le j \le M_l - 1, 0 \le k \le N_l - 1$.

By the additivity of integrals in (3.12), we have the following relationship between cell averages on adjacent levels:

$$\bar{u}^l_{j,k} = \frac{1}{4}(\bar{u}^{l-1}_{2j,2k} + \bar{u}^{l-1}_{2j+1,2k} + \bar{u}^{l-1}_{2j,2k+1} + \bar{u}^{l-1}_{2j+1,2k+1}). \tag{3.13}$$

For cell averages, the decimation process can then be defined by (3.13).

Conversely, to recover $\bar{u}^{l-1}$ from $\bar{u}^l$ (approximately), we interpolate point values of the primitive function of u in a tensor product sense. This yields the following formulas for the cell averages with the four types of index pairs:

(odd,odd):

$$\begin{aligned} \bar{u}^{l-1}_{2j+1,2k+1} &\approx \bar{I}(2j+1, 2k+1; \bar{u}^l) \\ &= \bar{u}^l_{j,k} + Q^s_x(j,k;\bar{u}^l) + Q^s_y(j,k;\bar{u}^l) + Q^s_{xy}(j,k;\bar{u}^l), \end{aligned} \tag{3.14a}$$

(even,odd):

$$\begin{aligned} \bar{u}^{l-1}_{2j,2k+1} &\approx \bar{I}(2j, 2k+1; \bar{u}^l) \\ &= \bar{u}^l_{j,k} - Q^s_x(j,k;\bar{u}^l) + Q^s_y(j,k;\bar{u}^l) - Q^s_{xy}(j,k;\bar{u}^l), \end{aligned} \tag{3.14b}$$

(odd,even):

$$\begin{aligned} \bar{u}^{l-1}_{2j+1,2k} &\approx \bar{I}(2j+1, 2k; \bar{u}^l) \\ &= \bar{u}^l_{j,k} + Q^s_x(j,k;\bar{u}^l) - Q^s_y(j,k;\bar{u}^l) - Q^s_{xy}(j,k;\bar{u}^l), \end{aligned} \tag{3.14c}$$

(even,even):

$$\begin{aligned} \bar{u}^{l-1}_{2j,2k} &\approx \bar{I}(2j, 2k; \bar{u}^l) \\ &= \bar{u}^l_{j,k} - Q^s_x(j,k;\bar{u}^l) - Q^s_y(j,k;\bar{u}^l) + Q^s_{xy}(j,k;\bar{u}^l), \end{aligned} \tag{3.14d}$$

where

$$Q^s_x(j,k;\bar{u}^l) = \sum_{i=1}^{s-1} \gamma_i(\bar{u}^l_{j+i,k} - \bar{u}^l_{j-i,k}), \tag{3.15a}$$

$$Q^s_y(j,k;\bar{u}^l) = \sum_{m=1}^{s-1} \gamma_m(\bar{u}^l_{j,k+m} - \bar{u}^l_{j,k-m}), \tag{3.15b}$$

$$\begin{aligned} Q^s_{xy}(j,k;\bar{u}^l) = \sum_{i=1}^{s-1} \gamma_i \sum_{m=1}^{s-1} \gamma_m(&\bar{u}^l_{j+i,k+m} - \bar{u}^l_{j+i,k-m} \\ &- \bar{u}^l_{j-i,k+m} + \bar{u}^l_{j-i,k-m}). \end{aligned} \tag{3.15c}$$

For the γ's we use the interpolation results from one dimension, which are

$$\begin{aligned} &\text{if } \bar{r} = 3, \text{ then } \gamma_1 = -\tfrac{1}{8}; \\ &\text{if } \bar{r} = 5, \text{ then } \gamma_1 = -\tfrac{22}{128}, \gamma_2 = \tfrac{3}{128}. \end{aligned} \tag{3.16}$$

Here $\bar{r}$ is the order of the approximation, with $\bar{r} = 2s - 1$.

Unlike the interpolation for point values, we have the same interpolation for the four types of cells, since $\bar{I}$ is the same polynomial, but it is evaluated at different locations. The stencil of cells is also the same for all four cells, whereas, in the case of point values, we had only one 2-D interpolation, two were degenerate 1-D interpolations, and one was an exact value.

To obtain the multiresolution analysis of a 2-D cell average array $\bar{u}^0$ and, in turn, to get back the cell averages $\bar{u}^0$ on the finest level $l = 0$ from its multiresolution representation $\bar{u}_{\bar{\mathbf{M}}}$, we define the following two procedures:

Encoding ($\bar{\mathbf{M}}$):

DO for $l = 1, 2, ..., L$

DO for $j = 0, 1, ..., M_l - 1; \quad k = 0, 1, ..., N_l - 1$

$\bar{u}^l_{j,k} = \frac{1}{4}(\bar{u}^{l-1}_{2j,2k} + \bar{u}^{l-1}_{2j+1,2k} + \bar{u}^{l-1}_{2j,2k+1} + \bar{u}^{l-1}_{2j+1,2k+1})$

ENDDO

DO for $j_0 = 0, 1, ..., M_{l-1} - 1; \; k_0 = 0, 1, ..., N_{l-1} - 1$

$\bar{d}^{l-1}_{j_0,k_0} = \bar{u}^{l-1}_{j_0,k_0} - \bar{I}(j_0, k_0; \bar{u}^l)$

ENDDO

ENDDO (3.17)

Decoding ($\bar{\mathbf{M}}^{-1}$):

DO for $l = L, L-1, ..., 1$

DO for $j_0 = 0, 1, ..., M_{l-1} - 1; \; k_0 = 0, 1, ..., N_{l-1} - 1$

$\bar{u}^{l-1}_{j_0,k_0} = \bar{d}^{l-1}_{j_0,k_0} + \bar{I}(j_0, k_0; \bar{u}^l)$

ENDDO

ENDDO (3.18)

The multiresolution analysis of $\bar{u}^0$ is then defined as

$$\bar{u}_{\bar{\mathbf{M}}} = (\bar{d}^0, \bar{d}^1, ..., \bar{d}^{L-1}, \bar{u}^L)^T = \bar{\mathbf{M}}\bar{u}^0. \tag{3.19}$$

Remark 3.4. The interpolation $\bar{I}$ of (3.17), as defined by (3.14a–d) and (3.15a–c), has to be performed for only three of the four index pairs of (3.14a–d). The fourth one can be computed using (3.13). For one dimension, in contrast, we had to interpolate for one out of every two cells (see [8]).

Remark 3.5. As in the case of point values, the multiresolution analysis (3.19) for cell averages can be done without requiring any additional storage. Because of (3.13), each (four times) coarser array of independent $\bar{d}$'s can be recursively fit into one-fourth of the finer array. Finally, on the coarsest level, the array $\bar{u}^L$ will take the place of the redundant $\bar{d}^L$'s.

3.3. Regularity analysis. We turn now to show how the multiresolution analysis for cell averages is used to characterize the regularity of the solution. The idea behind multiresolution schemes is to only use the direct flux evaluations with ENO reconstruction wherever it is indeed necessary: near discontinuities. At all other localities the fluxes, or the RHS of (2.10), do not have to be evaluated but can be interpolated from coarser grid levels using a multiresolution decoding procedure. The key, therefore, is to detect the discontinuities. This has been done successfully in one dimension using the multiresolution analysis of cell averages (see [8],[4]); in this section we extend those concepts to two dimensions.

Suppose we have a function $u(x,y)$ and its cell averages $\bar{u}^0$ as defined by (3.11). Assume further that $u(x,y)$ has $p-1$ and $q-1$ continuous partial derivatives with respect to x and y, respectively, and a jump discontinuity in the pth x-derivative and the qth y-derivative at location $(x^\star, y^\star)$. Then, by using 1-D and 2-D (tensor product) interpolation results, we get from (3.15a–c),

$$\begin{aligned} e_1 &= Q^s_{xy}(j,k;\bar{u}^l) + \frac{1}{4}(\bar{u}^{l-1}_{2j,2k} - \bar{u}^{l-1}_{2j+1,2k} - \bar{u}^{l-1}_{2j,2k+1} + \bar{u}^{l-1}_{2j+1,2k+1}) \\ &\sim T_x + T_y, \end{aligned} \tag{3.20a}$$

where, using constants c_1, c_2, c_3, c_4, we have

$$T_x = \begin{cases} c_1 h^{\bar{r}+1}_{x,l} \frac{\partial^{\bar{r}+1} u}{\partial x^{\bar{r}+1}} + c_2 h^{\bar{r}}_{x,l} h_{y,l} \frac{\partial^{\bar{r}+1} u}{\partial y \partial x^{\bar{r}}}, & \text{if } p > \bar{r}, \\ c_1 h^{p+1}_{x,l} \left[\frac{\partial^{p+1} u}{\partial x^{\bar{r}+1}}\right] + c_2 h^{p}_{x,l} h_{y,l} \left[\frac{\partial^{p+1} u}{\partial y \partial x^{p}}\right], & \text{if } p \le \bar{r}, \end{cases} \tag{3.20b}$$

$$T_y = \begin{cases} c_3 h^{\bar{r}+1}_{y,l} \frac{\partial^{\bar{r}+1} u}{\partial y^{\bar{r}+1}} + c_4 h_{x,l} h^{\bar{r}}_{y,l} \frac{\partial^{\bar{r}+1} u}{\partial x \partial y^{\bar{r}}}, & \text{if } q > \bar{r}, \\ c_3 h^{q+1}_{y,l} \left[\frac{\partial^{q+1} u}{\partial x^{q+1}}\right] + c_4 h_{x,l} h^{q}_{y,l} \left[\frac{\partial^{q+1} u}{\partial x \partial y^{q}}\right], & \text{if } q \le \bar{r}. \end{cases} \tag{3.20c}$$

In the above, the brackets [ ] denote the jump discontinuity, and the superscripts involving $\bar{r}$, p, or q mean powers of the particular base.

In addition,

$$\begin{aligned} e_2 &= Q^s_x(j,k;\bar{u}^l) + \frac{1}{4}(-\bar{u}^{l-1}_{2j,2k} + \bar{u}^{l-1}_{2j+1,2k} - \bar{u}^{l-1}_{2j,2k+1} + \bar{u}^{l-1}_{2j+1,2k+1}) \\ &\sim \begin{cases} h^{\bar{r}}_{x,l} \frac{\partial^{\bar{r}} u}{\partial x^{\bar{r}}}, & \text{if } p > \bar{r}, \\ h^{p}_{x,l} \left[\frac{\partial^{p} u}{\partial x^{p}}\right], & \text{if } p \le \bar{r}, \end{cases} \end{aligned} \tag{3.21}$$

$$\begin{aligned} e_3 &= Q^s_y(j,k;\bar{u}^l) + \frac{1}{4}(-\bar{u}^{l-1}_{2j,2k} - \bar{u}^{l-1}_{2j+1,2k} + \bar{u}^{l-1}_{2j,2k+1} + \bar{u}^{l-1}_{2j+1,2k+1}) \\ &\sim \begin{cases} h^{\bar{r}}_{y,l} \frac{\partial^{\bar{r}} u}{\partial y^{\bar{r}}}, & \text{if } q > \bar{r}, \\ h^{q}_{y,l} \left[\frac{\partial^{q} u}{\partial y^{q}}\right], & \text{if } q \le \bar{r}. \end{cases} \end{aligned} \tag{3.22}$$

See Appendix A for a detailed derivation of equations (3.20–3.22). We can also define an $e_4 = -e_1 - e_2 - e_3$ whose magnitude will simply be the sum of the e_is, $i = 1, 2, 3$. In equations (3.20), (3.21), and (3.22) the derivatives and jumps are evaluated at a point $\xi^\star$ which is within $h_{x,l}$ and $h_{y,l}$ of the point $(x^\star, y^\star)$; we left the evaluation argument off for brevity of notation.

As shown by (3.20), (3.21), and (3.22), the size of or the jump in the x- and y-derivatives can be estimated by the e_is. In addition, we observe that if we set

$$h_l = \max(h_{x,l}, h_{y,l}), \tag{3.23a}$$

$$\bar{p} = \min(p, q, \bar{r}), \tag{3.23b}$$

then we can write the following:

$$e_1 = O(h_l^{\bar{p}+1}), \tag{3.24a}$$

$$e_2 = O(h_l^{\bar{p}}), \tag{3.24b}$$

$$e_3 = O(h_l^{\bar{p}}), \tag{3.24c}$$

$$e_4 = O(h_l^{\bar{p}}). \tag{3.24d}$$

Because of the above analysis, it makes sense to use the e_is as the "regularity sensors" instead of the $\bar{d}$s of (3.17) and (3.18) directly. This constitutes one of the changes that are implemented in the generalization process from the 1-D multiresolution analysis to its 2-D counterpart. There is, of course, a linear relationship between the two sets, given below (implied by (3.14a–d), (3.15a–c), (3.20a–b), (3.21), and (3.22)):

$$e_1 = \frac{1}{4}(\bar{d}^{l-1}_{2j,2k} - \bar{d}^{l-1}_{2j+1,2k} - \bar{d}^{l-1}_{2j,2k+1} + \bar{d}^{l-1}_{2j+1,2k+1}), \tag{3.25a}$$

$$e_2 = \frac{1}{4}(-\bar{d}^{l-1}_{2j,2k} + \bar{d}^{l-1}_{2j+1,2k} - \bar{d}^{l-1}_{2j,2k+1} + \bar{d}^{l-1}_{2j+1,2k+1}), \tag{3.25b}$$

$$e_3 = \frac{1}{4}(-\bar{d}^{l-1}_{2j,2k} - \bar{d}^{l-1}_{2j+1,2k} + \bar{d}^{l-1}_{2j,2k+1} + \bar{d}^{l-1}_{2j+1,2k+1}), \tag{3.25c}$$

$$e_4 = -e_1 - e_2 - e_3. \tag{3.25d}$$

Equations (3.25a–d) are, in fact, what we used to compute the regularity coefficients. In practice we do not store the original $\bar{d}$s but store the e_is instead, since the latter will be then used for regularity analysis and subsequent truncation, and they take up the same amount of storage. For the rest of the paper, we shall therefore use the following notation for the regularity coefficients:

$$\bar{e}^l_{j,k} = \begin{cases} e_1, & \text{if } j = 2j_0, k = 2k_0, \\ e_2, & \text{if } j = 2j_0 + 1, k = 2k_0, \\ e_3, & \text{if } j = 2j_0, k = 2k_0 + 1, \\ e_4, & \text{if } j = 2j_0 + 1, k = 2k_0 + 1 \end{cases} \tag{3.26}$$

$\forall 0 \le j \le M_l - 1, 0 \le k \le N_l - 1$, where $0 \le j_0 \le M_{l+1} - 1, 0 \le k_0 \le N_{l+1} - 1$. Thus we arrive at a multiresolution analysis of $\bar{u}^0$:

$$\bar{u}_{\bar{\mathbf{M}}'} = (\bar{e}^0, \bar{e}^1, ..., \bar{e}^{L-1}, \bar{u}^L)^T = \bar{\mathbf{M}}'\bar{u}^0, \tag{3.27}$$

which is equivalent to (3.19).

To exploit the large amount of information hidden in (3.20a–c), (3.21), and (3.22), we argue as in [8]. Comparing the size of the regularity coefficients from adjacent grid levels $l-1$ and l, we can estimate the order of the effective smoothness $\bar{p}$. If two cells (j,k) and (j_0,k_0) (on the coarser and finer grids G^l and G^{l-1}, respectively) are close enough to the point $(x^\star, y^\star)$, then because of (3.2a),

$$\bar{e}^{l-1}_{j_0,k_0} \approx 2^{-\bar{p}} \bar{e}^{l}_{j,k}, \tag{3.28}$$

where now $0 \le j \le M_l - 1,\ 0 \le k \le N_l - 1$, $0 \le j_0 \le M_{l-1} - 1,\ 0 \le k_0 \le N_{l-1} - 1$, j_0 is of the same parity as j, and k_0 is of the same parity as k. A closer examination of (3.20–3.22) reveals that near discontinuities the coefficients $\bar{e}$ remain of the same size, regardless of which grid level they are computed on, and thus we conclude that $\bar{p}$ is zero, or of a low value. In smooth regions, on the other hand, they diminish in size as a power of two, where the exponent $\bar{p} > 0$, or even $\bar{p} > \bar{r}$.

We would like to point out the importance of performing multiresolution analysis on the cell averages of the solution, rather than its reconstructed point values or the point values of the fluxes. As suggested in [7] by Harten, there is a significant advantage in using cell averages for regularity analysis of a function that is allowed to have a finite number of jump discontinuities. This was already the case in one dimension (see [8], [4], and [6]). In two dimensions the argument for cell averages is further supported by the fact that the interpolation I of (3.7) for point values is not "symmetric" for the four points on the new level in the sense that it contains degenerate cases. All four interpolating polynomials $I_1, ..., I_4$ can still be thought of as one polynomial that is evaluated at locations which are special cases. The fact that three out of four locations are along integral isoparametric curves in the tensor product formulation will, however, result in a contribution only from a 1-D stencil of flux values. In addition, the stencil will not be entirely central to three of these four interpolation locations. The cell average interpolation $\bar{I}$ (3.14a–d), on the other hand, uses one stencil and one polynomial. The stencil is also symmetric for the four interpolation locations. For the same reason, as will be seen in later sections, even the selective "interpolation versus direct computation" decision for fluxes will be replaced by that of the RHS.

Remark 3.6. According to some of our numerical experiments, the multiresolution coefficients $\bar{d}$ could also be used successfully for regularity analysis, but the compression rates were lower overall. We can qualitatively attribute this to the fact that in general $\bar{d}^l = O(h_l^{\bar{p}})$, but $e_1 = O(h_l^{\bar{p}+1})$, and thus the use of the latter allows us to detect and avoid discontinuities in derivatives that are of one order higher.

3.4. Truncation. Using the regularity information obtained above, we now design the *truncation* procedure for two dimensions. First, define the following truncation function:

$$\mathrm{tr}(x_1, x_2, x_3, x_4, \epsilon) = \begin{cases} 0, & \text{if } |x_2|, |x_3|, c|x_4| \le \epsilon, \\ x_1, & \text{otherwise.} \end{cases} \tag{3.29}$$

In the above $c > 0$ is a predefined constant, and the first branch reflects the fact that all of the second, third, and fourth arguments must fall below a specified value for truncation to occur. Appendix A shows details on how the truncation errors of (3.20b–c) are derived, and it also suggests that the coefficients of the $\bar{p}$th-order terms are different from the $(\bar{p}+1)$st-order terms. The constant c reflects this fact; for the $\bar{r} = 3$ computations we used $c = 3$. Note also that the order of the arguments is

significant in (3.29). Let now

$$\hat{e}^l_{j,k} = \begin{cases} \mathrm{tr}(\bar{e}^l_{j,k}, \bar{e}^l_{j+1,k}, \bar{e}^l_{j,k+1}, \bar{e}^l_{j,k}, \epsilon_l) & \text{if } j = 2j_0, k = 2k_0, \\ \mathrm{tr}(\bar{e}^l_{j,k}, \bar{e}^l_{j,k}, \bar{e}^l_{j-1,k+1}, \bar{e}^l_{j-1,k}, \epsilon_l) & \text{if } j = 2j_0+1, k = 2k_0, \\ \mathrm{tr}(\bar{e}^l_{j,k}, \bar{e}^l_{j,k}, \bar{e}^l_{j+1,k-1}, \bar{e}^l_{j,k-1}, \epsilon_l) & \text{if } j = 2j_0, k = 2k_0+1, \\ \mathrm{tr}(\bar{e}^l_{j,k}, \bar{e}^l_{j-1,k}, \bar{e}^l_{j,k-1}, \bar{e}^l_{j-1,k-1}, \epsilon_l) & \text{if } j = 2j_0+1, k = 2k_0+1 \end{cases} \tag{3.30}$$

for all $0 \le j \le M_l - 1, \quad 0 \le k \le N_l - 1$. In other words, we require that all members of the triplet e_1, e_2, e_3 (as defined by (3.20–3.22)) be small for any member of the quadruplet e_1, e_2, e_3, e_4 to be truncated. Note that e_1, e_2, e_3 being small implies e_4 is small as well. Thus the test is performed only once for each quadruplet; by design, passing the test means that the approximations to the derivatives diminish in size upon refinement, implying that the solution is "smooth enough."

The operator form of the truncation operation can be written as

$$\hat{u}_{\bar{\mathbf{M}}'} = \mathbf{tr}(\bar{u}_{\bar{\mathbf{M}}'}) = (\hat{e}^0, \hat{e}^1, ..., \hat{e}^{L-1}, \bar{u}^L)^T. \tag{3.31}$$

That is, $\hat{u}_{\bar{\mathbf{M}}'}$ is the truncated multiresolution analysis of $\bar{u}^0$, so $\bar{u}^0$ will be available approximately by applying the decoding to $\hat{u}_{\bar{\mathbf{M}}'}$:

$$\bar{u}^0 \approx \hat{u}^0 = \bar{\mathbf{M}}'^{-1} \hat{u}_{\bar{\mathbf{M}}'} = \bar{\mathbf{M}}'^{-1} \mathbf{tr} \bar{\mathbf{M}}' \bar{u}^0. \tag{3.32}$$

The truncation thresholds ϵ_l still remain to be determined. Here we adopted the choice introduced in [8]:

$$\epsilon_l = 2^{-l}\epsilon, \tag{3.33}$$

where ϵ is a tolerance specified by the user. Namely, on coarser levels we allow smaller errors, and as we go to finer levels the truncation becomes less restrictive. If $\epsilon = 0$, then of course $\hat{u}^0 = \bar{u}^0$ and the original $\bar{u}^0$ array is recovered exactly.

Next, we define sets $\bar{D}^l$ of cell indices (j, k, l) for which $\hat{e}^l_{j,k} \neq 0$:

$$\bar{D}^l = \{(j,k,l) | \quad |\hat{e}^l_{j,k}| > \epsilon_l\}, \tag{3.34a}$$

where l is the current grid level, and, as usual, $0 \le j \le M_l - 1, 0 \le k \le N_l - 1$. The cumulative set

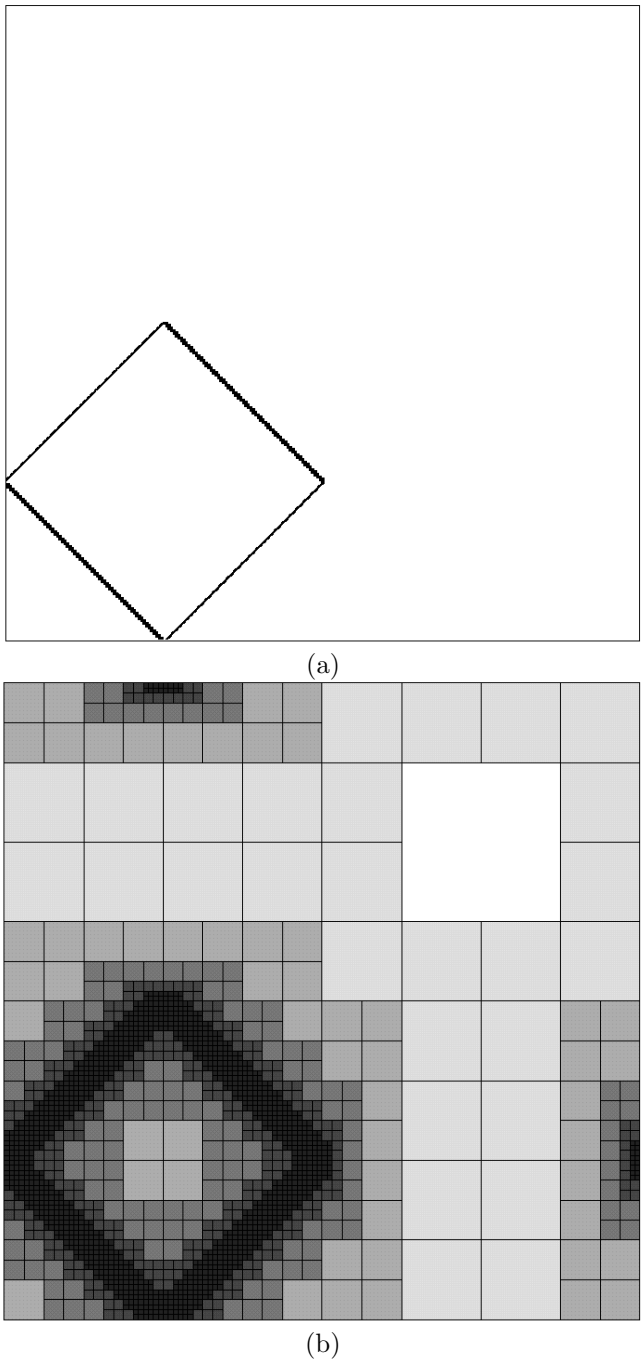
$$\bar{D} = \cup_{l=0}^{L-1} \bar{D}^l \tag{3.34b}$$

contains all the cells, on all grid levels, where the solution is not regular. These are the cells for which exact fluxes will be computed, and at the remaining cells interpolation from coarser levels will be sufficient.

Another set D is defined as the set of those cell faces on the fine grid for which fluxes have to be computed. These faces belong to one of the cells, on any resolution level, that is flagged for exact computation, i.e., that belongs to $\bar{D}$. If we shade the cells in $\bar{D}^l$ and interpret the resulting set as a 2-D domain and the fluxes at its edges as its boundaries, we can write

$$|D| = |\bar{D}| + |\cup_{l=0}^{L-1} \partial \bar{D}^l|. \tag{3.35}$$

For example, for the rotated step function shown in Fig. 1a, we get a set $\bar{D}$ as shown in Fig. 1b. The different grey scales here mean "flagged" cells at different grid levels.



(a)

(b)

FIG. 1. *Rotated 2-D step function and its multiresolution diagram.*

Darker means finer grids, lighter means coarser. In Fig. 1b we show five different levels of resolution. If we also assign a height to each cell in $\bar{D}$, we can visualize the multiresolution diagram as a 3-D plot, as in Fig. 2, which can be considered a sort of a generalized version of the 1-D multiresolution diagram first shown in [8]. Figure 1b shows the multiresolution diagram of Fig. 2 from a top view.

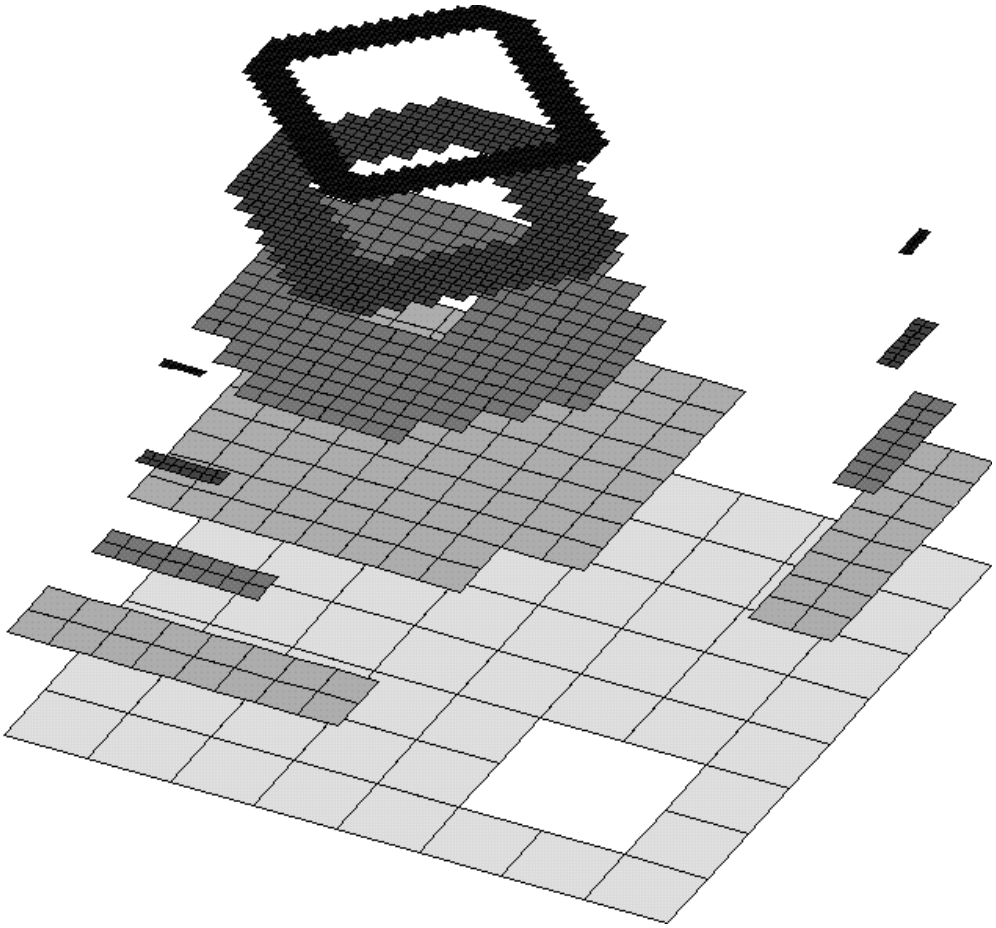


FIG. 2. *3-D view of multiresolution diagram for rotated 2-D step function.*

As a measure of the speedup obtained by skipping many of the direct flux evaluations, we define the *efficiency* μ:

$$\mu = \frac{M_0 N_0}{\frac{M_0 N_0}{4^L} + |D|}. \tag{3.36}$$

The numerator contains the number of flux evaluations that would have to be done without a multiresolution scheme, and the denominator reflects the total number of direct fluxes computed using the multiresolution scheme. On the coarsest grid G^l, all fluxes are directly evaluated: this is the first term of the denominator in (3.36). The second term of the denominator reflects the faces where fluxes are computed; for many problems with discontinuities, $|D|$ is close to $|\bar{D}|$.

4. The multiresolution scheme. In this section we combine the numerical scheme outlined in section 3 with the multiresolution analysis described in section 4 in order to devise a new numerical scheme that preserves the quality of the solution but is much more efficient. Our basic assumptions are that interpolating numerical divergence from coarser grids is considerably faster than directly computing the necessary fluxes and that the solution has few discontinuities. This seems to be a

reasonable starting point in light of the fact that nonlinear conservation laws can give rise to discontinuous solutions, but with usually few discontinuities. Since ENO reconstructions can be quite expensive, fixed stencil central interpolation from coarser grid levels significantly reduces the cost of flux computations, partly because it even avoids the use of a Riemann solver.

Let $v^n_{j,k}$ denote the fine grid cell averages of the solution, as used in section 2. In a multiresolution scheme the solution, discretized as cell averages, is subject to multiresolution analysis at each time step. In our semidiscrete formulation, this is done once for each evaluation of the RHS S. The RK2 method (2.18) is thus modified to read

$$v^{n+1} = \bar{\mathbf{E}}^{(1)}_{\mathbf{M}} \bar{\mathbf{M}}'^{-1} \mathbf{tr} \bar{\mathbf{M}}' \quad \bar{\mathbf{E}}^{(0)}_{\mathbf{M}} \bar{\mathbf{M}}'^{-1} \mathbf{tr} \bar{\mathbf{M}}' v^n, \tag{4.1}$$

where $\bar{\mathbf{E}}^{(\mathbf{0})}_{\mathbf{M}}$ and $\bar{\mathbf{E}}^{(\mathbf{1})}_{\mathbf{M}}$ are the two-stage, multiresolution equivalents of the evolution operator $\bar{\mathbf{E}}_{\mathbf{h}}$ in (2.2). Similar modifications could have been done for other time updates also.

We turn now to elaborate on each of the operations in (4.1), as they are carried out from right to left.

Step 1. Encoding $\bar{\mathbf{M}}'$: the encoding $\bar{\mathbf{M}}$ as defined by module (3.17) is modified as indicated by (3.25–3.26) so that the $\bar{e}$s are stored rather than the $\bar{d}$s. The resulting operation $\bar{\mathbf{M}}'$ then corresponds to (3.27). Given $\bar{u}^0 = v^n$, we have the following algorithm:

DO for $l = 1, 2, ..., L$

DO for $j = 0, 1, ..., M_l - 1; \quad k = 0, 1, ..., N_l - 1$

$$\bar{u}^l_{j,k} = \tfrac{1}{4}(\bar{u}^{l-1}_{2j,2k} + \bar{u}^{l-1}_{2j+1,2k} + \bar{u}^{l-1}_{2j,2k+1} + \bar{u}^{l-1}_{2j+1,2k+1})$$

ENDDO

DO for $j = 0, 1, ..., M_l - 1; \quad k = 0, 1, ..., N_l - 1$

$$\bar{d}^{l-1}_{2j,2k} = \bar{u}^{l-1}_{2j,2k} - \bar{I}(2j, 2k; \bar{u}^l)$$
$$\bar{d}^{l-1}_{2j+1,2k} = \bar{u}^{l-1}_{2j+1,2k} - \bar{I}(2j+1, 2k; \bar{u}^l)$$
$$\bar{d}^{l-1}_{2j,2k+1} = \bar{u}^{l-1}_{2j,2k+1} - \bar{I}(2j, 2k+1; \bar{u}^l)$$
$$\bar{d}^{l-1}_{2j+1,2k+1} = \bar{u}^{l-1}_{2j+1,2k+1} - \bar{I}(2j+1, 2k+1; \bar{u}^l)$$
$$\bar{e}^{l-1}_{2j,2k} = \tfrac{1}{4}(\bar{d}^{l-1}_{2j,2k} - \bar{d}^{l-1}_{2j+1,2k} - \bar{d}^{l-1}_{2j,2k+1} + \bar{d}^{l-1}_{2j+1,2k+1})$$
$$\bar{e}^{l-1}_{2j+1,2k} = \tfrac{1}{4}(-\bar{d}^{l-1}_{2j,2k} + \bar{d}^{l-1}_{2j+1,2k} - \bar{d}^{l-1}_{2j,2k+1} + \bar{d}^{l-1}_{2j+1,2k+1})$$
$$\bar{e}^{l-1}_{2j,2k+1} = \tfrac{1}{4}(-\bar{d}^{l-1}_{2j,2k} - \bar{d}^{l-1}_{2j+1,2k} + \bar{d}^{l-1}_{2j,2k+1} + \bar{d}^{l-1}_{2j+1,2k+1})$$
$$\bar{e}^{l-1}_{2j+1,2k+1} = -(\bar{e}^{l-1}_{2j,2k} + \bar{e}^{l-1}_{2j+1,2k} + \bar{e}^{l-1}_{2j,2k+1})$$

ENDDO

ENDDO (4.2)

Step 2. Truncation **tr**: the regularity analysis and the truncation can be combined into one module, and by using a Boolean flag array $\hat{i}^l_{j,k}$ we can implement $\bar{D}$ of (3.34b) as a set of on–off switches. The cells for which $\hat{i}^l_{j,k} = 1$ will be called "flagged," and they correspond to the cells that are not regular enough to warrant interpolation

from coarser levels. The value of $\hat{i}^l_{j,k}$ depends on four $\hat{e}^{l-1}$ values from one finer level and thus does not have to be stored on level $l-1$. Therefore, we shall have these flags only on levels 1 through L. We must also account for the creation of new irregularities in both the x- and y-directions, like shocks, for example, which will in turn lead to new, finer scales of resolution (take, for example, smooth initial data with Burgers's equation, where discontinuities are created in finite time). This condition is detected by relation (3.28), which is accomplished by the second **IF** statement of (4.3b). In addition, we also have to consider the propagation of discontinuities in all directions, with a finite speed that depends on the Courant–Friedrichs–Levy (CFL) number. Module (4.3b) assumes a CFL number less than unity, and thus it adds one cell of irregularity in each direction. The original idea comes from [8] for one dimension, and the issue, although trivial in the hyperbolic case, becomes the central problem when viscosity is added and waves do not propagate with finite speed. The algorithmic description of the truncation operation is the following:

(i) Initialize the $\hat{i}^l_{j,k}$:

$\epsilon_0 = \epsilon$
DO for $l = 1, 2, ..., L$
  DO for $j = 0, 1, ..., M_l - 1; \quad k = 0, 1, ..., N_l - 1$
    $\hat{i}^l_{jk} = 0$
  ENDDO
ENDDO (4.3a)

(ii) Truncate the $\bar{e}^l_{j,k}$ and set the $\hat{i}^l_{j,k}$:

DO for $l = 1, 2, ..., L$
  $\epsilon_l = \frac{1}{2}\epsilon_{l-1}$
  DO for $j = 0, 1, ..., M_l - 1; \quad k = 0, 1, ..., N_l - 1$
    IF $(|\bar{e}^{l-1}_{2j,2k}| \le \epsilon_l/c$ **AND** $|\bar{e}^{l-1}_{2j+1,2k}| \le \epsilon_l$ **AND** $|\bar{e}^{l-1}_{2j,2k+1}| \le \epsilon_l)$ **THEN**
      $\hat{e}^{l-1}_{j,k} = \hat{e}^{l-1}_{j+1,k} = \hat{e}^{l-1}_{j,k+1} = \hat{e}^{l-1}_{j+1,k+1} = 0$
    ELSE
      DO for $i = -1, 0, 1; \quad m = -1, 0, 1$
        $\hat{i}^l_{j-i,k-m} = 1$
      ENDDO
      $\hat{i}^l_{j,k} = 1$
      IF $(|\bar{e}^{l-1}_{2j,2k}| \ge 2^{\bar{r}+2}\epsilon_l/c$ **AND** $|\bar{e}^{l-1}_{2j+1,2k}| \ge 2^{\bar{r}+1}\epsilon_l$ **AND**
        $|\bar{e}^{l-1}_{2j,2k+1}| \ge 2^{\bar{r}+1}\epsilon_l$ **AND** $l > 1)$ **THEN**
          $\hat{i}^{l-1}_{2j,2k} = \hat{i}^{l-1}_{2j+1,2k} = \hat{i}^{l-1}_{2j,2k+1} = \hat{i}^{l-1}_{2j+1,2k+1} = 1$
      ENDIF

ENDIF

ENDDO

(4.3b) **ENDDO**

Step 3. Decoding $\bar{\mathbf{M}}'^{-1}$: the decoding $\bar{\mathbf{M}}^{-1}$ defined by module (3.18) is also modified to first obtain the $\hat{d}$s from the $\hat{e}$s, i.e., solve the system (3.25–3.26) for four of the $\hat{d}$s at a time (that system also holds if we substitute "$\hat{\ }$" for "$\bar{\ }$" on each d and e). Note that the $\hat{e}$s are truncated values, and the output from the decoding module will be $\hat{u}^0$, as indicated by (3.32). The input to the decoding routine is $\hat{u}^L = \bar{u}^L$. This yields the new decoding operation $\bar{\mathbf{M}}'^{-1}$:

DO for $l = L, L-1, ..., 1$

DO for $j = 0, 1, ..., M_l - 1; \quad k = 0, 1, ..., N_l - 1$

$$\hat{d}^{l-1}_{2j,2k} = \hat{e}^{l-1}_{2j,2k} - \hat{e}^{l-1}_{2j+1,2k} - \hat{e}^{l-1}_{2j,2k+1}$$
$$\hat{d}^{l-1}_{2j+1,2k} = -\hat{e}^{l-1}_{2j,2k} + \hat{e}^{l-1}_{2j+1,2k} - \hat{e}^{l-1}_{2j,2k+1}$$
$$\hat{d}^{l-1}_{2j,2k+1} = -\hat{e}^{l-1}_{2j,2k} - \hat{e}^{l-1}_{2j+1,2k} + \hat{e}^{l-1}_{2j,2k+1}$$
$$\hat{d}^{l-1}_{2j+1,2k+1} = -\hat{e}^{l-1}_{2j+1,2k+1}$$
$$\hat{u}^{l-1}_{2j,2k} = \hat{d}^{l-1}_{2j,2k} + \bar{I}(2j, 2k; \hat{u}^l)$$
$$\hat{u}^{l-1}_{2j+1,2k} = \hat{d}^{l-1}_{2j+1,2k} + \bar{I}(2j+1, 2k; \hat{u}^l)$$
$$\hat{u}^{l-1}_{2j,2k+1} = \hat{d}^{l-1}_{2j,2k+1} + \bar{I}(2j, 2k+1; \hat{u}^l)$$
$$\hat{u}^{l-1}_{2j+1,2k+1} = \hat{d}^{l-1}_{2j+1,2k+1} + \bar{I}(2j+1, 2k+1; \hat{u}^l)$$

ENDDO

(4.4) **ENDDO**

Step 4. Evolution $\bar{\mathbf{E}}^{(i)}_{\mathbf{M}}, i = 0, 1$: the evolution operations $\bar{\mathbf{E}}^{(i)}_{\mathbf{M}}$ correspond to the time updates of the solution for each stage of the Runge–Kutta method. They all use the same flux computation and RHS computation modules, which are now described in detail.

Step 4a. Flux computation:

(i) The numerical flux computations start on the coarsest level L, where the point values of the fluxes are evaluated directly using (2.5b) and then assembled using (2.9). On the coarsest level, of course, each cell face will be (much) larger than on the finest grid level. Since we require resolution on the original (finest) grid, the fluxes on the coarse grid must be computed by separately evaluating the individual fine grid fluxes for each component of the large cell face. The fine grid flux values are evaluated using (2.5b) and fine grid reconstructed values of u. Each flux will be a function of all the cell averages around the current cell which participate in the selection procedure outlined in section 2.3. This will obviously depend on the spatial accuracy of the reconstruction and the particular selection procedure. The entire $\hat{u} = \hat{u}^0$ array will certainly include all the cell averages, and, as usual, we use it as a symbolic argument. On the other hand, for the functions $\bar{f}$ and $\bar{g}$ below, we use the aforementioned flux composition from the fine grid, when the indicated level is not 0. (For all the algorithms below we use integral indices for f and g; although, in the formulation of section 2, they

correspond to the appropriate "half" index of the particular cell face.) We have the following double **DO** loop on level L:

DO for $j = 0, 1, ..., M_L - 1; \quad k = 0, 1, ..., N_L - 1$

$f^L_{j,k} = \bar{f}(j, k, L; \hat{u}^0)$

$g^L_{j,k} = \bar{g}(j, k, L; \hat{u}^0)$

(4.5a) **ENDDO**

(ii) Next, we hierarchically build up the fluxes on the finer levels, using the previously computed values on level L. Recall that the set D, whose cardinality is symbolically given by (3.35), comprises all the cell faces at all levels where fluxes need to be computed directly. So on each level thereafter a decision is made at each cell face whether to compute the flux there or not, based on values of the array $\hat{i}$ defined by (4.3a–b). We have

DO for $l = L, L-1, \ldots, 1$

DO for $j = 0, 1, ..., M_l - 1; \quad k = 0, 1, ..., N_l - 1$

IF $\hat{i}^l_{j,k} = 1$ **THEN**

DO for $i = -1, 0, 1; m = -1, 0, 1$

$f^{l-1}_{2j+i,2k+m} = \bar{f}(2j + i, 2k + m, l - 1; \hat{u}^0)$

$g^{l-1}_{2j+i,2k+m} = \bar{g}(2j + i, 2k + m, l - 1; \hat{u}^0)$

ENDDO

ENDIF

ENDDO

(4.5b) **ENDDO**

In both (4.5a) and (4.5b) we use the following definition of the flux functions $\bar{f}$ and $\bar{g}$ for $l > 0$:

$$\bar{f}(j, k, l; \hat{u}^0) = \sum_{m=2^l k}^{2^l(k+1)-1} \bar{f}(j, m, 0; \hat{u}^0), \tag{4.5c}$$

$$\bar{g}(j, k, l; \hat{u}^0) = \sum_{i=2^l j}^{2^l(j+1)-1} \bar{g}(i, k, 0; \hat{u}^0). \tag{4.5d}$$

Since for the regularity analysis we visit each cell once, for those cells that are next to other flagged cells, the fluxes for some of the cell faces may have been computed already during a previous cell's turn. Also, some fine grid fluxes may have been computed during a sweep of the previous level of resolution. If such flagged fluxes exist, we skip the flux computation, so we never have to compute a fine grid flux twice. Note that fluxes are never interpolated; they are either computed exactly or skipped entirely.

Step 4b. RHS computation:

(i) The RHS S, as defined by (2.10), is also first computed on the coarsest level L, and it is using the fluxes computed by module (4.5a). In pseudo code form we have

DO for $j = 0, 1, ..., M_L - 1; \quad k = 0, 1, ..., N_L - 1$

$S^L_{j,k} = S(j, k, L; \hat{u}^0)$

(4.6a) **ENDDO**

(ii) The numerical divergences for each level finer than L are recursively computed by a decoding procedure similar to (3.18). We use the same interpolation $\bar{I}$ described by (3.14–3.15) but now for S instead of v. It is at this stage where the decision is made whether to compute the RHS directly using (3.10) or to interpolate from coarser levels in the sense of cell averages. See Appendix B for a proof that the RHS does indeed fit a cell average multiresolution representation. The algorithm reads as follows:

DO for $l = L, L-1, ..., 1$

DO for $j = 0, 1, ..., M_l - 1; \quad k = 0, 1, ..., N_l - 1$

IF $\hat{i}^l_{j,k} = 1$ **THEN**

DO for $i = 0, 1; \quad m = 0, 1$

$S^{l-1}_{2j+i,2k+m} = S(2j + i, 2k + m, l - 1; \hat{u}^0)$

ENDDO

ELSE

DO for $i = 0, 1; \quad m = 0, 1$

$S^{l-1}_{2j+i,2k+m} = \bar{I}(2j + i, 2k + m; S^l)$

ENDDO

ENDIF

ENDDO

(4.6b) **ENDDO**

where we used the following definition of the numerical divergence function:

$$S(j,k,l;u^0) = -\frac{1}{h_{x,l}}\left(\bar{f}(j,k,l;\hat{u}^0) - \bar{f}(j-1,k,l;\hat{u}^0)\right) - \frac{1}{h_{y,l}}\left(\bar{g}(j,k,l;\hat{u}^0) - \bar{g}(j,k-1,l;\hat{u}^0)\right) \tag{4.6c}$$

Step 4c. Time update of the cell averages: once the RHS $S = S^0$ is obtained on the finest grid, the time update for the cell average array $v = \hat{u}^0$ takes place. Steps 1–4 are then repeated for each additional stage of the time stepping procedure.

The above multiresolution scheme can be adapted to any other semidiscrete scheme, as long as the scheme is based on a finite volume formulation which computes cell averages. Note that time updates are done on the finest level, and wherever flux computations are needed, they are done using the cell average array on the finest level also.

While the basic principles behind the 2-D multiresolution schemes are the same as for their 1-D counterparts, there are significant differences in implementation. The

main difference is the use of multiresolution interpolation of the numerical divergence, or RHS, in smooth regions, rather than interpolation of the fluxes. As we have remarked earlier, the symmetry offered by the former is necessary to preserve the oscillation-free nature of the solution (except for small oscillations on the level of the truncation error). Intuitively it is also clear that because of the degenerate cases used in (3.7), the always fully 2-D stencil of cell averages cannot, in general, be used to measure the regularity of a 1-D array. We have experimented with other formulations where, in the second-order case, for example, the flux was computed using the trapezoidal rule instead of the midpoint rule, and thus we could, one out of four times, copy flux values from coarse grid to fine. Such variations, however, reduce the generality of the method presented and only offer marginal increase in efficiency.

It should also be pointed out that the current treatment using the divergence instead of flux values does not contradict the original 1-D method of [8]. In one dimension the point value multiresolution interpolation of the fluxes is, in fact, equivalent to the cell average multiresolution interpolation of the RHS. This explains why the interpolation of fluxes was successful in one dimension.

The regularity analysis presents other implementation details which make two dimensions and one dimension quite different. As can be seen from algorithm (4.3b), for instance, we took a very conservative approach in accounting for creation of shocks. We "padded" the set $\bar{D}$ with nonzero $\hat{i}$ values one multiresolution "layer" away from the current irregularity, a procedure that may not always be necessary. We also increased the multiresolution stencil by adding one layer of cells to the current location of the irregularity. The need for the latter in one dimension is explained in [8] by the observation that discontinuities (or signals, in general) can move only one cell away from the current one, during one time step, if a CFL number of 1 or less is used; the same argument holds in two dimensions as well. Slight improvements of the efficiency ratios could be possible by reducing the padding, but in the absence of a more rigorous theory, we need extensive numerical experimentation to justify such steps.

5. Numerical experiments. We now show numerical results obtained using the scheme presented in section 4. As stated earlier, the computational domain was $[-1,1]\times[-1,1]$, and periodic boundary conditions were used in both x and y. The 2-D linear wave equation and the 2-D Burgers equation (without viscosity) were solved with fluxes defined by (2.6a) and (2.7a), respectively. We ran the code using the following initial conditions (IC):

$$u_0(x,y) = \sin \pi x \sin \pi y \tag{5.1}$$

and

$$u_0(x,y) = \begin{cases} 1, & \text{if } |x+y+1| \le \frac{1}{2} \text{ and } |y-x| \le \frac{1}{2}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.2}$$

The tables use the following standard definitions for errors:

$$e_\infty = \max_{\substack{0\le j\le M_0-1 \\ 0\le k\le N_0-1}} |v_{j,k}^{0,n} - w_{j,k}^n|, \tag{5.3a}$$

$$e_p = \left[\frac{1}{M_0 N_0} \sum_{j=0}^{M_0-1} \sum_{k=0}^{N_0-1} |v_{j,k}^{0,n} - w_{j,k}^n|^p \right]^{\frac{1}{p}}, \quad p = 1, 2, \tag{5.3b}$$

where $w^n_{j,k}$ is the solution obtained using direct flux and RHS computations everywhere, since we mainly want to measure the deviation caused by skipping numerous ENO flux evaluations. For the linear case, where the exact solution is also readily available, we also define

$$E_1 = \frac{1}{M_0 N_0} \sum_{j=0}^{M_0-1} \sum_{k=0}^{N_0-1} |v^{0,n}_{j,k} - u(x_{j,k}, y_{j,k}, t)|, \tag{5.3c}$$

with $u(x_{j,k}, y_{j,k}, t), t = n\tau$, being the exact solution. The most relevant indicator of how well the multiresolution scheme is performing seems to be the efficiency variable, denoted by μ and defined by (3.36). μ is, of course, related to the data compression ω that can be achieved by truncating the small multiresolution coefficients that fall below tolerance:

$$\omega = \frac{M_0 N_0}{\frac{M_0 N_0}{4^L} + |\bar{D}|}, \tag{5.4}$$

where $\bar{D}$ is the set of nonzero multiresolution coefficients as defined in (3.34). In the tables we show both of these indicators; note that for one dimension they were equal.

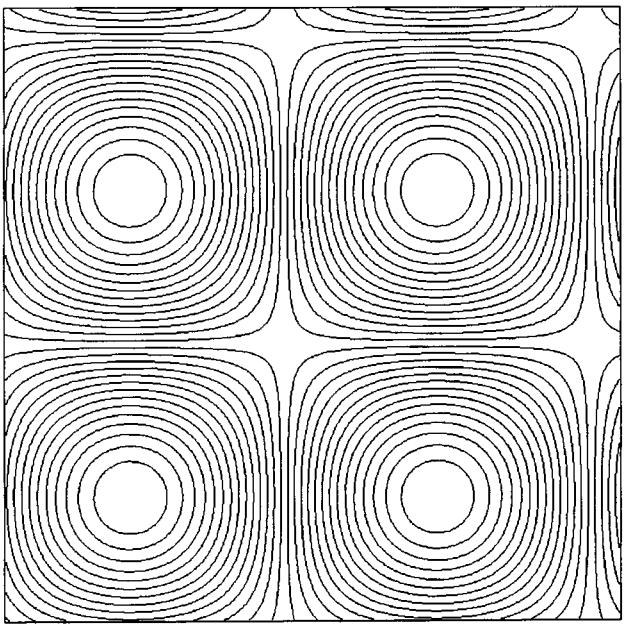
Throughout the computations, we used a CFL number of $\frac{1}{2}$, and a multiresolution tolerance parameter $\epsilon = 10^{-3}$ which was used along with (3.34) to determine the truncation thresholds at each level. The number of grid cells used in each direction was $M_0 = 256, N_0 = 256$, and the total number of multiresolution levels was 6 ($L = 5$, $l = 0$ is the finest grid). For all computations with $r \leq 3$, the multiresolution interpolation in the sense of cell averages was third order; i.e., the parameter s used in (3.15a–c) was 2.

In the next two sections, we shall discuss two main cases corresponding to $r = 1, 2$ used in the reconstruction (2.11a). As in all the multiresolution diagrams to follow, we use the visualization method first shown in Fig. 1b, where darker shades correspond to finer levels, lighter ones to coarser levels. Wherever squares are "blank," those cells are not flagged and thus their divergence is predicted from coarser grid levels. For the solution, we used the standard contour line plotter.

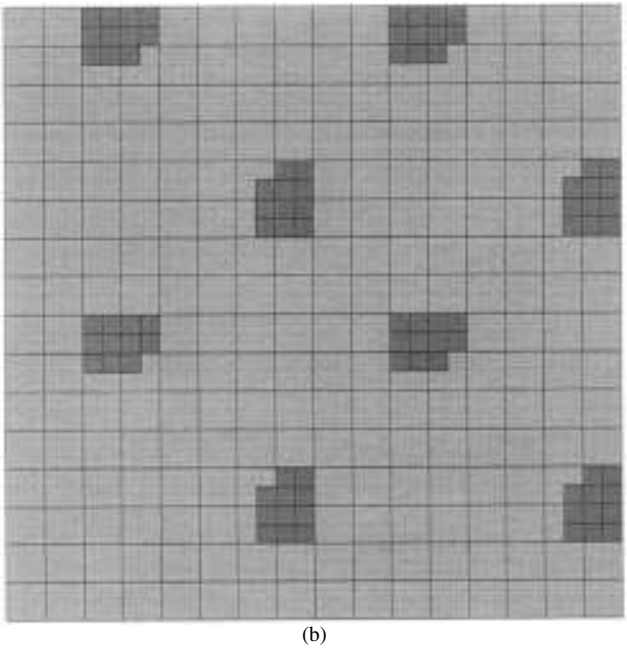
5.1. First-order reconstruction. The $r = 1$ reconstruction corresponds to a piecewise constant interpolation in each cell (see (2.12)). It has been shown in [5] that in two dimensions a total variation diminishing scheme is at most first-order accurate. This reconstruction guarantees an oscillation-free solution and ultimately convergence.

A. *The linear case.*

Case (i): *Smooth initial data.* For the smooth initial data (5.1) we ran up to 1000 time steps, at the end of which we obtained the contour plot shown in Fig. 3a. First-order schemes are extremely dissipative, which is exhibited throughout all the computations with $r = 1$. The range of the solution shrank from $[-1, 1]$ to $[-0.738, 0.738]$, but in comparison with the scheme without multiresolution, the l_1 error (Table 1) is an order of magnitude less than when compared with the exact solution. This confirms that whatever quality the solution would have had without multiresolution is preserved. The smoothing effect of first-order schemes leads to a steady increase in the efficiency μ, up to more than 7 at $n = 1000$. Note that the multiresolution diagram of Fig. 3b shows significant coefficients only on the three coarsest levels, correlating to the high efficiency achieved.



(a)



(b)

FIG. 3. *Solution at* $n = 1000$, $r = 1$, 2-*D wave equation, IC* (5.1).

Case (ii): *Discontinuous initial data.* Next, IC (5.2) was used, leading to Figs. 4a and 4b after 1000 time steps. Again the solution became very smooth, but no deterioration was due to multiresolution analysis; all the errors e are well below the E_1 error. As for Fig. 4b, there are only two symbols at the finest level, and the outline of where the rotated step function should be can easily be distinguished on the second

TABLE 1

*Numerical solution of 2-D linear wave equation, $r = 1$. The * at the end of a line means that there is a figure for that particular case.*

IC	n	μ	ω	e_∞	e_1	e_2	E_1
(5.1)	10	4.91	16.65	1.28×10^{-4}	1.72×10^{-5}	3.13×10^{-5}	1.22×10^{-3}
	100	4.90	16.65	1.45×10^{-3}	1.88×10^{-4}	2.64×10^{-4}	1.33×10^{-2}
	200	5.09	17.66	1.71×10^{-3}	2.93×10^{-4}	4.03×10^{-4}	2.50×10^{-2}
	500	5.49	20.48	2.77×10^{-3}	5.86×10^{-4}	7.59×10^{-4}	5.79×10^{-2}
	1000	7.19	38.64	3.14×10^{-3}	1.22×10^{-3}	1.40×10^{-3}	1.06×10^{-1}*
(5.2)	10	4.22	5.50	2.40×10^{-4}	5.76×10^{-7}	4.45×10^{-6}	9.52×10^{-3}
	100	3.15	5.60	1.18×10^{-2}	6.27×10^{-4}	1.50×10^{-3}	3.05×10^{-2}
	200	2.90	14.28	1.78×10^{-2}	2.92×10^{-3}	5.20×10^{-3}	4.37×10^{-2}
	500	2.71	15.26	1.78×10^{-2}	3.31×10^{-3}	5.13×10^{-3}	6.71×10^{-2}
	1000	2.81	17.45	1.72×10^{-2}	3.24×10^{-3}	4.90×10^{-3}	9.13×10^{-2}*

finest level. As shown by Table 1, after an initial decrease in the transient, μ shows a monotone increase, as the smoothing progresses.

B. *The nonlinear case.*

Case (i): *Smooth initial data.* When Burgers's equation was run with IC (5.1) we found tremendous decrease in efficiency (Table 2), which is due simply to the nature of the physical solution which develops many discontinuities. Note, however, the increase in efficiency between steps 400 and 700, where the smoothing effect started to outweigh the effect of the development of the discontinuities (Fig. 5a). The multiresolution diagram shows coefficients on the finest level only near discontinuities which run tangent to the propagation of the wave.

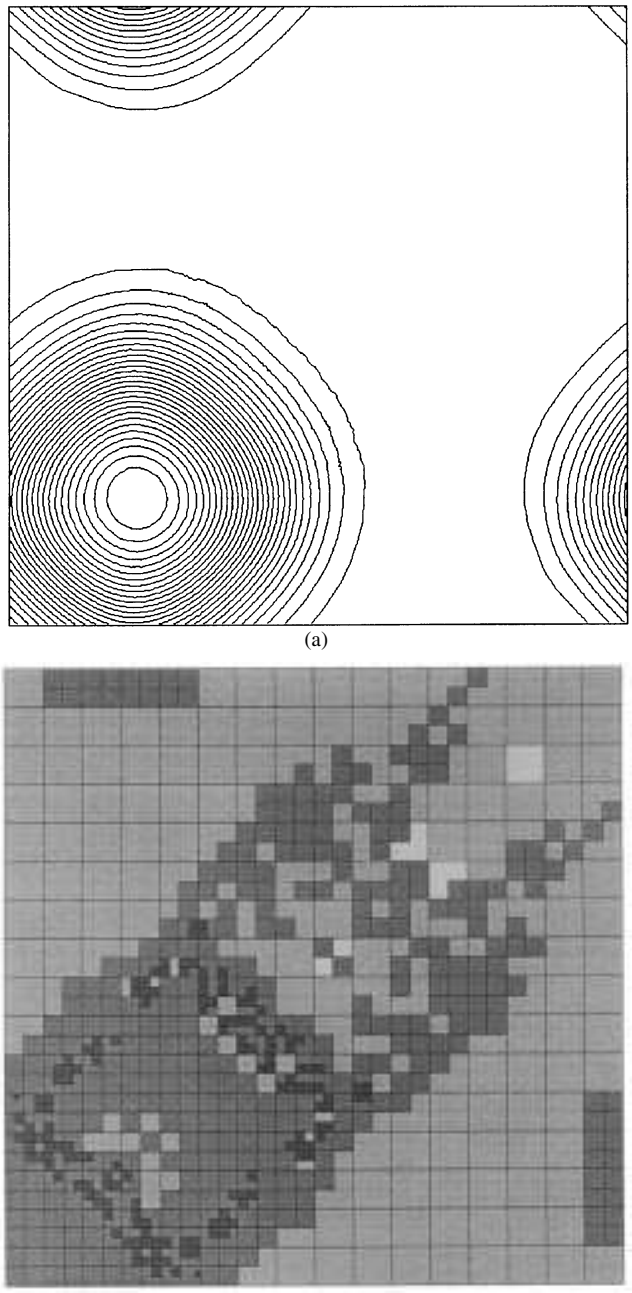
Case (ii): *Discontinuous initial data.* For IC (5.2) the efficiency also decreased in time, approximately in a linear fashion, due to the smearing of the initial discontinuities. No new discontinuities were created here. Figure 6b shows a clear outline of fine grid multiresolution coefficients where the discontinuities lie (including the location of the stationary rarefaction fan).

5.2. Second-order reconstruction. In this subsection we show computational results for the second-order case, where the strategy outlined in section 2.3 was used to find an acceptable second-order ENO reconstruction. We expect much improved accuracy compared with the first-order case, which may, in fact, mean a decrease in efficiency.

A. *The linear case.*

Case (i): *Smooth initial data.* The solution in Fig. 7a shows a much more accurate solution than in the first-order case. The corresponding multiresolution diagram (Fig. 7b) again displays significant coefficients on the three coarsest levels, fairly evenly distributed throughout the domain, leading to efficiencies of more than 5 (Table 3). Interestingly, the e-errors did not improve, or even increased from those of the first-order case, but the E_1 error shows about an order of magnitude improvement, which is expected. This seems to indicate no correlation between the errors committed by using multiresolution interpolation and the order of reconstruction, a fact which will be supported by the rest of the computations.

Case (ii): *Discontinuous initial data.* The efficiency decrease in time shown in the previous case is replicated for IC (5.2) as well (Table 3). Figure 8a shows a solution that even qualitatively appears to be much more accurate than that of Fig. 4a, leading to efficiencies just below 2 at $n = 1000$. Figure 8b also shows some multiresolution



(a)

(b)

FIG. 4. *Solution at* $n = 1000$, $r = 1$, 2-*D wave equation, IC* (5.2).

coefficients away from the discontinuities, on the second-finest level. This leads us to believe that some small oscillations created by the (not TVD) second-order ENO reconstruction is propagated in the direction of the wave velocity. Finally, we note the rather high e_∞ errors which may be due to the dispersive nature of second-order schemes. The e_1 and e_2 errors stay small, and the E_1 error shows a significant reduction from the first-order case.

TABLE 2

*Numerical solution of 2-D inviscid Burgers's equation, $r = 1$. The * at the end of a line means that there is a figure for that particular case.*

IC	n	μ	ω	e_∞	e_1	e_2
(5.1)	10	5.79	20.68	8.65×10^{-4}	7.18×10^{-5}	1.28×10^{-4}
	100	3.52	5.16	3.21×10^{-3}	2.29×10^{-4}	3.93×10^{-4}
	200	2.87	3.62	3.40×10^{-3}	2.25×10^{-4}	4.17×10^{-4}
	400	1.89	2.00	4.34×10^{-3}	2.81×10^{-4}	4.40×10^{-4}
	700	1.98	2.49	6.13×10^{-3}	1.12×10^{-3}	1.53×10^{-3}*
(5.2)	50	4.41	6.57	2.33×10^{-3}	4.31×10^{-5}	1.83×10^{-4}
	150	4.08	5.90	3.87×10^{-3}	9.46×10^{-5}	3.07×10^{-4}
	250	3.68	5.29	8.90×10^{-3}	1.15×10^{-4}	3.82×10^{-4}
	350	3.39	4.77	9.64×10^{-3}	1.48×10^{-4}	4.38×10^{-4}
	400	3.31	4.63	9.25×10^{-3}	1.68×10^{-4}	4.55×10^{-4}*

B. *The nonlinear case.*

Case (i): *Smooth initial data.* For IC (5.1) we again noticed a seemingly exponential decrease in efficiency due to the physics, and we use this case as a pathological case, or "worst case scenario" (Table 4). Figure 9a now shows many, quite sharp discontinuities, which are also clearly shown on Fig. 9b by multiresolution coefficients on the finest level.

Case (ii): *Discontinuous initial data.* This case is probably the most relevant to a nonlinear hyperbolic computation of the cases discussed so far. The discontinuity is well preserved (Fig. 10a), and the efficiencies and errors are similar to the first-order case (Table 4). The multiresolution diagram of Fig. 10b again reflects the regularity of the solution well. As Fig. 10a is compared with Fig. 6a, we note the expected improvement in the accuracy of the solution, especially noticeable on the "less rounded" shock front of the second-order case.

6. Summary. The semidiscrete formulation of 2-D multiresolution schemes presented in this paper showed the advantage of using multiresolution analysis in order to selectively use ENO reconstructions only where that is truly needed. The number of flux computations were thus reduced by factors ranging from about 2 to 6. We achieved this by not affecting the quality of the solution. The efficiencies, on the average, were comparable with the corresponding 1-D cases. While more theory is needed both for ENO schemes and multiresolution analysis, the computational results are very promising.

Appendix A. In this appendix we show (lengthy) derivations for expressions (3.21), (3.22), and (3.20b–c), in that order. To avoid multiple subscripts and superscripts, we first introduce a simpler notation, as follows, also illustrated in Fig. A.1. For a given level l and indices (j, k) on l, where $0 \le j \le M_l - 1$, $0 \le k \le N_l - 1$, let

$$u_0 = \bar{u}^l_{j,k}, \tag{A.1a}$$

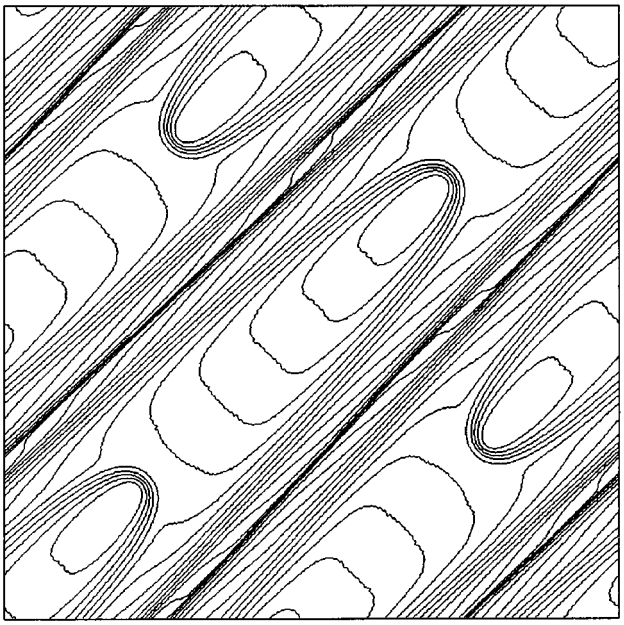
$$u_1 = \bar{u}^{l-1}_{2j,2k}, \tag{A.1b}$$

$$u_2 = \bar{u}^{l-1}_{2j+1,2k}, \tag{A.1c}$$

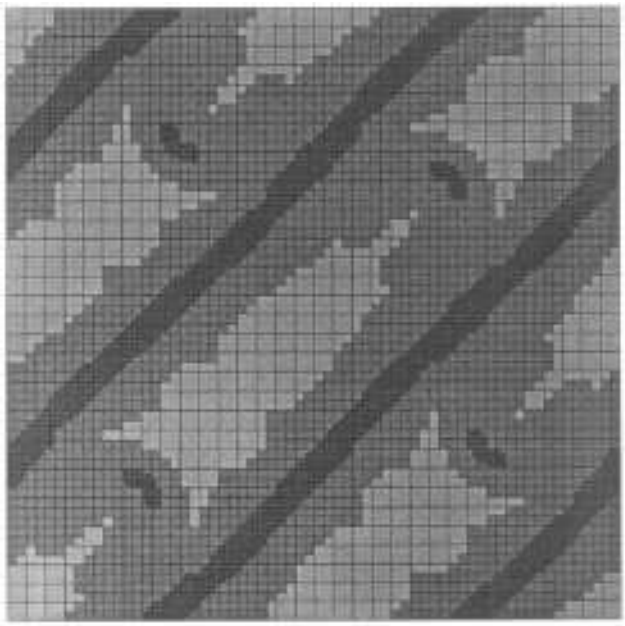
$$u_3 = \bar{u}^{l-1}_{2j,2k+1}, \tag{A.1d}$$

$$u_4 = \bar{u}^{l-1}_{2j+1,2k+1}, \tag{A.1e}$$

$$Q_x = Q^s_x(j, k; \bar{u}^l), \tag{A.1f}$$



(a)



(b)

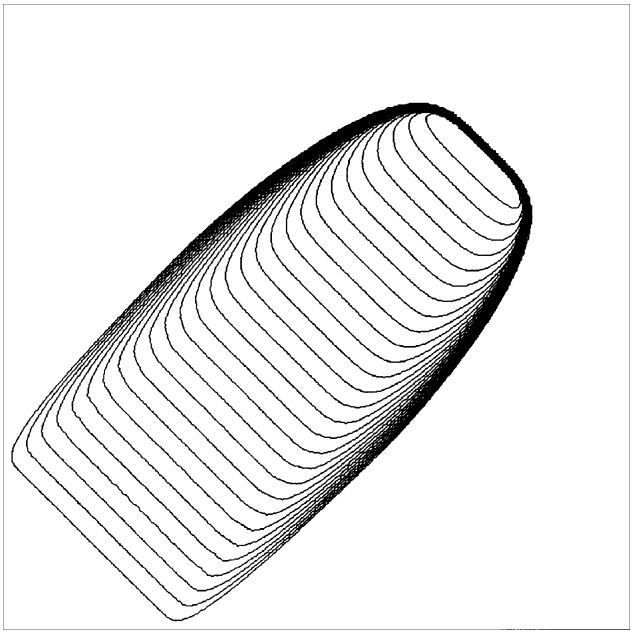
FIG. 5. *Solution at* $n = 700$, $r = 14$, *Burgers's equation, IC* (5.1).

$$Q_y = Q_x^s(j,k;\bar{u}^l), \tag{A.1g}$$

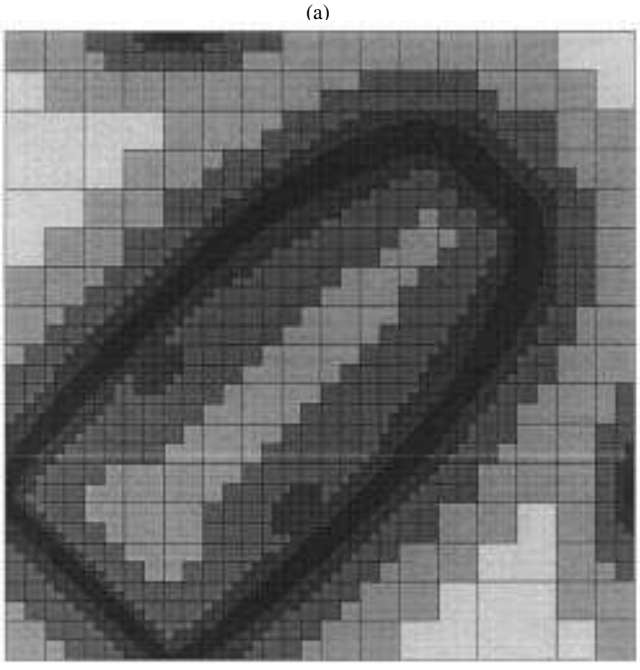
$$Q_{xy} = Q_{xy}^s(j,k;\bar{u}^l), \tag{A.1h}$$

$$d_1 = \bar{d}^{l-1}_{2j,2k}, \tag{A.1i}$$

$$d_2 = \bar{d}^{l-1}_{2j+1,2k}, \tag{A.1j}$$



(a)



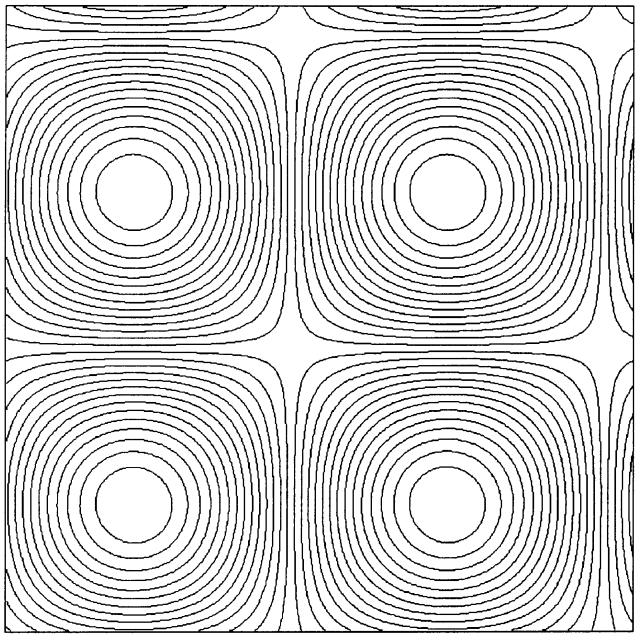
(b)

FIG. 6. *Solution at* $n = 400$, $r = 1$, *Burgers's equation, IC* (5.2).

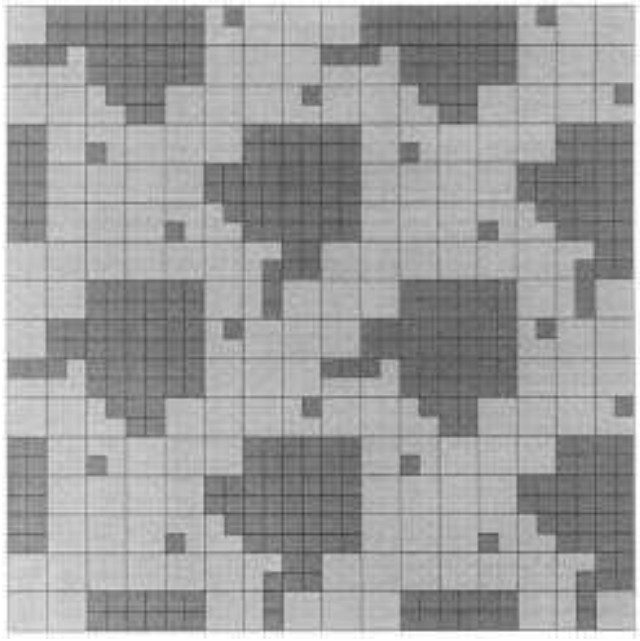
(A.1k) $$d_3 = \vec{d}^{\,l-1}_{2j,2k+1},$$

(A.1l) $$d_4 = \vec{d}^{\,l-1}_{2j+1,2k+1}.$$

Using these definitions and (3.14–3.15) with (3.17), we get



(a)



(b)

FIG. 7. *Solution at* $n = 1000$, $r = 2$, *ENO*.2, 2-*D wave equation, IC* (5.1).

$$d_1 = u_1 - u_0 + Q_x + Q_y - Q_{xy}, \tag{A.2a}$$

$$d_2 = u_2 - u_0 - Q_x + Q_y + Q_{xy}, \tag{A.2b}$$

$$d_3 = u_3 - u_0 + Q_x - Q_y + Q_{xy}, \tag{A.2c}$$

$$d_4 = u_4 - u_0 - Q_x - Q_y - Q_{xy}; \tag{A.2d}$$

TABLE 3
*Numerical solution of 2-D linear wave equation, ENO.2, $r = 2$. The * at the end of a line means that there is a figure for that particular case.*

IC	n	μ	ω	e_∞	e_1	e_2	E_1
(5.1)	10	5.63	21.33	1.41×10^{-3}	2.87×10^{-5}	6.32×10^{-5}	2.96×10^{-4}
	100	5.60	21.56	6.34×10^{-3}	2.56×10^{-4}	4.43×10^{-4}	2.12×10^{-4}
	200	5.52	20.69	9.68×10^{-3}	3.47×10^{-4}	6.62×10^{-4}	3.20×10^{-4}
	500	5.54	21.33	1.33×10^{-2}	6.63×10^{-4}	1.20×10^{-3}	6.46×10^{-4}
	1000	5.75	22.26	1.64×10^{-2}	1.16×10^{-3}	1.91×10^{-3}	1.21×10^{-3}*
(5.2)	10	4.31	5.80	2.64×10^{-4}	1.09×10^{-6}	9.10×10^{-6}	6.97×10^{-3}
	100	3.27	4.84	4.95×10^{-2}	6.00×10^{-4}	1.83×10^{-3}	1.34×10^{-2}
	200	2.56	4.63	4.03×10^{-2}	1.47×10^{-3}	2.80×10^{-3}	1.59×10^{-2}
	500	1.97	5.23	3.24×10^{-2}	2.26×10^{-3}	3.43×10^{-3}	1.95×10^{-2}
	1000	1.92	4.22	3.08×10^{-2}	2.07×10^{-3}	3.13×10^{-3}	2.23×10^{-2}*

and (3.25a–d) yield

$$e_1 = \frac{1}{4}(d_1 - d_2 - d_3 + d_4), \tag{A.3a}$$

$$e_2 = \frac{1}{4}(-d_1 + d_2 - d_3 + d_4), \tag{A.3b}$$

$$e_3 = \frac{1}{4}(-d_1 - d_2 + d_3 + d_4), \tag{A.3c}$$

$$e_4 = -(e_1 + e_2 + e_3). \tag{A.3d}$$

Note that system (A.3) above also corresponds to vector equation (4.10). We will also need the following (see Fig. A.1):

$$u_{13} = \frac{1}{2}(u_1 + u_3), \tag{A.4a}$$

$$u_{12} = \frac{1}{2}(u_1 + u_2), \tag{A.4b}$$

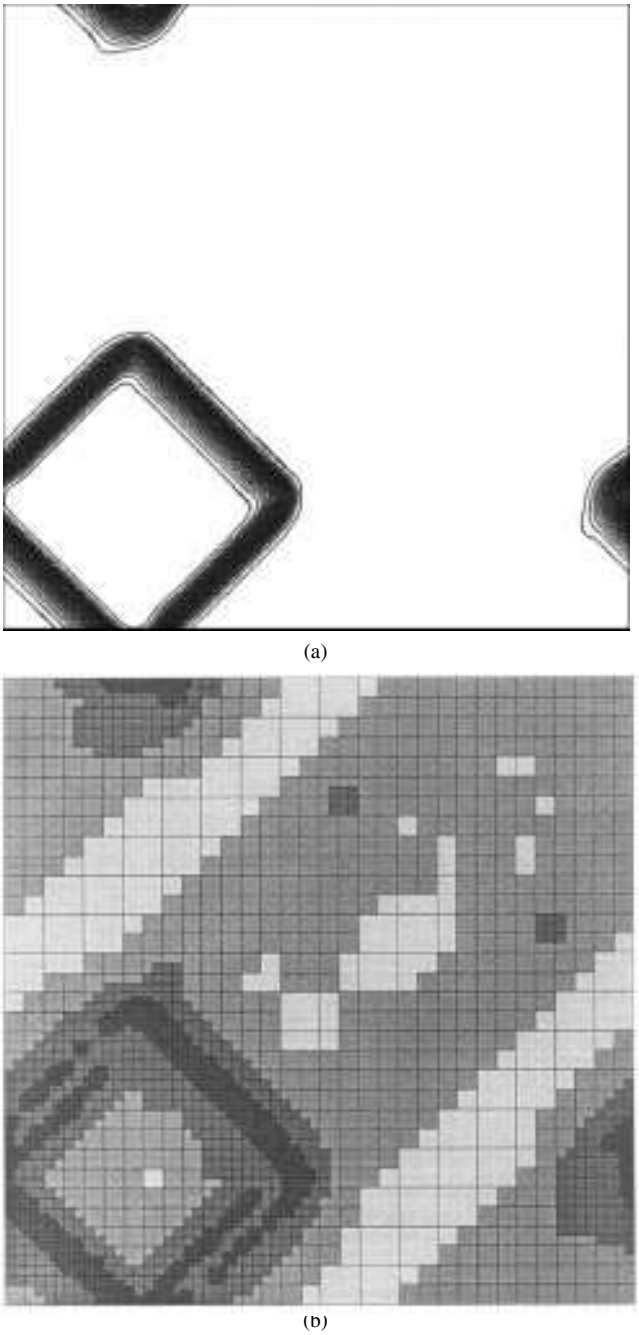
$$u_{24} = \frac{1}{2}(u_2 + u_4), \tag{A.4c}$$

$$u_{34} = \frac{1}{2}(u_3 + u_4). \tag{A.4d}$$

The cell which contains the cell average u_i, $i = 1, 2, 3, 4$ will also be referred to as "cell i."

(i) To derive (3.21) we use results from 1-D multiresolution analysis in [9] and interpolation results for the x-direction, based on the assumptions of section 3.3. We do this by restricting our attention, for the moment, to the kth row and note that $u_0 + Q_x$ is nothing else but the interpolated value of the cell average of the solution u over the cell which is the union of cells 1 and 3. Similarly, $u_0 - Q_x$ is the approximated value of the cell average over the union of cells 2 and 4, based on interpolating along the j-direction. That is,

$$u_{13} - u_0 - Q_x = \begin{cases} c_x h_{x,l}^{\bar{r}} \frac{\partial^{\bar{r}} u}{\partial x^{\bar{r}}}(\xi_1), & \text{if } p > \bar{r}, \\ c_x h_{x,l}^{p} \left[\frac{\partial^p u}{\partial x^p}\right](\xi_1), & \text{if } p \le \bar{r}, \end{cases} \tag{A.5a}$$



(a)

(b)

FIG. 8. *Solution at* $n = 1000$, $r = 2$, *ENO*.2, 2-*D wave equation, IC* (5.2).

$$u_{24} - u_0 + Q_x = \begin{cases} c_x h_{x,l}^{\bar{r}} \frac{\partial^{\bar{r}} u}{\partial x^{\bar{r}}}(\xi_2), & \text{if } p > \bar{r}, \\ c_x h_{x,l}^{p} \left[\frac{\partial^{p} u}{\partial x^{p}}\right](\xi_2), & \text{if } p \le \bar{r}, \end{cases} \tag{A.5b}$$

where c_x is the Taylor series coefficient which depends on the order and $\xi_i = (x_i, y_i)$, $i = 1, 2$ are points in a cell that belong to the stencil of Q_x. Substituting equations

TABLE 4

*Numerical solution of 2-D inviscid Burgers's equation, ENO.2, $r = 2$. The * at the end of a line means that there is a figure for that particular case.*

IC	n	μ	ω	e_∞	e_1	e_2
(5.1)	10	6.17	26.95	1.13×10^{-3}	6.35×10^{-5}	1.01×10^{-4}
	100	3.67	5.60	1.00×10^{-2}	2.50×10^{-4}	4.93×10^{-4}
	200	2.86	4.04	1.48×10^{-2}	3.27×10^{-4}	6.17×10^{-4}
	400	1.85	2.35	1.51×10^{-2}	7.70×10^{-4}	1.38×10^{-3}
	700	1.59	2.02	5.49×10^{-2}	3.16×10^{-3}	5.04×10^{-3}*
(5.2)	50	4.11	5.53	2.15×10^{-2}	8.93×10^{-5}	6.75×10^{-4}
	150	3.76	5.19	2.21×10^{-2}	1.07×10^{-4}	5.25×10^{-4}
	250	3.39	4.58	1.76×10^{-2}	1.38×10^{-4}	5.29×10^{-4}
	350	3.11	4.13	1.54×10^{-2}	1.70×10^{-4}	5.48×10^{-4}
	400	3.00	3.97	1.78×10^{-2}	1.94×10^{-4}	5.82×10^{-4}*

(A.2a–d) into (A.3b) and then using (A.4a) and (A.4c), we get

$$\begin{aligned} e_2 &= Q_x + \frac{1}{4}(-u_1 + u_2 - u_3 + u_4) \\ &= \frac{1}{2}(u_{24} - u_{13}). \end{aligned} \tag{A.6}$$

Now substitute (A.5a–b) into (A.6) and obtain

$$e_2 = \begin{cases} c_x h_{x,l}^{\bar{r}} \frac{\partial^{\bar{r}} u}{\partial x^{\bar{r}}}(\xi), & \text{if } p > \bar{r}, \\ c_x h_{x,l}^{p} \left[\frac{\partial^p u}{\partial x^p}\right](\xi), & \text{if } p \le \bar{r}, \end{cases} \tag{A.7}$$

where again ξ is some point in a cell of the j-stencil. This proves (3.21). □

(ii) To prove (3.22), we proceed similarly as in the previous case. Along the jth row we have

$$u_{12} - u_0 - Q_y = \begin{cases} c_y h_{y,l}^{\bar{r}} \frac{\partial^{\bar{r}} u}{\partial y^{\bar{r}}}(\xi_1), & \text{if } q > \bar{r}, \\ c_y h_{y,l}^{q} \left[\frac{\partial^q u}{\partial y^q}\right](\xi_1), & \text{if } q \le \bar{r}, \end{cases} \tag{A.8a}$$

$$u_{34} - u_0 + Q_y = \begin{cases} c_y h_{y,l}^{\bar{r}} \frac{\partial^{\bar{r}} u}{\partial y^{\bar{r}}}(\xi_2), & \text{if } q > \bar{r}, \\ c_y h_{y,l}^{q} \left[\frac{\partial^q u}{\partial y^q}\right](\xi_2), & \text{if } q \le \bar{r}. \end{cases} \tag{A.8b}$$

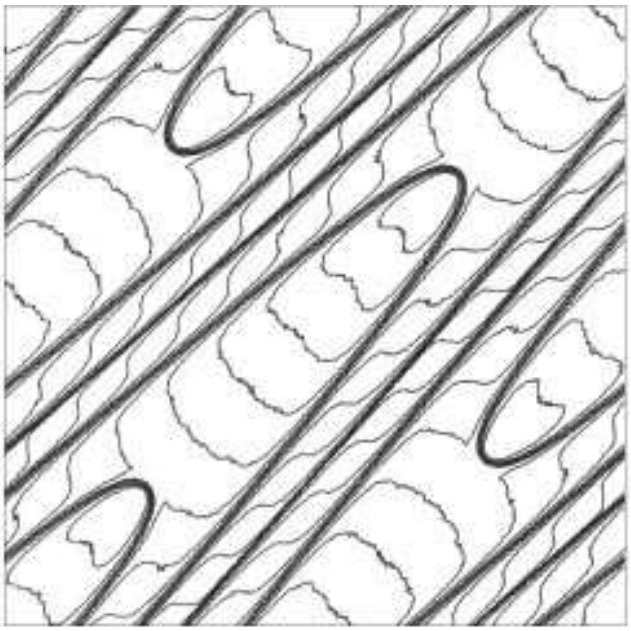
Here again c_y is the Taylor series coefficient based on the order, and $\xi_i = (x_i, y_i), i = 1, 2$ are points in a cell that now belong to the stencil of Q_y. From (A.2a–d) and (A.3c), and then from (A.4b) and (A.4d), we get

$$\begin{aligned} e_3 &= Q_y + \frac{1}{4}(-u_1 - u_2 + u_3 + u_4) \\ &= \frac{1}{2}(u_{34} - u_{12}). \end{aligned} \tag{A.9}$$

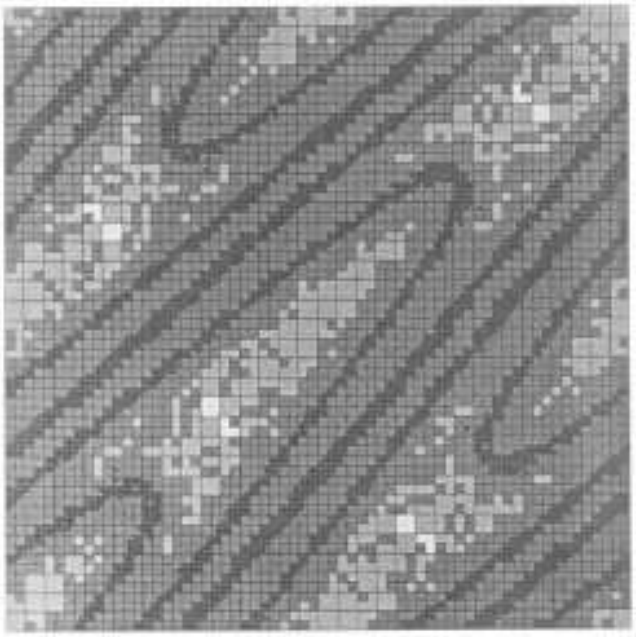
Using (A.8a–b) yields

$$e_3 = \begin{cases} c_y h_{y,l}^{\bar{r}} \frac{\partial^{\bar{r}} u}{\partial y^{\bar{r}}}(\xi), & \text{if } q > \bar{r}, \\ c_y h_{y,l}^{q} \left[\frac{\partial^q u}{\partial y^q}\right](\xi), & \text{if } q \le \bar{r}, \end{cases} \tag{A.10}$$

which is (3.22). □



(a)



(b)

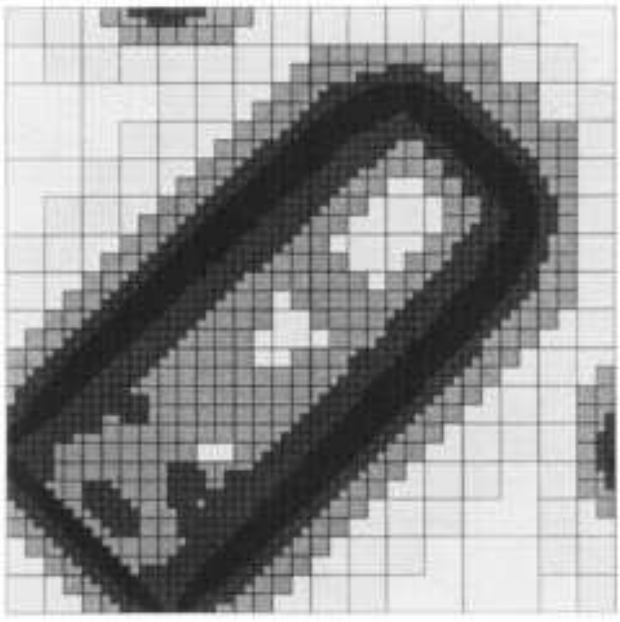
FIG. 9. *Solution at* $n = 700$, $r = 2$, *ENO*.2, *Burgers's equation, IC* (5.1).

(iii) Substituting (A.2a–d) into (A.3a) we get the following for e_1:

$$e_1 = Q_{xy} + \frac{1}{4}(u_1 - u_2 - u_3 + u_4). \tag{A.11}$$

First, let us assume that $p, q > \bar{r} + 1$; that is, u is sufficiently differentiable so that all the terms of the 2-D Taylor expansion exist and are well defined. We shall relate Q_{xy}

(a)



(b)

FIG. 10. *Solution at* $n = 400$, $r = 2$, *ENO*.2, *Burgers's equation, IC* (5.2).

back to Q_x and Q_y and use 1-D interpolation results. By observation,

$$Q_{xy} = \sum_{m=1}^{s-1} \gamma_m (Q_{x_{k+m}} - Q_{x_{k-m}}), \tag{A.12}$$

where

$$Q_{x_k} = Q_x^s(j,k;\bar{u}^l).$$

For each row $k \pm m$ we have, in a similar fashion as (A.5a–b),

$$u^-_{k\pm m} - u_{k\pm m} - Q_{x_{k\pm m}} = c_x h^{\bar{r}}_{x,l} \frac{\partial^{\bar{r}} u}{\partial x^{\bar{r}}}(x^-, y_{k\pm m}) + c_1 h^{\bar{r}+1}_{x,l} \frac{\partial^{\bar{r}+1} u}{\partial x^{\bar{r}+1}}(\xi), \tag{A.13a}$$

$$u^+_{k\pm m} - u_{k\pm m} + Q_{x_{k\pm m}} = c_x h^{\bar{r}}_{x,l} \frac{\partial^{\bar{r}} u}{\partial x^{\bar{r}}}(x^+, y_{k\pm m}) + c_1 h^{\bar{r}+1}_{x,l} \frac{\partial^{\bar{r}+1} u}{\partial x^{\bar{r}+1}}(\xi), \tag{A.13b}$$

where c_x and c_1 are constants, and ξ is used as a generic symbol for a point in the stencil corresponding to the particular row of interpolation. In (A.13a–b), u^+_k and u^-_k are cell averages corresponding to the composite cells made up of cells $(2j, 2k)$ and $(2j, 2k+1)$ and cells $(2j+1, 2k)$ and $(2j+1, 2k+1)$, respectively, (see Fig. A.1); i.e,

$$u^-_k = \frac{1}{2}(\bar{u}^{l-1}_{2j,2k} + \bar{u}^{l-1}_{2j,2k+1}),$$

$$u^+_k = \frac{1}{2}(\bar{u}^{l-1}_{2j+1,2k} + \bar{u}^{l-1}_{2j+1,2k+1}).$$

Similarly, x^+ and x^- are x-coordinates of the centroids corresponding to cell averages u^+_k and $u^-_k \forall k$. (Namely, x^+ is the location of centroids of column $2j+1$, and x^- is that of column $2j$.) If we express the $\frac{\partial^{\bar{r}} u}{\partial x^{\bar{r}}}$ function about the point $(x^\star, y^\star) = (x^{l-1}_{2j,2k}, y^{l-1}_{2j,2k})$ as a Taylor series truncated after the second-order term, we get

$$\frac{\partial^{\bar{r}} u}{\partial x^{\bar{r}}}(x^\pm, y_{k\pm m}) = \frac{\partial^{\bar{r}} u}{\partial x^{\bar{r}}}(x^\star, y^\star) \pm \frac{1}{4} h_{x,l} \frac{\partial^{\bar{r}+1} u}{\partial x^{\bar{r}+1}}(\xi) \pm m h_{y,l} \frac{\partial^{\bar{r}+1} u}{\partial y \partial x^{\bar{r}}}(\xi). \tag{A.14}$$

It follows from (A.13a–b) and (A.14) that

$$\begin{aligned} Q_{x_{k\pm m}} = &-\frac{1}{2}(u^+_{k\pm m} - u^-_{k\pm m}) - c_x h^{\bar{r}}_{x,l} \frac{\partial^{\bar{r}+1} u}{\partial x^{\bar{r}+1}}(x^\star, y^\star) \\ &+ c_1 h^{\bar{r}+1}_{x,l} \frac{\partial^{\bar{r}+1} u}{\partial x^{\bar{r}+1}}(\xi) + c_2 h^{\bar{r}}_{x,l} h_{y,l} \frac{\partial^{\bar{r}+1} u}{\partial y \partial x^{\bar{r}}}(\xi), \end{aligned} \tag{A.15}$$

where coefficients of $(\bar{r}+1)$-order terms were absorbed into constants c_1 and c_2. Substitution of (A.15) into (A.12) and use of c_1 and c_2 as generic coefficients of like terms yield

$$\begin{aligned} Q_{xy} = &-\frac{1}{2}\sum_{m=1}^{s-1} \gamma_m (u^+_{k+m} - u^+_{k-m}) + \frac{1}{2}\sum_{m=1}^{s-1} \gamma_m (u^-_{k+m} - u^-_{k-m}) \\ &+ c_1 h^{\bar{r}+1}_{x,l} \frac{\partial^{\bar{r}+1} u}{\partial x^{\bar{r}+1}}(\xi) + c_2 h^{\bar{r}}_{x,l} h_{y,l} \frac{\partial^{\bar{r}+1} u}{\partial y \partial x^{\bar{r}}}(\xi). \end{aligned} \tag{A.16}$$

Let us introduce now

$$Q^\pm_y = \sum_{m=1}^{s-1} \gamma_m (u^\pm_{k+m} - u^\pm_{k-m}), \tag{A.17}$$

and substitute this expression into (A.16):

$$Q_{xy} = -\frac{1}{2}Q_y^+ + \frac{1}{2}Q_y^- + c_1 h_{x,l}^{\bar{r}+1}\frac{\partial^{\bar{r}+1}u}{\partial x^{\bar{r}+1}}(\xi) + c_2 h_{x,l}^{\bar{r}} h_{y,l}\frac{\partial^{\bar{r}+1}u}{\partial y \partial x^{\bar{r}}}(\xi). \tag{A.18}$$

On the other hand, we observe that (A.17) is also part of an expression for 1-D interpolation along fine grid columns "+" and "−" (on constant j lines) and therefore, as in (A.13a–b), we can write

$$u_2 - u_{24} - Q_y^+ = c_y h_{y,l}^{\bar{r}}\frac{\partial^{\bar{r}}u}{\partial y^{\bar{r}}}(x^+, y^-) + c_3 h_{y,l}^{\bar{r}+1}\frac{\partial^{\bar{r}+1}u}{\partial y^{\bar{r}+1}}(\xi), \tag{A.19a}$$

$$u_4 - u_{24} + Q_y^+ = -c_y h_{y,l}^{\bar{r}}\frac{\partial^{\bar{r}}u}{\partial y^{\bar{r}}}(x^+, y^+) + c_3 h_{y,l}^{\bar{r}+1}\frac{\partial^{\bar{r}+1}u}{\partial y^{\bar{r}+1}}(\xi), \tag{A.19b}$$

$$u_1 - u_{13} - Q_y^- = c_y h_{y,l}^{\bar{r}}\frac{\partial^{\bar{r}}u}{\partial y^{\bar{r}}}(x^-, y^-) + c_3 h_{y,l}^{\bar{r}+1}\frac{\partial^{\bar{r}+1}u}{\partial y^{\bar{r}+1}}(\xi), \tag{A.20a}$$

$$u_3 - u_{13} + Q_y^- = -c_y h_{y,l}^{\bar{r}}\frac{\partial^{\bar{r}}u}{\partial y^{\bar{r}}}(x^-, y^+) + c_3 h_{y,l}^{\bar{r}+1}\frac{\partial^{\bar{r}+1}u}{\partial y^{\bar{r}+1}}(\xi), \tag{A.20b}$$

where now y^+ and y^- are y-coordinates of centroids corresponding to cell averages in rows $2j+1$ and $2j$, respectively; c_3 is a constant that multiplies the appropriate Taylor series term. Expand the function $\frac{\partial^{\bar{r}}u}{\partial x^{\bar{r}}}$ about the point $(x^\star, y^\star)$ to get

$$\frac{\partial^{\bar{r}}u}{\partial x^{\bar{r}}}(x^\pm, y^\pm) = \frac{\partial^{\bar{r}}u}{\partial x^{\bar{r}}}(x^\star, y^\star) \pm \frac{1}{4}h_{x,l}\frac{\partial^{\bar{r}+1}u}{\partial x \partial y^{\bar{r}}}(\xi) \pm \frac{1}{4}h_{y,l}\frac{\partial^{\bar{r}+1}u}{\partial y^{\bar{r}+1}}(\xi). \tag{A.21}$$

Upon expressing Q_y^+ and Q_y^- from the pairs (A.19a–b) and (A.20a–b), respectively, and substituting them, along with (A.21), into (A.18), we finally get

$$\begin{aligned} Q_{xy} = &-\frac{1}{4}(u_1 - u_2 - u_3 + u_4) \\ &+ c_1 h_{x,l}^{\bar{r}+1}\frac{\partial^{\bar{r}+1}u}{\partial x^{\bar{r}+1}}(\xi) + c_2 h_{x,l}^{\bar{r}} h_{y,l}\frac{\partial^{\bar{r}+1}u}{\partial y \partial x^{\bar{r}}}(\xi) \\ &+ c_3 h_{y,l}^{\bar{r}+1}\frac{\partial^{\bar{r}+1}u}{\partial y^{\bar{r}+1}}(\xi) + c_4 h_{x,l} h_{y,l}^{\bar{r}}\frac{\partial^{\bar{r}+1}u}{\partial x \partial y^{\bar{r}}}(\xi). \end{aligned} \tag{A.22}$$

Substitution of (A.22) into (A.11) yields now the first branches of (3.20b–c). The second branches are obtained by replacing the 2-D Taylor expansions used above by the difference of two expansions, one to the left and one to the right of each discontinuity in the derivative. The cancellations of the $\bar{r}$th-order terms will still occur, the same way as in the smooth case. In effect, we get (A.22), with the exception that the $(\bar{r}+1)$st-order derivatives will be replaced by the $(p+1)$st- and $(q+1)$st-order jumps. The final result will thus be summarized by (3.20a–c). □

Appendix B. In this appendix we show that the RHS of (2.4) fits the multiresolution representation of cell averages. We do this by working directly with the fluxes that comprise the RHS, without assigning a specific method of evaluation to the numerical fluxes. In this sense, the RHS of (6.4) can be defined for each multiresolution level l as

$$S_{j,k}^l = -\frac{1}{h_{x,l}}(\bar{f}_{j+\frac{1}{2},k}^l - \bar{f}_{j-\frac{1}{2},k}^l) - \frac{1}{h_{y,l}}(\bar{g}_{j,k+\frac{1}{2}}^l - \bar{g}_{j,k-\frac{1}{2}}^l), \tag{B.1}$$

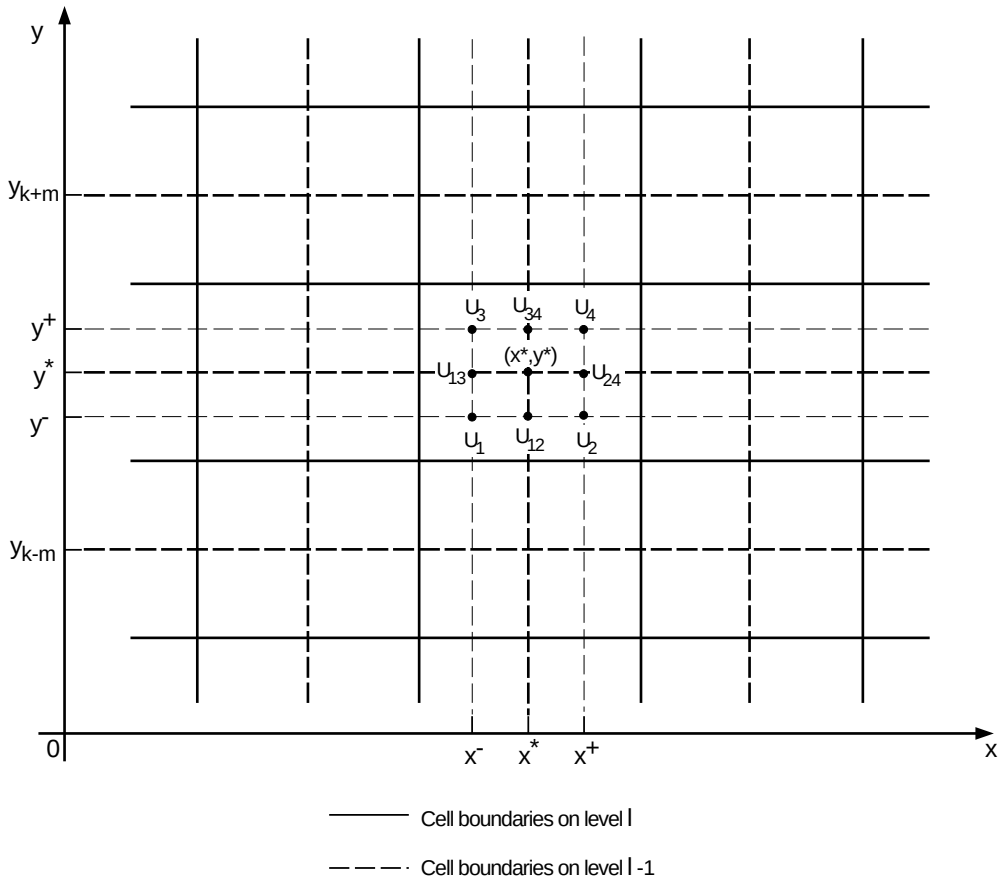


FIG. A.1. *Estimation of* 2*-D regularity coefficients (used in Appendix* A*).*

where

$$\bar{f}^l_{j\pm\frac{1}{2},k} = \int_{y^l_{k-\frac{1}{2}}}^{y^l_{k+\frac{1}{2}}} f(u(x^l_{j\pm\frac{1}{2}}, y, t))dy, \tag{B.2a}$$

$$\bar{g}^l_{j,k\pm\frac{1}{2}} = \int_{x^l_{j-\frac{1}{2}}}^{x^l_{j+\frac{1}{2}}} g(u(x, y^l_{k\pm\frac{1}{2}}, t))dx. \tag{B.2b}$$

Here we have used the half indices to denote the cell interfaces; for the purposes of this proof, the cell centroids are at the locations x^l_j, y^l_k of whole subscripts from which we have also dropped the subscript that is kept constant, for more clarity.

Because of the additivity of the integrals in (B.2a–b), we can write

$$\begin{aligned}\bar{f}^l_{j\pm\frac{1}{2},k} &= \int_{y^{l-1}_{2k-\frac{1}{2}}}^{y^{l-1}_{2k+\frac{1}{2}}} f(u(x^l_{j\pm\frac{1}{2}}, y, t))dy + \int_{y^{l-1}_{2k+\frac{1}{2}}}^{y^{l-1}_{2k+\frac{3}{2}}} f(u(x^l_{j\pm\frac{1}{2}}, y, t))dy \\ &= \bar{f}^{l-1}_{2j+\frac{1}{2}\pm1,2k} + \bar{f}^{l-1}_{2j+\frac{1}{2}\pm1,2k+1},\end{aligned} \tag{B.3a}$$

$$\bar{g}^l_{j,k\pm\frac{1}{2}} = \int_{x^{l-1}_{2j-\frac{1}{2}}}^{x^{l-1}_{2j+\frac{1}{2}}} g(u(x, y^l_{k\pm\frac{1}{2}}, t))dx + \int_{x^{l-1}_{2j+\frac{1}{2}}}^{x^{l-1}_{2j+\frac{3}{2}}} g(u(x, y^l_{k\pm\frac{1}{2}}, t))dx$$
$$= \bar{g}^{l-1}_{2j,2k+\frac{1}{2}\pm1} + \bar{g}^{l-1}_{2j+1,2k+\frac{1}{2}\pm1}. \tag{B.3b}$$

Substituting (B.3a–b) into (B.1), we can now relate the RHS on two adjacent levels:

$$\begin{aligned} S^l_{j,k} =& -\frac{1}{2h_{x,l-1}}(\bar{f}^{l-1}_{2j+\frac{3}{2},2k} + \bar{f}^{l-1}_{2j+\frac{3}{2},2k+1} - \bar{f}^{l-1}_{2j-\frac{1}{2},2k} - \bar{f}^{l-1}_{2j-\frac{1}{2},2k+1}) \\ & -\frac{1}{2h_{y,l-1}}(\bar{g}^{l-1}_{2j,2k+\frac{3}{2}} + \bar{g}^{l-1}_{2j+1,2k+\frac{3}{2}} - \bar{g}^{l-1}_{2j,2k-\frac{1}{2}} - \bar{g}^{l-1}_{2j+1,2k-\frac{1}{2}}) \\ =& \frac{1}{4}(S^{l-1}_{2j,2k} + S^{l-1}_{2j+1,2k} + S^{l-1}_{2j,2k+1} + S^{l-1}_{2j+1,2k+1}), \end{aligned} \tag{B.4}$$

where we have added and subtracted fluxes on the $j+\frac{1}{2}$ and $k+\frac{1}{2}$ cell interfaces. (B.4) shows that the cell average representation of the RHS is meaningful, and it describes, in fact, the averaging process for this multiresolution representation when we compute the coarser level RHS from the finer levels. □

REFERENCES

[1] R. Abgrall and A. Harten, *Multiresolution Representation in Unstructured Meshes:* I. *Preliminary Report*, UCLA CAM Report 94-20, UCLA, Los Angeles, CA, 1994.

[2] B. L. Bihari, *Implementation of a High Order* 2*-D ENO Scheme*, Computational Fluid Dynamics Department, Rockwell Science Center, Thousand Oaks, CA, preprint.

[3] B. L. Bihari, *Multiresolution Schemes for the Numerical Solution of* 2*-D Conservation Laws*, II, in preparation.

[4] B. L. Bihari and A. Harten, *Application of generalized wavelets: An adaptive multiresolution scheme*, J. Comput. Appl. Math., 61 (1995), pp. 275–321.

[5] J. B. Goodman and R. J. LeVeque, *On the accuracy of stable schemes for* 2*D scalar conservation laws*, Math. Comp., 45 (1985), pp. 15–21.

[6] A. Harten, *Adaptive Multiresolution Schemes for Shock Computations*, J. Comput. Phys., 115 (1994), pp. 319–338.

[7] A. Harten, *Discrete multiresolution analysis and generalized wavelets*, Appl. Numer. Math., 12 (1993), pp. 153–193.

[8] A. Harten, *Multiresolution Algorithms for the Numerical Solutions of Hyperbolic Conservation Laws*, UCLA CAM Report 93-03, UCLA, Los Angeles, CA, 1993; Comm. Pure and Appl. Math., to appear.

[9] A. Harten, *Multiresolution Representation of Cell Averaged Data*, UCLA CAM Report 94-21, UCLA, Los Angeles, CA, 1994.

[10] A. Harten, *Multiresolution Representation of Data* I. *Preliminary Report*, UCLA CAM Report 93-13, UCLA, Los Angeles, CA, 1993.

[11] A. Harten, *Multiresolution Representation of Data* II. *General Framework*, UCLA CAM Report 94-10, UCLA, Los Angeles, CA, 1994.

[12] A. Harten and S. R. Chakravarthy, *Multidimensional ENO Schemes for General Geometries*, UCLA CAM Report 91-16, UCLA, Los Angeles, CA, 1991.

[13] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy, *Uniform high order accurate essentially non-oscillatory schemes*, III, J. Comput. Phys., 71 (1987), pp. 231–303.

[14] A. Harten and S. Osher, *Uniformly high order accurate nonoscillatory schemes*, I, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.

[15] A. Harten and I. Yad-Shalom, *Fast multiresolution algorithms for matrix-vector multiplication*, SIAM J. Numer. Anal., 31 (1994), pp. 1191–1218.

[16] C.-W. Shu, G. Erlebacher, T. A. Zang, D. Whitaker, and S. Osher, *High-Order ENO Schemes Applied to Two- and Three-Dimensional Compressible Flow*, ICASE Report No. 91-38, Institute for Computer Applications in Science and Engineering, Hampton, VA, 1991.