SIAM J. SCI. COMPUT.
Vol. 31, No. 2, pp. 1221–1248

A HIERARCHICAL MULTIRESOLUTION ADAPTIVE MESH REFINEMENT FOR THE SOLUTION OF EVOLUTION PDEs*

S. JAIN†, P. TSIOTRAS†, AND H.-M. ZHOU‡

Abstract. In this paper, we propose a novel multiresolution adaptive mesh refinement algorithm for solving initial-boundary value problems (IBVP) for evolution PDEs. The proposed algorithm dynamically adapts the grid to any existing or emerging irregularities in the solution, by refining the grid only at those places where the solution exhibits sharp features. The main advantage of the proposed grid adaptation method is that it results in a grid with a fewer number of nodes when compared to adaptive grids generated by existing multiresolution-based mesh refinement techniques. Several examples show the robustness and stability of the proposed algorithm.

Key words. mesh refinement, multiresolution, adaptive PDE methods, ENO, weighted ENO

AMS subject classifications. 65M50, 65N50, 65M06, 65D05

DOI. 10.1137/070708329

1. Introduction.

It is well known that the solution of evolution PDEs is often not smooth even if the initial data are smooth. For instance, shocks may develop in hyperbolic conservation laws. To capture discontinuities in the solution with high accuracy, one needs to use a fine resolution grid. The use of a uniformly fine grid requires a large amount of computational resources in terms of both CPU time and memory. Hence, in order to solve evolution equations in a computationally efficient manner, the grid should adapt dynamically to reflect local changes in the solution.

Several adaptive gridding techniques for solving PDEs have been proposed in the literature. A nice survey of the early works on the subject can be found in [5, 47]. Currently, popular adaptive methods for solving PDEs are (i) moving mesh methods [2, 1, 4, 7, 6, 13, 14, 17, 29, 32, 35, 36, 45], in which an equation is derived that moves a grid of a fixed number of finite difference cells or finite elements so as to follow and resolve any local irregularities in the solution; (ii) the so-called adaptive mesh refinement method [8, 9, 10, 11], in which the mesh is refined locally based on the difference between the solutions computed on the coarser and the finer grids, and (iii) wavelet-based or multiresolution-based methods [3, 12, 15, 18, 19, 20, 23, 24, 26, 37, 38, 48, 49, 50], which take advantage of the fact that functions with localized regions of sharp transition can be compressed efficiently. Our proposed method falls under this latter category.

Mallat [34] formulated the basic idea of multiresolution analysis for orthonormal wavelets in $L^2(\mathbb{R})$. Harten [19, 20, 21] later proposed a general framework for multiresolution representation of data by integrating ideas from three different fields, namely, theory of wavelets, numerical solution of PDEs, and subdivision schemes. Recently, Alves et al. [3] proposed an adaptive multiresolution scheme, similar to the multiresolution approach proposed by Harten [19, 20] and Holmstrom [23] for solving hyperbolic

*Received by the editors November 15, 2007; accepted for publication (in revised form) August 18, 2008; published electronically December 31, 2008. This work was partially supported by NSF through awards CMS-0510259 and DMS-0410062.

http://www.siam.org/journals/sisc/31-2/70832.html

†School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0150 (sachin.jain@gatech.edu, tsiotras@gatech.edu).

‡School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332-0160 (hmzhou@math.gatech.edu).

PDEs. These approaches share similar underlying ideas. Namely, the first step is to interpolate the function values at the points belonging to a particular resolution level from the corresponding points at the coarser level and find the interpolative error at the points of that particular resolution level. Once this step has been performed for all resolution levels, all the points that have an interpolative error greater than a prescribed threshold are added to the grid, along with their neighboring points at the same level and the neighboring points at the next finer level. The main difference between these approaches is that in Harten's approach [19, 20], the solution for each time step is represented on the finest grid, and one calculates the interpolative errors at all the points of the finest grid at each mesh refinement step. On the other hand, Holmstrom [23] and Alves et al. [3], compute the interpolative error only at the points that are in the adaptive grid. If a value that does not exist is needed, Holmstrom interpolates the required function value recursively from coarser scales. Alternatively, Alves et al. [3] add to the grid the points that were used to predict the function values at all previously added points, in order to compute the interpolative error during the next mesh adaptation. Parallel to Harten's original idea, a modified approach has also been developed by Cohen, Müller, and coauthors for the solution of evolution PDEs [15, 18, 37, 38].

In this paper, we first propose a novel multiresolution scheme for data compression, which results in a higher compression rate compared to the multiresolution approach by Harten [19, 20, 21] for the same desired accuracy. Subsequently, we apply the proposed encoding scheme to solve initial-boundary value problems (IBVP) encountered in evolution PDEs. The proposed multiresolution scheme for data compression works with any of the interpolation techniques mentioned in [21]. One of the key features of our algorithm is that it is a "top-down" (from coarse to fine scale) approach, and we use the most recently updated information to make predictions. Moreover, our interpolations are not restricted to the use of only the retained points at the coarser level, but also use the retained points at the same level (and even the next finer level in the case of solving PDEs). This allows for a more accurate interpolation, which, in turn, leads to fewer points in the final grid. In the proposed algorithm, we continuously keep on updating the grid as we go from the coarsest level to finer levels. If the interpolative error at a point that belongs to a particular level is greater than the prescribed threshold, we add that point to the grid. In the case of solving PDEs, we also add the neighboring points at the same level and the neighboring points at the next level to the grid. We make use of the fact that the point at which the interpolative error is greater than a prescribed threshold, this point is added to the grid, and, in addition, it can be used to predict the remaining points at the same level and the levels below it. Moreover, for refining the mesh for solving evolution PDEs, we predict the function value at a particular point only from the points that are already present in the grid; hence we avoid recursive interpolations from the coarser scales as is done by Holmstrom [23]. At the same time, we do not need to add any extra points to the grid that are required just for computing the interpolative errors at the next mesh refinement step, as is done by Alves et al. [3].

The paper is organized as follows. In the first part of the paper, we start by formulating the problem, and we present the new multiresolution scheme for data compression, followed by the mesh refinement algorithm for the solution of evolution PDEs. Next, we compare the proposed data compression scheme with Harten's data compression scheme (and the mesh refinement approach with the one of Alves et al. for the case of solution of hyperbolic PDEs). We show that the proposed algorithm

results, in general, in a fewer number of grid points compared to Harten's approach (for the case of data compression) and the approach of Alves et al. [3] (for the case of the mesh refinement for the solution of PDEs). In the second part of the paper, we present an algorithm for solving the IBVP for evolution equations on an adaptive nonuniform grid, generated using the proposed mesh refinement technique. This analysis is followed by several numerical examples that show the robustness of the proposed approach and the advantages in terms of computational time compared to the uniform mesh case.

2. Problem statement. Many problems in engineering and physics can be written in the form of an IBVP for an evolution equation:

$$\text{(IBVP)}: \quad \begin{cases} u_t + f(u_{xx}, u_x, u, x) = 0 & \text{in } \mathcal{D} \times (0, \infty), \\ u = g & \text{on } \overline{\mathcal{D}} \times \{t = 0\}, \end{cases} \tag{2.1}$$

where $\overline{\mathcal{D}} = \mathcal{D} \cup \partial\mathcal{D}$, with $\mathcal{D} \subset \mathbb{R}$ bounded. The function $f : \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^m \times \mathcal{D} \to \mathbb{R}$ and the initial function $g : \overline{\mathcal{D}} \to \mathbb{R}^m$ are given. The unknown is the function $u : \overline{\mathcal{D}} \times [0, \infty) \to \mathbb{R}^m$. The algorithm proposed in this paper works for other boundary conditions as well, but, for simplicity in the analysis below, we only use *periodic*, *Dirichlet*, and *Neumann* boundary conditions. Without loss of generality, we will further assume that $\mathcal{D} = (0, 1)$.

In (IBVP), the initial function g can be irregular. Even if g is smooth, discontinuities such as shocks (in hyperbolic conservation laws) and kinks (in Hamilton–Jacobi (HJ) equations) can develop in the solution u at some later time. Therefore, we would like to adapt the grid dynamically to any existing or emerging irregularities in the solution, instead of using a fine mesh over the whole spatial and temporal domain. In the next section, we propose a novel grid refinement technique for solving (IBVP) in a computationally efficient manner.

3. Adaptive gridding. First, we give a brief overview on dyadic grids, which are used in the proposed multiresolution mesh refinement scheme for solving (IBVP).

3.1. Dyadic grids. Since $\overline{\mathcal{D}} = [0, 1]$, we consider dyadic grids of the form

$$\mathcal{V}_j = \{x_{j,k} \in [0,1] : x_{j,k} = k/2^j,\ 0 \le k \le 2^j\}, \quad J_{\min} \le j \le J_{\max}, \tag{3.1}$$

where j denotes the resolution level, k the spatial location, and $J_{\min},\ J_{\max} \in \mathbb{Z}_0^+$. We denote by $\mathcal{W}_j$ the set of grid points belonging to $\mathcal{V}_{j+1} \setminus \mathcal{V}_j$. Therefore,

$$\mathcal{W}_j = \{y_{j,k} \in [0,1] : y_{j,k} = (2k+1)/2^{j+1},\ 0 \le k \le 2^j - 1\}, \quad J_{\min} \le j \le J_{\max} - 1. \tag{3.2}$$

Throughout the rest of this paper, we will use the notations $x_{j,k}$ and $y_{j,k}$ to represent elements of $\mathcal{V}_j$ and $\mathcal{W}_j$, respectively. Hence, $x_{j+1,k} \in \mathcal{V}_{j+1}$ is given by

$$x_{j+1,k} = \begin{cases} x_{j,k/2}, & \text{if } k \text{ is even}, \\ y_{j,(k-1)/2}, & \text{otherwise}. \end{cases} \tag{3.3}$$

An example of a dyadic grid with $J_{\min} = 0$ and $J_{\max} = 5$ is shown in Figure 3.1.

With a slight abuse of notation, we write $\mathcal{V}_{j+1} = \mathcal{V}_j \oplus \mathcal{W}_j$, although $\mathcal{W}_j$ is not an orthogonal complement of $\mathcal{V}_j$ in $\mathcal{V}_{j+1}$. The subspaces $\mathcal{V}_j$ are nested, $\mathcal{V}_{J_{\min}} \subset \mathcal{V}_{J_{\min}+1} \cdots \subset \mathcal{V}_{J_{\max}}$, with $\lim_{J_{\max} \to \infty} \mathcal{V}_{J_{\max}} = \overline{\mathcal{D}}$. The sequence of subspaces $\mathcal{W}_j$ satisfy $\mathcal{W}_j \cap \mathcal{W}_\ell = \varnothing$ for $j \neq \ell$.

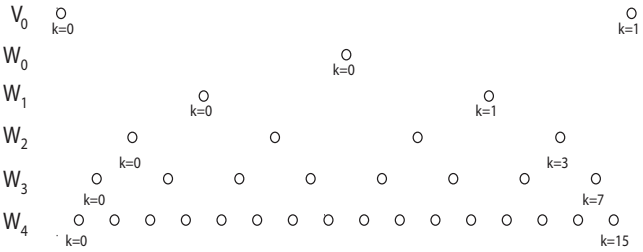


FIG. 3.1. *Example of a dyadic grid.*

Before presenting the multiresolution grid adaptation scheme for solving IBVP for evolution PDEs, we propose a novel multiresolution scheme for data compression that will form the basis of our grid adaptation scheme for solving (IBVP).

3.2. Data compression. Suppose $g : \overline{\mathcal{D}} \to \mathbb{R}$ is specified on a grid $\mathcal{V}_{J_{\max}}$,

$$\mathcal{U}_{J_{\max}} = \{g_{j,k} : x_{j,k} \in \mathcal{V}_{J_{\max}}\}, \tag{3.4}$$

where $g_{j,k} = g(x_{j,k})$. Let $\mathcal{I}(x; \mathcal{X}_{\mathsf{Grid}})$ denote *any* pth order interpolation of $\mathcal{U} = \{g_{j,k} : x_{j,k} \in \mathcal{X}_{\mathsf{Grid}}\}$, where $\mathcal{X}_{\mathsf{Grid}} = \{x_{j_\ell,k_\ell}\}_{\ell=i}^{i+p} \subset \mathsf{Grid}$, where

$$\begin{aligned}\mathsf{Grid} =&\{x_{j_i,k_i} : x_{j_i,k_i} \in [0,1],\ 0 \le k_i \le 2^{j_i},\ J_{\min} \le j_i \le J_{\max},\ \text{for } i = 0 \dots N,\\ &\text{and } x_{j_i,k_i} < x_{j_{i+1},k_{i+1}},\ \text{for } i = 0 \dots N-1\} \subset \mathcal{V}_{J_{\max}}\end{aligned} \tag{3.5}$$

and $x \in [x_{j_i,k_i}, x_{j_{i+p},k_{i+p}}]$. In (3.5), Grid can be uniform or nonuniform. Then the encoding algorithm for compressing the signal g is as follows.

Step 1. Initialize an intermediate grid $\mathsf{Grid}_{\text{int}} = \mathcal{V}_{J_{\min}}$, with function values $\mathcal{U}_{\text{int}} = \mathcal{U}_{\min}$, where $\mathcal{U}_{\min} = \{g_{J_{\min},k} : 0 \le k \le 2^{J_{\min}}\}$. Set $j = J_{\min}$.

Step 2. DO for $k = 0, \dots, 2^j$.

(a) Compute the interpolated function value $\hat{g}(y_{j,k}) = \mathcal{I}(y_{j,k}, \mathcal{X}_{\mathsf{Grid}_{\text{int}}})$.

(b) If the interpolative error coefficient at the point $y_{j,k}$,

$$d_{j,k} = |g(y_{j,k}) - \hat{g}(y_{j,k})| > \epsilon, \tag{3.6}$$

where ϵ is the prescribed threshold, then add $y_{j,k}$ to the intermediate grid $\mathsf{Grid}_{\text{int}}$ and the corresponding function value $g(y_{j,k})$ to $\mathcal{U}_{\text{int}}$.

Step 3. Increment j by 1. If $j < J_{\max}$, go to Step 2, otherwise move on to the next step.

Step 4. Terminate the algorithm. The final nonuniform grid representing the compressed information is $\mathsf{Grid}_{\text{M}} = \mathsf{Grid}_{\text{int}}$, and the corresponding function values are the set $\mathcal{U}_{\text{M}} = \mathcal{U}_{\text{int}}$.

If we represent the above nonlinear encoding procedure by an operator M, then we can write

$$\mathcal{U}_{\text{M}} = M\mathcal{U}_{\text{J}_{\max}}. \tag{3.7}$$

One should note that in Harten's approach [19, 20, 21], the points of a particular resolution level $\mathcal{W}_j$ are interpolated only from the corresponding points belonging to $\mathcal{V}_j$. In our approach, instead, we continuously keep on updating the grid, and the points $\{g(y_{j,k})\}_{k=0}^{2^j-1}$ of level $\mathcal{W}_j$ are interpolated from the function values at the points

in $\mathcal{V}_j \oplus \mathcal{W}_j$. Hence, by making use of the extra information from levels $\mathcal{W}_j$—which, in any case, will be added to the adaptive grid—we are able to reduce the number of grid points in the final grid. This process results in higher compression factors, as will be shown in section 3.4 via several examples.

The interpolation operator $\mathcal{I}$ can be easily constructed using piecewise-polynomial interpolation, essentially nonoscillatory (ENO) interpolation, cubic-spline interpolation, trigonometric interpolation, etc., [21]. For piecewise-polynomial interpolation, the stencil $\mathcal{X}_{\mathsf{Grid}_{\mathrm{int}}}$ consists of the $p+1$ nearest points to x in $\mathsf{Grid}_{\mathrm{int}}$. By $p+1$ nearest points, here we mean one neighboring point on the left of x, one neighboring point on the right of x, and the remaining $p-1$ points are the points nearest to x in the set $\mathsf{Grid}_{\mathrm{int}}$. In case two points are at the same distance from x, that is, if a point on the left and a point on the right are equidistant to x, then we choose a point so as to equalize the number of points on both sides. One may use Neville's algorithm to construct the respective interpolating polynomials on the fly. For ENO interpolation, the stencil $\mathcal{X}_{\mathsf{Grid}_{\mathrm{int}}}$ consists of one neighboring point on the left and one neighboring point on the right of x in the set $\mathsf{Grid}_{\mathrm{int}}$, and the remaining $p-1$ points are selected from the set $\mathsf{Grid}_{\mathrm{int}}$ that give the least oscillatory polynomial. For more details on ENO interpolation, the reader is referred to [21, 39].

Next, we give the decoding algorithm, that is, the algorithm for computing

$$\hat{\mathcal{U}}_{J_{\max}} = M^{-1}\mathcal{U}_{\mathrm{M}}. \tag{3.8}$$

One way of decoding the information back from the compressed signal in nonlinear schemes is to keep track of the stencils that were used for interpolating the function values at a particular point while encoding the information and use the same stencils to decode the information from the compressed signal. An alternative way (described below) is to follow the same approach as in the encoding algorithm.

Step 1. Initialize $\mathsf{Grid}_{\mathrm{int}} = \mathcal{V}_{J_{\min}}$, with function values $\hat{\mathcal{U}}_{J_{\max}} = \mathcal{U}_{\mathrm{int}} = \mathcal{U}_{\min}$, where $\mathcal{U}_{\min} = \{g_{J_{\min},k} : 0 \le k \le 2^{J_{\min}}\}$. Set $j = J_{\min}$.

Step 2. DO for $k = 0, \ldots, 2^j$.
If $g(y_{j,k}) \in \mathcal{U}_{\mathrm{M}}$, then add $g(y_{j,k})$ to $\hat{\mathcal{U}}_{J_{\max}}$, $\mathcal{U}_{\mathrm{int}}$ and $y_{j,k}$ to $\mathsf{Grid}_{\mathrm{int}}$, otherwise add $\hat{g}(y_{j,k}) = \mathcal{I}(y_{j,k}, \mathcal{X}_{\mathsf{Grid}_{\mathrm{int}}})$ to $\hat{\mathcal{U}}_{J_{\max}}$.

Step 3. Increment j by 1. If $j < J_{\max}$, go to Step 2, otherwise move on to the next step.

Step 4. Terminate the algorithm.

Next, we derive an error estimate between the original signal $\mathcal{U}_{J_{\max}}$ and the decoded signal $\hat{\mathcal{U}}_{J_{\max}}$.

PROPOSITION 1. *Let $\mathcal{U}_{J_{\max}}$ be defined as in* (3.4)*, and let $\hat{\mathcal{U}}_{J_{\max}} = M^{-1}\mathcal{U}_M$, where M^{-1} denotes the decoding algorithm described above. Then for $1 \le m < \infty$,*

$$E_m(g) = \|\mathcal{U}_{J_{\max}} - \hat{\mathcal{U}}_{J_{\max}}\|_m = \left(\frac{1}{2^{J_{\max}}+1}\sum_{k=0}^{2^{J_{\max}}} |g_{J_{\max},k} - \hat{g}_{J_{\max},k}|^m\right)^{\frac{1}{m}} \le \left(\frac{2^{J_{\max}} - 2^{J_{\min}}}{2^{J_{\max}}+1}\right)^{\frac{1}{m}} \epsilon \tag{3.9}$$

and

$$E_\infty(g) = \|\mathcal{U}_{J_{\max}} - \hat{\mathcal{U}}_{J_{\max}}\|_\infty = \max_{0 \le k \le 2^{J_{\max}}} |g_{J_{\max},k} - \hat{g}_{J_{\max},k}| \le \epsilon. \tag{3.10}$$

Proof. First, we note that

$$|g_{J_{\min},k} - \hat{g}_{J_{\min},k}| = 0, \quad k = 0, \ldots, 2^{J_{\min}}, \tag{3.11}$$

since $g_{J_{\min},k} \in \mathcal{U}_{\mathrm{M}}$ for all $k = 0, \ldots, 2^{J_{\min}}$. Next, since the function values in the set $\hat{\mathcal{U}}_{J_{\max}}$ are interpolated directly only from the function values in $\mathcal{U}_{\mathrm{M}}$, we have direct control over the error. Therefore,

$$|g(y_{j,k}) - \hat{g}(y_{j,k})| \leq \epsilon, \quad k = 0, \ldots, 2^j - 1 \tag{3.12}$$

for $j = J_{\min}, \ldots, J_{\max} - 1$. Hence, we have

$$\|\mathcal{U}_{J_{\max}} - \hat{\mathcal{U}}_{J_{\max}}\|_\infty = \max_{0 \leq k \leq 2^{J_{\max}}} |g_{j,k} - \hat{g}_{j,k}| \leq \epsilon. \tag{3.13}$$

For $1 \leq m < \infty$, we have

$$\|\mathcal{U}_{J_{\max}} - \hat{\mathcal{U}}_{J_{\max}}\|_m^m \tag{3.14}$$

$$= \frac{1}{2^{J_{\max}} + 1} \left(\sum_{k=0}^{2^{J_{\min}}} |g_{J_{\min},k} - \hat{g}_{J_{\min},k}|^m + \sum_{j=J_{\min}}^{J_{\max}-1} \sum_{k=0}^{2^j - 1} |g(y_{j,k}) - \hat{g}(y_{j,k})|^m \right) \tag{3.15}$$

$$\leq \frac{\epsilon^m}{2^{J_{\max}} + 1} \sum_{j=J_{\min}}^{J_{\max}-1} \sum_{k=0}^{2^j - 1} 1 = \epsilon^m \frac{2^{J_{\max}} - 2^{J_{\min}}}{2^{J_{\max}} + 1}. \tag{3.16}$$

Consequently, for $1 \leq m < \infty$,

$$\|\mathcal{U}_{J_{\max}} - \hat{\mathcal{U}}_{J_{\max}}\|_m \leq \left(\frac{2^{J_{\max}} - 2^{J_{\min}}}{2^{J_{\max}} + 1} \right)^{\frac{1}{m}} \epsilon, \tag{3.17}$$

which completes the proof. ∎

Example 1. Consider $g_1 : \overline{\mathcal{D}} \to \mathbb{R}$,

$$g_1(x) = \begin{cases} 1, & \frac{1}{3} \leq x \leq \frac{2}{3}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.18}$$

and $g_2 : \overline{\mathcal{D}} \to \mathbb{R}$,

$$g_2(x) = \begin{cases} 0, & 0 \leq x < \frac{1}{6}, \\ 1, & \frac{1}{6} \leq x < \frac{1}{3}, \\ 0, & \frac{1}{3} \leq x < \frac{1}{2}, \\ \sin(\pi x), & \frac{1}{2} \leq x < \frac{2}{3}, \\ 0, & \frac{2}{3} \leq x < \frac{5}{6}, \\ x, & \frac{5}{6} \leq x \leq 1. \end{cases} \tag{3.19}$$

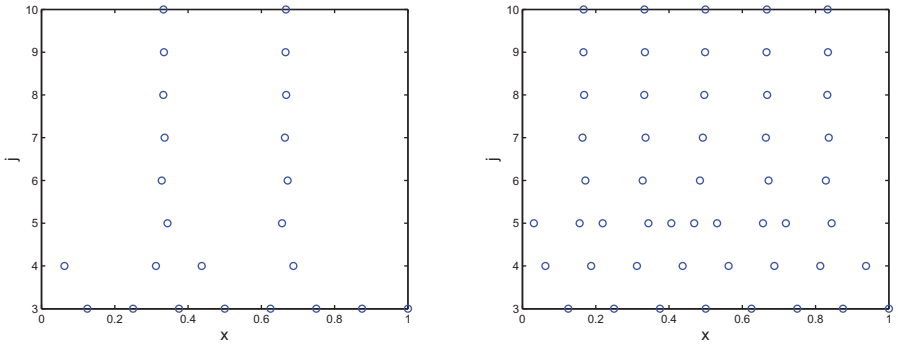
For this example, we consider a grid with $J_{\min} = 2$ and $J_{\max} = 10$ and use ENO interpolation for the encoding and the decoding algorithms described above. For both g_1 and g_2, the data compression factor

$$C = \frac{2^{J_{\max}} + 1 - N_{\mathrm{g_p}}}{2^{J_{\max}} + 1} \times 100\%, \tag{3.20}$$

where $N_{\mathrm{g_p}}$ denotes the number of grid points, along with the decoding errors E_m, $m = 1, 2, \infty$, with an interpolating polynomial of degree $p = 3$ and different thresholds

TABLE 3.1
Example 1. *Data compression along with the decoding errors for the proposed approach.*

	ϵ	C	E_∞	E_1	E_2
g_1	10^{-3}	97.56	0	0	0
g_2	10^{-3}	94.93	2.2741×10^{-5}	8.2103×10^{-7}	2.8111×10^{-6}
g_2	10^{-7}	93.85	9.0426×10^{-8}	3.7983×10^{-9}	1.2662×10^{-8}



(a) $g_1(x)$. (b) $g_2(x)$.

FIG. 3.2. *Example* 1. *Grid point distribution for* $\epsilon = 1 \times 10^{-3}$.

ϵ, are summarized in Table 3.1. First, we consider $\epsilon = 10^{-3}$ for both the functions g_1 and g_2. The proposed algorithm compressed the given signals g_1 and g_2 using only 25 and 52 points, respectively. The decoding errors E_m $(m = 1, 2, \infty)$ are well below the threshold for both these functions. The grid point distributions for both g_1 and g_2 are shown in Figure 3.2. Next, for g_2, we decrease the threshold to 10^{-7}. It is observed that the proposed encoding algorithm increased the number of points used for compressing the signal; once again, the decoding errors are below the prescribed threshold.

Next, we present the grid adaptation algorithm for solving (IBVP) using the decoding algorithm mentioned above.

3.3. Grid adaptation for the solution of (IBVP). Assume we are given a nonuniform grid of the form (3.5). For simplicity, we denote by $u^n_{j,k}$ the value of $u(x,t)$ evaluated at $x = x_{j,k}$ and $t = t_n$, where $0 \le k \le 2^j$, $J_{\min} \le j \le J_{\max}$, $n \in \mathbb{Z}^+_0$, $t_0 = 0$, $t_n = t_{n-1} + \Delta t_n$ for $n > 0$, and Δt_n is the time step based on the Courant–Friedrichs–Levy (CFL) condition [46] for hyperbolic equations and the von Neumann condition [46] for all other evolution equations. The "top-down" approach of our algorithm allows one to add and remove points using the most recently updated information. To this end, suppose $u(x, t_n)$ is specified on the grid $\mathsf{Grid}_{\text{old}}$, with corresponding solution values $\mathcal{U}_{\text{old}} = \{u^n_{j,k} : x_{j,k} \in \mathsf{Grid}_{\text{old}}\}$, where $\mathsf{Grid}_{\text{old}}$ can be either regular or irregular.[1] We assume $\mathsf{Grid}_{\text{old}} \supseteq \mathcal{V}_{J_{\min}}$. Our aim is to find a new grid $\mathsf{Grid}_{\text{new}}$, by adding or removing points from $\mathsf{Grid}_{\text{old}}$, reflecting local changes in the solution. To this end, we initialize an intermediate grid $\mathsf{Grid}_{\text{int}} = \mathcal{V}_{J_{\min}}$, with the function values $\mathcal{U}_{\text{int}} = \{u^n_{J_{\min},k} : u^n_{J_{\min},k} \in \mathcal{U}_{\text{old}},\ 0 \le k \le 2^{J_{\min}}\}$, and we set $j = J_{\min}$. The mesh refinement algorithm proceeds as follows.

[1] Typically, $\mathsf{Grid}_{\text{old}}$ at time $t = 0$ is regular with $\mathsf{Grid}_{\text{old}} = \mathcal{V}_{J_{\max}}$.

Step 1. Find the points belonging to the intersection of $\mathcal{W}_j$ and $\mathsf{Grid}_{\text{old}}$, that is,

$$Y = \{y_{j,k_i} : y_{j,k_i} \in \mathcal{W}_j \cap \mathsf{Grid}_{\text{old}}, \text{ for } i = 1, \ldots, M,\ 1 \leq M \leq 2^j\}. \tag{3.21}$$

Step 2. Set $i = 1$.

(a) Compute the interpolated function values at point $y_{j,k_i} \in Y$, $\hat{u}(y_{j,k_i})$, that is, $\hat{u}_\ell(y_{j,k_i}) = \mathcal{I}(y_{j,k_i}, \mathcal{X}_{\mathsf{Grid}_{\text{int}}})$, where $\hat{u}_\ell$ is the ℓth element of $\hat{u}$ for $\ell = 1, \ldots, m$.[2]

(b) If at the point y_{j,k_i},[3]

$$d_{j,k_i}(u^n) = \max_{\ell=1,\ldots,m} |u_\ell(y_{j,k_i}, t_n) - \hat{u}_\ell(y_{j,k_i})| < \epsilon, \tag{3.22}$$

go to Step 2(f), otherwise add y_{j,k_i} to the intermediate grid $\mathsf{Grid}_{\text{int}}$ and move on to the next step.

(c) Add to $\mathsf{Grid}_{\text{int}}$ N_1 points on the left and N_1 points on the right neighboring to the point y_{j,k_i} in $\mathcal{W}_j$. This step accounts for the possible displacement of any sharp features of the solution during the next time integration step. The value of N_1 dictates the frequency of mesh adaptation and is provided by the user. The larger the N_1, the smaller the frequency of mesh adaptation will be, at the expense of a larger number of grid points in the adaptive grid. Hence, there is a trade-off between the frequency of mesh adaptation and the number of grid points.

(d) Add to $\mathsf{Grid}_{\text{int}}$ $2N_2$ neighboring points at the next finer level $\{y_{j+1,2k_i+\ell}\}_{\ell=-N_2+1}^{N_2}$, where $1 \leq N_2 \leq 2N_1$. This step accounts for the possibility that the solution becomes steeper in this region. Our experience has shown that $N_2 = N_1$ is a good choice.

(e) Add the function values at all of the newly added points to $\mathcal{U}_{\text{int}}$. If the function value at any of the newly added points is not known, we interpolate the function value at that point from the points in $\mathsf{Grid}_{\text{old}}$ and their function values in $\mathcal{U}_{\text{old}}$ using $\mathcal{I}(\cdot, \mathcal{X}_{\mathsf{Grid}_{\text{old}}})$.

(f) Increment i by 1. If $i \leq M$, go to Step 2(a), otherwise move on to the next step.

Step 3. Increment j by 1. If $j < J_{\max}$, go to Step 1, otherwise move on to the next step.

Step 4. Terminate the algorithm. The final nonuniform grid is $\mathsf{Grid}_{\text{new}} = \mathsf{Grid}_{\text{int}}$, and their corresponding function values are the set $\mathcal{U}_{\text{new}} = \mathcal{U}_{\text{int}}$.

Remark 1. Although the proposed grid adaptation algorithm will work for any interpolation technique, in this paper, we use ENO interpolation to avoid any unphysical interpolation of the data.

Remark 2. For sake of brevity, in this paper, we work only with the point-value discretization of data, but the proposed encoding and decoding algorithms (or the grid adaptation algorithm for solving PDEs) will also work for discretizations based on cell-averages.

Next, we explain the proposed grid adaptation algorithm with the help of a simple example.

Example 2. Consider a dyadic grid $\mathcal{V}_4$ and the function

$$g(x) = \begin{cases} 1, & x = x_{4,k}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.23}$$

[2]We would like to remind the reader that $u \in \mathbb{R}^m$.

[3]Note that $u(y_{j,k}, t_n) \in \mathcal{U}_{\text{old}}$ for all $y_{j,k} \in Y$.

with $k = 6$, so that g denotes an impulse located at $x = x_{4,6} = 0.375$. Let $J_{\min} = 0$, $J_{\max} = 4$, $p = 1$, $\epsilon = 0.1$, $N_1 = N_2 = 1$, and consider $\mathsf{Grid}_{\text{old}} = \mathcal{V}_{J_{\max}}$. For this example, the proposed grid adaptation algorithm is illustrated in Figure 3.3.

In Figure 3.3, the solid circles show the points belonging to the intermediate grid $\mathsf{Grid}_{\text{int}}$ and those belonging to $\mathsf{Grid}_{\text{new}}$. The empty squares show the points that are being tested or have been tested for inclusion in $\mathsf{Grid}_{\text{int}}$. If the interpolative error coefficient at a point is greater than the prescribed threshold, then we show that point by a solid square. The left and the right neighbors are shown by left and right triangles, respectively. For reference, all points at that particular level are shown by empty circles. Next, we give a point-by-point illustration of the proposed algorithm for Example 2.

$\mathcal{V}_0$. Initialize $\mathsf{Grid}_{\text{int}}$ with $\mathcal{V}_{J_{\min}} = \mathcal{V}_0$.

$\mathcal{W}_0$. Check if the function value at the point $y_{0,0} \in \mathcal{W}_0$ can be interpolated from the nearest $p+1 = 2$ points in $\mathsf{Grid}_{\text{int}}$, which, in this case, are the points $x_{0,0}$ and $x_{0,1}$. Since, for this example, $g(y_{0,0})$ can be interpolated from the points in $\mathsf{Grid}_{\text{int}}$, we do not include $y_{0,0}$ in $\mathsf{Grid}_{\text{int}}$. Next, we move on to level $\mathcal{W}_1$.

$\mathcal{W}_1$. $y_{1,0}$. Since $g(y_{1,0})$ can again be interpolated from the function values at points $x_{0,0}, x_{0,1} \in \mathsf{Grid}_{\text{int}}$, we do not include $y_{1,0}$ in $\mathsf{Grid}_{\text{int}}$ and move on to the next point $y_{1,1}$.

$y_{1,1}$. For the same reason as before, we do not include this point and move on to the next level.

$\mathcal{W}_2$. $y_{2,0}$. For the same reason as before, we do not include this point.

$y_{2,1}$. Moving further to $y_{2,1}$, we find that $g(y_{2,1})$ cannot be interpolated from the neighboring two points $x_{0,0}, x_{0,1} \in \mathsf{Grid}_{\text{int}}$. Hence, we include $y_{2,1}$ in $\mathsf{Grid}_{\text{int}}$ along with points $y_{2,0}, y_{2,2} \in \mathcal{W}_2$ and $y_{3,2}, y_{3,3} \in \mathcal{W}_3$.

$y_{2,2}$. Next, we check point $y_{2,2}$. The nearest points to $y_{2,2}$ in $\mathsf{Grid}_{\text{int}}$ are $y_{3,3}$ and $x_{0,1}$. Since $g(y_{2,2})$ can be interpolated from $y_{3,3}$ and $x_{0,1}$, we move to the next point $y_{2,3}$ belonging to the level $\mathcal{W}_2$.

$y_{2,3}$. For the same reason as for point $y_{2,2}$, we do not include $y_{2,3}$.

$\mathcal{W}_3$. $y_{3,0}$. Since $y_{3,0}$ can be interpolated from the existing points in $\mathsf{Grid}_{\text{int}}$, we do not include $y_{3,0}$ in the grid.

$y_{3,1}$. For the same reason as for $y_{3,0}$, we do not include $y_{3,1}$ in the grid.

$y_{3,2}$. Subsequently, we check $y_{3,2}$. Since $g(y_{3,2})$ cannot be interpolated from the nearest two points $y_{2,0}, y_{2,1} \in \mathsf{Grid}_{\text{int}}$, we include $y_{3,2}$ (which, in any case, is already present in $\mathsf{Grid}_{\text{int}}$) along with points $y_{3,1}$ and $y_{3,3}$ (already present in $\mathsf{Grid}_{\text{int}}$) in $\mathsf{Grid}_{\text{int}}$.

$y_{3,3}$. Moving on to the next point $y_{3,3}$, we see again that $g(y_{3,3})$ cannot be interpolated from the nearest two points $y_{2,1}, y_{2,2} \in \mathsf{Grid}_{\text{int}}$. Hence, we include $y_{3,3}$ (already present in $\mathsf{Grid}_{\text{int}}$) along with $y_{3,2}$ (already present in $\mathsf{Grid}_{\text{int}}$) and $y_{3,4}$ in $\mathsf{Grid}_{\text{int}}$.

$y_{3,4}$. Since $g(y_{3,4})$ can be interpolated from the two nearest points $y_{3,3}, y_{2,2} \in \mathsf{Grid}_{\text{int}}$, we move on to the next point $y_{3,5}$.

$y_{3,5}$ The nearest two points to $y_{3,5}$ in $\mathsf{Grid}_{\text{int}}$ are $y_{2,2}$, $x_{0,1}$, and since $g(y_{3,5})$ can be interpolated from these two points, we do not include $y_{3,5}$ in $\mathsf{Grid}_{\text{int}}$.

$y_{3,6}$. For the same reason as for $y_{3,5}$ we do not add point $y_{3,6}$ to the grid.

$y_{3,7}$. For the same reason as for $y_{3,5}$ we do not add point $y_{3,7}$ to the grid.

$\mathsf{Grid}_{\text{new}}$. The final adaptive grid $\mathsf{Grid}_{\text{new}}$ is shown by the solid circles in Figure 3.3.

The adaptive grid generated using the previous algorithm depends on how we select points along the grid, that is, whether we move from left to right or from right

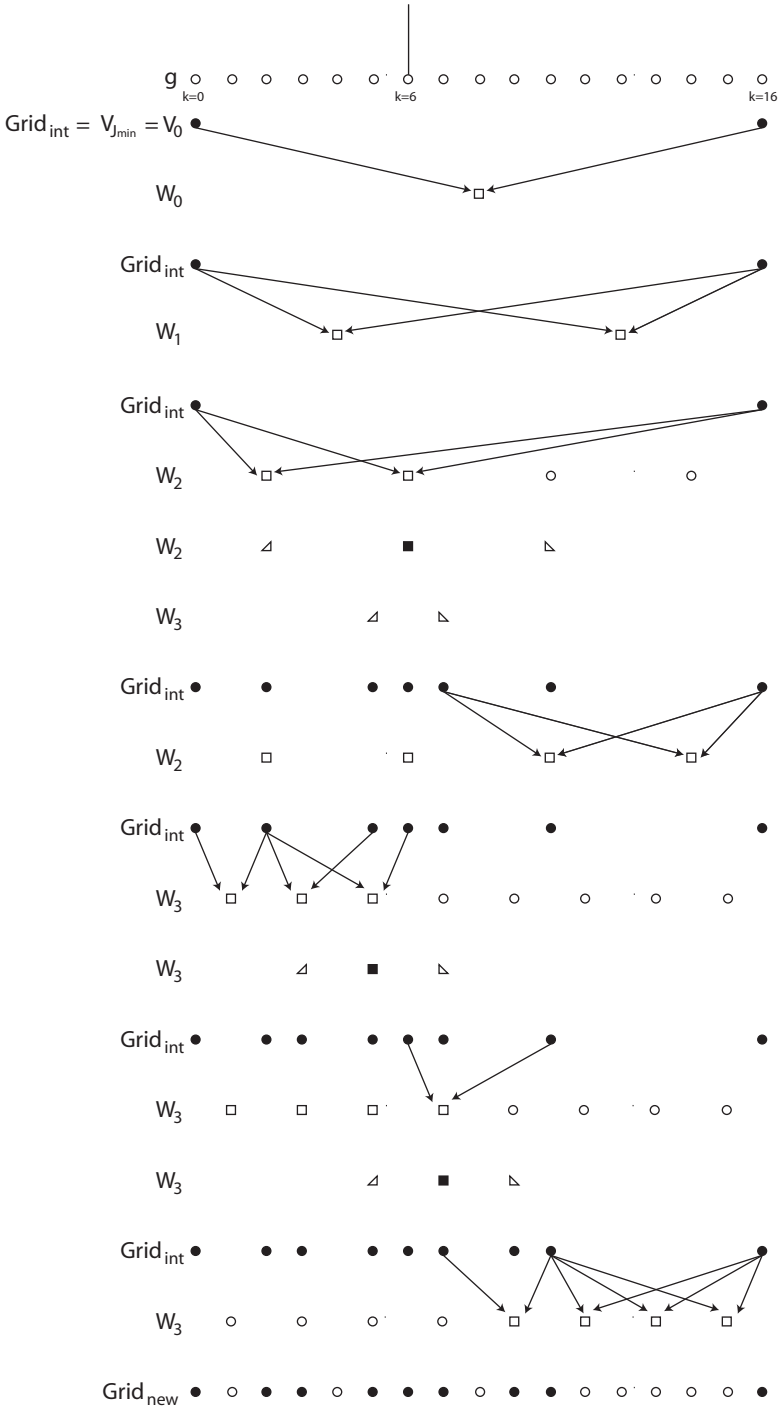


FIG. 3.3. *Demonstration of the grid adaptation algorithm using Example* 2.

to left across each level. It also depends on the location of the singularity. If the singularity is located in the middle, then it does not matter whether we move from left to right or from right to left. The result will be the same nonuniform grid. If, on the other hand, the singularity is not in the middle, then the final grid depends on the way in which we traverse across each level. To illustrate this fact, we again consider Example 2, but this time with $k = 1$. Hence, the impulse is located at $x = x_{4,1}$. If we go from left to right, then the adaptive grid consists of the points $x_{4,0},\ x_{4,1},\ x_{4,3},\ x_{4,5},\ x_{4,16}$. If we go from right to left, then the grid consists of the points $x_{4,0}, x_{4,1}, x_{4,3}, x_{4,16}$. Let $k = 15$, which implies that the impulse is located at $x = x_{4,15}$. If we go from left to right, then the grid consists of the points $x_{4,0},\ x_{4,13},\ x_{4,15},\ x_{4,16}$, and if we go from right to left, then the grid consists of the points $x_{4,0}, x_{4,11}, x_{4,13}, x_{4,15}, x_{4,16}$. Note that, in the proposed algorithm, it is not mandatory to traverse across a level only from the leftmost point or from the rightmost point. We can instead start from any point at that level, each time resulting in a different grid. This suggests that by using a suitable probability distribution function to choose the order in which the points at each particular level are selected, one may be able to further optimize the grid. We will not elaborate on this observation in this paper.

3.4. Comparison with existing multiresolution-based approaches. The proposed encoding and grid adaptation algorithms result, in general, in a fewer number of grid points when compared to Harten's multiresolution scheme [19, 20, 21], the grid adaptation algorithms of Harten [19, 20], Holmstrom [23], and Alves et al. [3]. First, we explain why this is so and then we give several examples to demonstrate this fact.

In the encoding algorithms (grid adaptation algorithms for the solution of IBVP) of existing approaches [19, 20, 23, 3], one interpolates $\{g(y_{j,k})\}_{k=0}^{2^j-1}$ only from the function values at the points belonging to $\mathcal{V}_j$ for $j = J_{\min} \dots J_{\max} - 1$, and only then, one adds to the adaptive grid, the points $y_{j,k}$ (along with the points $\{y_{j,k+\ell}\}_{\ell=-N_1}^{N_1}$ and $\{y_{j+1,2k+\ell}\}_{\ell=-N_2+1}^{N_2}$) for all the pairs (j,k) such that $d_{j,k} > \epsilon$. In the proposed method, we continuously keep on updating the adaptive grid instead. If the interpolative error coefficient at $y_{j,k}$, where $0 \le k \le 2^j - 1$ and $J_{\min} \le j \le J_{\max} - 1$, is greater than the prescribed threshold, we add $y_{j,k}$ to the adaptive grid, while at the same time we also add to the adaptive grid the neighboring points at the same level $\{y_{j,k+\ell}\}_{\ell=-N_1}^{N_1}$, as well as the neighboring points at the next level $\{y_{j+1,2k+\ell}\}_{\ell=-N_2+1}^{N_2}$. We use the newly added point(s) also for interpolating the remaining points at level $\mathcal{W}_j$ and the levels below it. In other words, in the proposed approach, the values $\{g(y_{j,k})\}_{k=0}^{2^j-1}$ are interpolated from the function values at the points in $\mathcal{V}_j \oplus \mathcal{W}_j$ for $j = J_{\min}, \dots, J_{\max}-1$ (in $\mathcal{V}_j \oplus \mathcal{W}_j \oplus \mathcal{W}_{j+1}$ for $J_{\min} \le j \le J_{\max} - 2$ and in $\mathcal{V}_j \oplus \mathcal{W}_j$ for $j = J_{\max} - 1$). Hence, by making use of the extra information from levels $\mathcal{W}_j$ (and $\mathcal{W}_{j+1}$), which, in any case, will be added to the adaptive grid, we are able to reduce the number of grid points in the final grid.

We illustrate this fact with the help of several examples.

Example 3. We again consider the functions g_1 and g_2 given by (3.18) and (3.19), respectively, and a grid with $J_{\min} = 2$ and $J_{\max} = 10$. In order to compare the proposed encoding algorithm (using ENO interpolation) with Harten's encoding algorithm (using ENO interpolation) and Harten's encoding algorithm (using central interpolation), we set $N_1 = N_2 = 0$ in the grid adaptation algorithm of section 3.3. One should note that, for $N_1 = N_2 = 0$, the grid adaptation algorithm given in section 3.3 is simply the decoding algorithm described in section 3.2. The number

TABLE 3.2
Example 3. *Comparison of the proposed decoding approach with Harten's approach.*

	ϵ	$N_{\mathrm{g_p}}$	$N_{\mathrm{g_{Heno}}}$	$N_{\mathrm{g_{Hc}}}$	$N_{\mathrm{g_p}}/N_{\mathrm{g_{Heno}}}$	$N_{\mathrm{g_p}}/N_{\mathrm{g_{Hc}}}$
g_1	10^{-3}	25	29	53	0.86	0.47
g_2	10^{-3}	52	58	108	0.90	0.48
g_2	10^{-7}	63	73	119	0.86	0.53

of grid points used by the proposed algorithm $N_{\mathrm{g_p}}$, Harten's algorithm (using ENO interpolation) $N_{\mathrm{g_{Heno}}}$, and Harten's algorithm (using central interpolation) $N_{\mathrm{g_{Hc}}}$ for different thresholds using interpolating polynomial of degree $p = 3$ are summarized in Table 3.2. For both functions g_1 and g_2, we found that the proposed algorithm results in up to 14% fewer number of points in the compressed data compared to Harten's approach (using ENO interpolation) and up to 53% fewer points compared to Harten's approach (using central interpolation). It is reminded that Harten's encoding algorithm (using central interpolation) forms the basis of the multiresolution scheme of Holmstrom [23] and Alves et al. [3] for solving PDEs.

In Harten's approach, the solution at each time step is represented on the finest grid, and one encodes and decodes the solution at each time step in order to calculate the interpolative errors. In other words, the interpolative errors are computed at all points of the fine grid at each mesh refinement step. Holmstrom [23], on the other hand, calculates the interpolative error coefficients only at the points that are in the adaptive grid; if a function value is needed that does not exist in the present grid, the function value is interpolated recursively from coarser scales. In the algorithm of Alves et al. [3], one also adds to the grid the points that were used to predict the function values at all the previously added points in order to compute the interpolative error during the next mesh adaptation. Therefore, in the approach of Alves et al. when a point $y_{j,k}$ ($0 \le k \le 2^j - 1$ and $J_{\min} \le j \le J_{\max} - 1$) is added to the grid, one also include its parents, which were used to predict the function value at that point. The parents are not needed for approximating the given function to the prescribed accuracy but are included just for calculating the interpolative error coefficient at the point $y_{j,k}$ during the next mesh adaptation. In the proposed algorithm, on the other hand, whenever a point is being checked for inclusion in the adaptive grid, we predict the function value at that point only from the points already existing in the adaptive grid. Hence, if that point is inserted in the grid, we do not need to add any extra points (i.e., its parents). This also alleviates the task of keeping track of the parents from the rest of the points, as in the approach of Alves et al., and the task of recursively calculating the function values from the coarser resolution levels as is done in the approach of Holmstrom. Next, we give several examples to compare the proposed grid adaptation approach with the algorithm of Alves et al. [3] for solving evolution PDEs.

Example 4. First we consider a very simple example. For this example, we consider a dyadic grid $\mathcal{V}_4$ and the function

$$g(x) = \begin{cases} 1, & x = x_{4,k}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.24}$$

with an impulse located at $x = k/2^4$, where $0 \le k \le 16$. Let $J_{\min} = 0$, $J_{\max} = 4$, $p = 1$, $\epsilon = 0.1$, and $N_1 = N_2 = 1$. Table 3.3 shows the number of grid points used by the proposed grid adaptation algorithm $N_{\mathrm{g_p}}$ and the number of points $N_{\mathrm{g_A}}$ used

TABLE 3.3
Example 4. *Comparison of the proposed algorithm with the algorithm of Alves et al.*

k	$N_{\mathrm{g_P}}$	$N_{\mathrm{g_A}}$	$N_{\mathrm{g_P}}/N_{\mathrm{g_A}}$	k	$N_{\mathrm{g_P}}$	$N_{\mathrm{g_A}}$	$N_{\mathrm{g_P}}/N_{\mathrm{g_A}}$
0	9	9	1	9	6	9	0.67
1	5	6	0.83	10	9	12	0.75
2	7	9	0.78	11	6	9	0.67
3	6	8	0.75	12	11	12	0.92
4	11	12	0.92	13	5	7	0.71
5	6	9	0.67	14	7	9	0.78
6	9	12	0.75	15	4	6	0.67
7	6	9	0.67	16	9	9	1
8	13	13	1				

TABLE 3.4
Example 5. *Comparison of the proposed algorithm with the algorithm of Alves et al.*

	ϵ	$N_{\mathrm{g_P}}$	$N_{\mathrm{g_A}}$	$N_{\mathrm{g_P}}/N_{\mathrm{g_A}}$
g_1	10^{-3}	53	93	0.57
g_2	10^{-3}	108	185	0.58

by the grid adaptation scheme of Alves et al. [3] for $k = 0, \ldots, 16$. We found that when the impulse is located at either the left boundary ($k = 0$) or the right boundary ($k = 16$) or in the middle of the domain ($k = 8$), both the traditional approach and the proposed approach result in the same grid. For all other cases, the grids generated are different. Moreover, we see that the proposed algorithm results in a fewer number of grid points. For this example, the proposed algorithm outperforms the algorithm of Alves et al. [3] by up to 33%.

Example 5. We again consider the functions g_1 and g_2 given by (3.18) and (3.19), respectively, and a grid with $J_{\min} = 2$ and $J_{\max} = 10$. This time we set $N_1 = N_2 = 1$ in the proposed grid adaptation algorithm. Table 3.4 gives the number of points used by the proposed grid adaptation algorithm $N_{\mathrm{g_P}}$ and the number of points $N_{\mathrm{g_A}}$ used by the grid adaptation scheme of Alves et al. [3]. For this example, we observe that the proposed grid adaptation algorithm outperforms the algorithm of Alves et al. [3] by up to 43%.

We are now ready to present the algorithm for solving the (IBVP) on an adaptive, nonuniform grid.

4. Numerical solution of the IBVP for evolution equations. The numerical scheme for discretizing (IBVP) depends on $f(u_{xx}, u_x, u, x)$. The proposed grid adaptation algorithm will work for many numerically stable discretization schemes for (IBVP). We use different schemes for the numerical examples discussed in this paper, depending on the problem. Hence, in the next section, we describe only the techniques we use for calculating the spatial derivatives u_x and u_{xx} on the nonuniform grid, and we state the numerical schemes in the examples themselves.

4.1. Calculation of spatial derivatives. To calculate the derivative u_x on the adaptive nonuniform grid $\mathsf{Grid}_{\mathrm{new}}$, we use the weighted ENO (WENO) scheme [27, 28, 33, 39] on nonuniform grids. To this end, let the nonuniform grid be given as in (3.5). Now define

$$D^+ u^n_{j_i,k_i} = \frac{u^n_{j_{i+1},k_{i+1}} - u^n_{j_i,k_i}}{x_{j_{i+1},k_{i+1}} - x_{j_i,k_i}}, \qquad D^- u^n_{j_i,k_i} = \frac{u^n_{j_i,k_i} - u^n_{j_{i-1},k_{i-1}}}{x_{j_i,k_i} - x_{j_{i-1},k_{i-1}}}. \tag{4.1}$$

A third-order ENO approximation [22, 42, 43] to $(u_x^{\pm})^n_{j_i,k_i} = u_x^{\pm}(x_{j_i,k_i}, t_n)$ is given by one of the following expressions:

$$((u_x^{\pm})^n_{j_i,k_i})_1 = \frac{v_1}{3} - \frac{7v_2}{6} + \frac{11v_3}{6}, \tag{4.2a}$$

or

$$((u_x^{\pm})^n_{j_i,k_i})_2 = -\frac{v_2}{6} + \frac{5v_3}{6} + \frac{v_4}{3}, \tag{4.2b}$$

or

$$((u_x^{\pm})^n_{j_i,k_i})_3 = \frac{v_3}{3} + \frac{5v_4}{6} - \frac{v_5}{6}, \tag{4.2c}$$

where, for calculating $(u_x^-)^n_{j_i,k_i}$, we use $v_1 = D^- u^n_{j_{i-2},k_{i-2}}$, $v_2 = D^- u^n_{j_{i-1},k_{i-1}}$, $v_3 = D^- u^n_{j_i,k_i}$, $v_4 = D^- u^n_{j_{i+1},k_{i+1}}$, $v_5 = D^- u^n_{j_{i+2},k_{i+2}}$, and, for calculating $(u_x^+)^n_{j_i,k_i}$, we use $v_1 = D^+ u^n_{j_{i+2},k_{i+2}}$, $v_2 = D^+ u^n_{j_{i+1},k_{i+1}}$, $v_3 = D^+ u^n_{j_i,k_i}$, $v_4 = D^+ u^n_{j_{i-1},k_{i-1}}$, $v_5 = D^+ u^n_{j_{i-2},k_{i-2}}$. The basic idea behind a third-order ENO scheme is to choose either $((u_x^{\pm})^n_{j_i,k_i})_1$ or $((u_x^{\pm})^n_{j_i,k_i})_2$ or $((u_x^{\pm})^n_{j_i,k_i})_3$ for approximating $(u_x^{\pm})^n_{j_i,k_i}$ by choosing the smoothest possible polynomial interpolation of u.

It is reminded that a WENO approximation of $(u_x^{\pm})^n_{j_i,k_i}$ is a convex combination of the approximations in (4.2a), (4.2b), and (4.2c), that is,

$$(u_x^{\pm})^n_{j_i,k_i} = \sum_{\ell=1}^{3} \omega_\ell ((u_x^{\pm})^n_{j_i,k_i})_\ell, \tag{4.3}$$

where $0 \le \omega_\ell \le 1$ for $\ell = 1, 2, 3$ and $\omega_1 + \omega_2 + \omega_3 = 1$. The weights for fifth-order accuracy are given by [27, 39]

$$\omega_\ell = \frac{\alpha_\ell}{\alpha_1 + \alpha_2 + \alpha_3}, \qquad \ell = 1, 2, 3, \tag{4.4}$$

where

$$\alpha_\ell = \frac{\bar{\alpha}_\ell}{(S_\ell + \delta)^2}, \qquad \ell = 1, 2, 3, \tag{4.5}$$

$$S_1 = \frac{13}{12}(v_1 - 2v_2 + v_3)^2 + \frac{1}{4}(v_1 - 4v_2 + 3v_3)^2, \tag{4.6}$$

$$S_2 = \frac{13}{12}(v_2 - 2v_3 + v_4)^2 + \frac{1}{4}(v_2 - v_4)^2, \tag{4.7}$$

$$S_3 = \frac{13}{12}(v_3 - 2v_4 + v_5)^2 + \frac{1}{4}(3v_3 - 4v_4 + v_5)^2, \tag{4.8}$$

and

$$\bar{\alpha}_1 = 0.1, \quad \bar{\alpha}_2 = 0.6, \quad \bar{\alpha}_3 = 0.3. \tag{4.9}$$

In (4.5), δ is used to prevent the denominator from becoming zero. In our computations, we have used $\delta = 10^{-6}$.

For the sake of brevity, we denote the cell walls by

$$x_{j_{i-1/2},k_{i-1/2}} = \frac{x_{j_{i-1},k_{i-1}} + x_{j_i,k_i}}{2}, \quad x_{j_{i+1/2},k_{i+1/2}} = \frac{x_{j_i,k_i} + x_{j_{i+1},k_{i+1}}}{2}. \tag{4.10}$$

In order to calculate $(u_{xx})^n_{j_i,k_i} = u_{xx}(x_{j_i,k_i}, t_n)$ on a nonuniform grid (3.5), we use the centered second difference scheme [46]

$$(u_{xx})^n_{j_i,k_i} = \frac{\left(\frac{u^n_{j_{i+1},k_{i+1}} - u^n_{j_i,k_i}}{x_{j_{i+1},k_{i+1}} - x_{j_i,k_i}} - \frac{u^n_{j_i,k_i} - u^n_{j_{i-1},k_{i-1}}}{x_{j_i,k_i} - x_{j_{i-1},k_{i-1}}}\right)}{x_{j_{i+1/2},k_{i+1/2}} - x_{j_{i-1/2},k_{i-1/2}}}. \tag{4.11}$$

We are now ready to give the algorithm for solving the (IBVP) for evolution equation (2.1).

4.2. Solution of the (IBVP) for evolution PDEs. Based on the problem, the desired accuracy, and the computational hardware, we choose the minimum resolution level $J_{\min}$, the maximum resolution level $J_{\max}$, the threshold ϵ, the order of the interpolating polynomial p, and the parameters N_1, N_2 required for the grid adaptation algorithm described in section 3.3. The final time t_f is assumed to be given.

To solve (IBVP) on an adaptive grid, we first initialize $\mathsf{Grid}_{\text{old}} = \mathcal{V}_{J_{\max}}$, $\mathcal{U}_{\text{old}} = \{g(x_{J_{\max},k})\}_{k=0}^{2^{J_{\max}}}$ and set $t = 0$, $n = 0$.[4] Then the algorithm proceeds as follows.

Step 1. Given $\mathsf{Grid}_{\text{old}}$, $\mathcal{U}_{\text{old}}$, find the new grid $\mathsf{Grid}_{\text{new}}$ and the function values at all the points in $\mathsf{Grid}_{\text{new}}$, $\mathcal{U}_{\text{new}} = \{u^n_{j,k} : x_{j,k} \in \mathsf{Grid}_{\text{new}}\}$ using the grid adaptation algorithm given in section 3.3. The new grid $\mathsf{Grid}_{\text{new}}$ is the grid on which we will propagate the solution from time t to time $t + \Delta t_{\text{adapt}}$, where $\Delta t_{\text{adapt}} = k\Delta t_n$ ($k \in \mathbb{N}$) is the time after which the grid should be adapted again. For hyperbolic equations,

$$k = \left\lfloor \frac{N_1 \cdot \Delta x_{\min}}{\Delta t_n \cdot \text{wave speed}} \right\rfloor, \tag{4.12}$$

where $\Delta x_{\min} = \min_{\mathsf{Grid}_{\text{new}}}(x_{j_{i+1},k_{i+1}} - x_{j_i,k_i})$. The reader is referred to [46, 39] for details on computing the wave speed. For all other evolution equations, use $k = 1$.

Step 2. Compute the solution at time $t = t_{n+1}$ at all the points belonging to $\mathsf{Grid}_{\text{new}}$, $\mathcal{U}_{\text{new}} = \{u^{n+1}_{j,k} : x_{j,k} \in \mathsf{Grid}_{\text{new}}\}$ using any numerically stable scheme and increment n by 1. Keep on repeating this step while $t < \Delta t_{\text{adapt}}$. If $t \geq t_f$, terminate the algorithm.

Step 3. Reassign the sets: $\mathsf{Grid}_{\text{old}} \leftarrow \mathsf{Grid}_{\text{new}}$, $\mathcal{U}_{\text{old}} \leftarrow \mathcal{U}_{\text{new}}$. It should be noted that we do not interpolate the function values at the finest level during the mesh refinement process. In the proposed mesh refinement algorithm, we check only the retained points in $\mathsf{Grid}_{\text{old}}$ to further add and remove points in the grid. The interpolative error coefficients are computed only at the points $y_{j,k} \in \mathsf{Grid}_{\text{old}}$, and the solution $\mathcal{U}_{\text{old}}$ for all $y_{j,k} \in \mathsf{Grid}_{\text{old}}$ is known from the previous step.

Step 4. Calculate the new time at which the next mesh adaptation should take place $t_{\text{adapt}} = t + \Delta t_{\text{adapt}}$. Go to Step 1.

Remark 3. As pointed out earlier, Δt_n is computed based on the CFL condition [46] for hyperbolic equations and the von Neumann condition [46] for all other evolution equations. For both the CFL condition and the von Neumann condition, Δt_n depends on $\Delta x_{\min}$. Hence, in the proposed algorithm, Δt_n changes adaptively depending on $\Delta x_{\min}$, which also changes adaptively.

[4] In case of hardware limitations, we suggest using $\mathsf{Grid}_{\text{old}} = \mathcal{V}_{J_{\text{int}}}$, $\mathcal{U}_{\text{old}} = \{g(x_{J_{\text{int}},k})\}_{k=0}^{2^{J_{\text{int}}}}$, where $J_{\min} < J_{\text{int}} < J_{\max}$ is chosen based on the specific computational hardware used.

5. Numerical examples. In this section, we present several examples to demonstrate the stability and robustness of our algorithm. These examples also illustrate the ability of the algorithm to automatically capture and follow any existing or self-sharpening features of the solution that develop in time.

Example 6. Consider a nonlinear conservation law

$$u_t + (F(u))_x = 0. \tag{5.1}$$

To ensure that shocks and other steep gradients move at the right speed, (5.1) should be written in a discrete conservation form [31, 39]:

$$u^{n+1}_{j_i,k_i} = u^n_{j_i,k_i} - \Delta t_{n+1} \frac{\mathcal{F}^n_{j_{i+1/2},k_{i+1/2}} - \mathcal{F}^n_{j_{i-1/2},k_{i-1/2}}}{x_{j_{i+1/2},k_{i+1/2}} - x_{j_{i-1/2},k_{i-1/2}}}, \tag{5.2}$$

where $\mathcal{F}^n_{j_{i\pm1/2},k_{i\pm1/2}} = \mathcal{F}(x_{j_{i\pm1/2},k_{i\pm1/2}}, t_n)$. For a specific example, we consider the inviscid Burgers' equation

$$u_t + \left(\frac{1}{2}u^2\right)_x = 0. \tag{5.3}$$

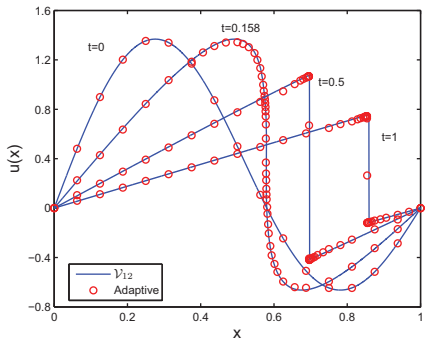
We use the same smooth initial condition and the Dirichlet boundary condition as in [3], that is,

$$g(x) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x), \quad u(0,t) = u(1,t) = 0, \tag{5.4}$$

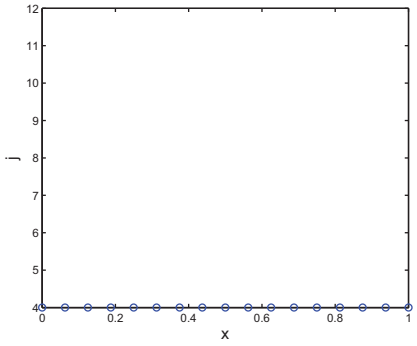
to check the ability of the proposed algorithm to capture the shock. The solution is a wave that develops a very steep gradient and subsequently moves towards $x = 1$. Because of the zero boundary values, the wave amplitude diminishes with increasing time.

For solving (5.3)–(5.4), we use (5.2) along with the ENO–Roe scheme proposed by Shu and Osher [43] on a nonuniform grid for calculating the numerical flux functions $\mathcal{F}^n_{j_{i\pm1/2},k_{i\pm1/2}}$. For temporal integration, we use a third-order total variation diminishing (TVD) Runge–Kutta (RK) scheme [42]. The numerical solution at times $t = 0$ seconds (s), $t = 0.158$ s, $t = 0.5$ s, and $t = 1$ s using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ are shown in Figure 5.1a. The other parameters used in the grid adaptation procedure are $p = 3$, $\epsilon = 0.01$, and $N_1 = N_2 = 1$. Figure 5.1 also shows the grid point distribution in the adaptive mesh at times $t = 0$ s, $t = 0.1$ s, $t = 0.158$ s, and $t = 1$ s. We see that, as the shock continues to develop, the algorithm adds points at the finer levels of resolution in the region where the shock is developing and removes points from the regions where the solution is getting smoother. Similar conclusions can be drawn by looking at the time evolution of the number of grid points (Figure 5.1f). We observe that the number of grid points increases as the shock continues to develop, and once the solution is smooth everywhere except for the region of the shock, the number of grid points remains pretty much steady, with the number of grid points oscillating about a mean value of 43 points. This shows that the proposed strategy uses only the grid points that are actually necessary to attain a given precision, and the algorithm is able to add and remove points when and where is needed.

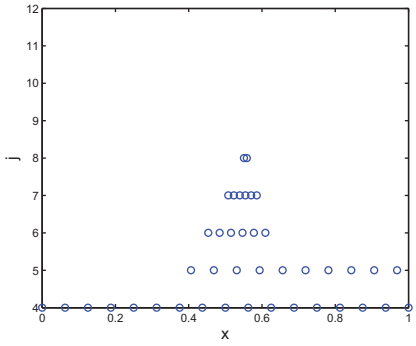
A comparison of CPU times for the uniform and adaptive grids, along with the L_1 error ($E_1(u)$) between the solution of the proposed multiresolution algorithm and the fine grid solution evaluated at grid $\mathcal{V}_{J_{\max}}$, and the number of grid points used by



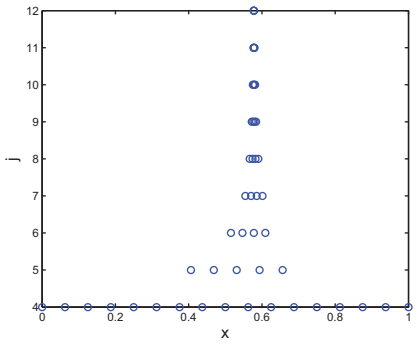
(a) *Solution* $u(x,t)$.



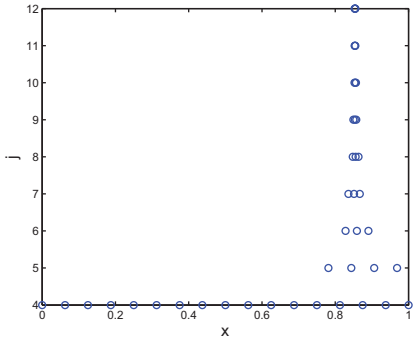
(b) *Grid point distribution at* $t = 0\,\mathrm{s}$.



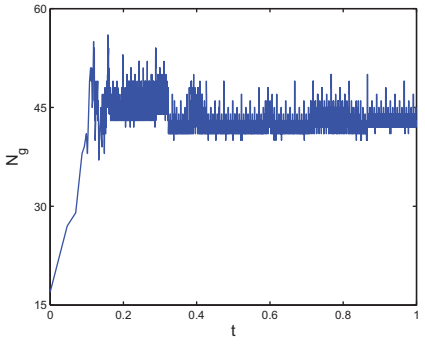
(c) *Grid point distribution at* $t = 0.1\,\mathrm{s}$.



(d) *Grid point distribution at* $t = 0.158\,\mathrm{s}$.



(e) *Grid point distribution at* $t = 1\,\mathrm{s}$.



(f) *Time evolution of the number of grid points.*

FIG. 5.1. *Example* 6. *Parameters used in the simulation are* $J_{\min} = 4$, $J_{\max} = 12$, $p = 1, \epsilon = 0.01$, *and* $N_1 = N_2 = 1$.

the proposed algorithm at the final time step for different $J_{\max} = 8, 9, 10, 11, 12$ are summarized in Table 5.1. We observe a major speedup in the computational time compared to the uniform mesh, and the speedup factors increase at an approximate rate of two. The proposed approach results in speedup factors that are higher than those reported in [3]. For scale $J_{\max} = 12$, the speedup factor using the proposed approach is 63.7, which is about 27% higher than the one reported in [3]. It is reminded that we chose $N_1 = 1$ and Alves et al. [3] chose $N_1 = 2$, which implies that,

TABLE 5.1
Example 6. L_1 *error and computational times for uniform versus adaptive mesh.*

	Uniform mesh		Adaptive mesh			
$J_{\max}$	N_{g} in $\mathcal{V}_{J_{\max}}$	t_{cpu} (s)	N_{g} at t_f in $\mathsf{Grid}_{\mathrm{new}}$	$E_1(u)$	t_{cpu} (s)	Speedup
8	$2^8+1=257$	2.7106	31	7.1991×10^{-3}	0.5835	4.6454
9	$2^9+1=513$	9.3851	34	7.1717×10^{-3}	1.2737	7.3684
10	$2^{10}+1=1025$	36.6631	37	7.7397×10^{-3}	2.6622	13.7717
11	$2^{11}+1=2049$	223.8606	40	7.7220×10^{-3}	6.0399	37.0636
12	$2^{12}+1=4097$	804.9415	43	7.8012×10^{-3}	12.6301	63.7320

TABLE 5.2
Example 6. L_1 *errors at different times for* $J_{\max}=12$.

t	N_{g} in $\mathsf{Grid}_{\mathrm{new}}$	C (%)	$E_1(u)$
0.158	49	98.80	8.2093×10^{-3}
0.5	42	98.97	9.3552×10^{-3}
1	43	98.95	7.8012×10^{-3}

in our case, mesh refinement was performed twice as many times as was performed in [3] for the same problem, and, even then, the speedup factor is 27% higher than the one reported in [3]. The L_1 errors ($E_1(u)$) along with the number of grid points used by the proposed algorithm at times $t=0.158\,\mathrm{s}$, $t=0.5\,\mathrm{s}$, and $t=1\,\mathrm{s}$ for $J_{\max}=12$ have been summarized in Table 5.2.

In the next two examples, we consider HJ equations, that is, evolution equations as in (2.1), where

$$f(u_{xx},u_x,u,x)=f(u_x). \tag{5.5}$$

For discretizing $f(u_x)$, we use the Lax–Friedrich's (LF) scheme [16, 39, 40, 43]

$$f(u_x)=\hat{f}^{\mathrm{LF}}(u_x^-,u_x^+)=f\left(\frac{u_x^++u_x^-}{2}\right)-\frac{1}{2}\alpha^x(u_x^+-u_x^-), \tag{5.6}$$

where, $\alpha^x=\max_{u_x\in I^x}|f_1(u_x)|$, f_1 is the partial derivative of f with respect to u_x, $I^x=[u_x^{\min},u_x^{\max}]$, and the minimum and the maximum values of u_x are identified by considering all the values of u_x^- and u_x^+ on the nonuniform grid.

Example 7. First, we consider the HJ equation with convex $f(u_x)$, taken from [40]:

$$u_t+\frac{(u_x+1)^2}{2}=0, \tag{5.7}$$

with the initial condition and the periodic boundary condition as in [40], that is,

$$g(x)=-\cos\pi x,\quad u(-1,t)=u(1,t),\quad -1\le x<1. \tag{5.8}$$

For solving the problem using the proposed algorithm, we first convert the above mentioned problem from $x\in[-1,1]$ to $\hat{x}\in[0,1]$ by using a simple change of variables $x=2\hat{x}-1$. With a slight abuse of notation, we denote $\hat{x}$ by x, and hence the problem(5.7)–(5.8) transforms to

$$u_t+\frac{(\frac{1}{2}u_x+1)^2}{2}=0, \tag{5.9}$$

with the initial condition and the periodic boundary condition

$$g(x) = -\cos\pi(2x-1), \quad u(0,t) = u(1,t). \tag{5.10}$$

The derivatives u_x^+, u_x^- in the LF discretization are approximated using a WENO scheme, and the temporal integration is performed using a third-order TVD RK scheme. The numerical solution at times $t = 0\,\mathrm{s}$, $t = 1.5/\pi^2\,\mathrm{s}$, $t = 3.5/\pi^2\,\mathrm{s}$, $t = 7/\pi^2\,\mathrm{s}$, $t = 10/\pi^2\,\mathrm{s}$, and $t = 14/\pi^2\,\mathrm{s}$ using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ are shown in Figure 5.2a. The other parameters used in the grid adaptation algorithm are $p = 3$, $\epsilon = 0.001$, and $N_1 = N_2 = 2$. Figure 5.2 also shows the grid point distribution in the adaptive mesh at times $t = 0\,\mathrm{s}$, $t = 3.5/\pi^2\,\mathrm{s}$, $t = 7/\pi^2\,\mathrm{s}$, and $t = 14/\pi^2\,\mathrm{s}$. We see that as the kink continues to develop, the algorithm adds points at the finer levels of resolution in the region where the kink is developing and removes points from the regions where the solution is getting smoother and smoother. As HJ equation (5.9) continues to evolve further in time, the discontinuity in the first derivative of the solution is smoothing out, and, as a result, the algorithm starts removing points from the finer levels of resolution. This, again, demonstrates that the proposed strategy uses only the grid points that are actually necessary to attain a given precision, and the algorithm is able to add and remove points when and where needed. The L_1 errors ($E_1(u)$) along with the number of grid points used by the proposed algorithm at times $t = 1.5/\pi^2\,\mathrm{s}$, $t = 3.5/\pi^2\,\mathrm{s}$, $t = 7/\pi^2\,\mathrm{s}$, $t = 10/\pi^2\,\mathrm{s}$, and $t = 14/\pi^2\,\mathrm{s}$ for $J_{\max} = 12$ have been summarized in Table 5.3.

Example 8. Next, we consider the HJ equation with nonconvex $f(u_x)$,

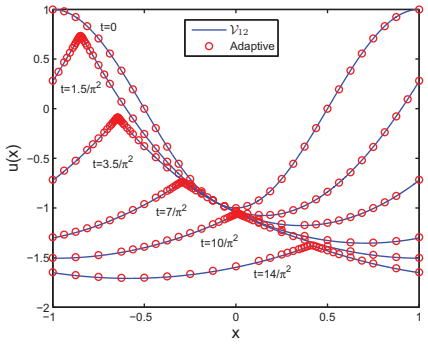
$$u_t - \cos(\alpha u_x + 1) = 0, \tag{5.11}$$

with

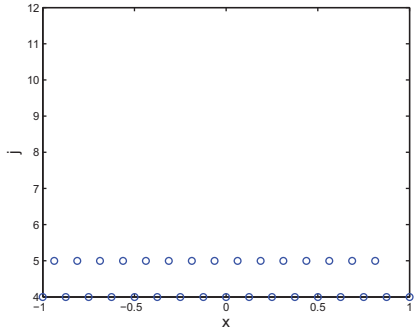
$$g(x) = -\cos\pi(2x-1), \quad u(0,t) = u(1,t), \tag{5.12}$$

where α is a constant. We again use an LF scheme (5.6) for solving the IBVP (5.11)–(5.12). The derivatives u_x^+, u_x^- in the LF discretization are approximated using a WENO scheme, and the temporal integration is performed using a third-order TVD RK scheme.

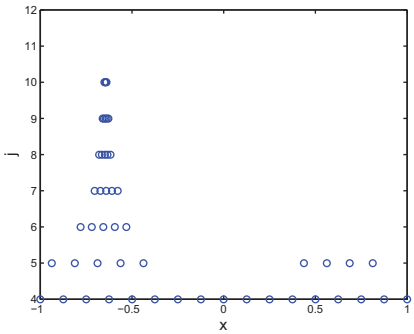
The choice of $\alpha = 0.5$ results in the commonly used test problem for one-dimensional (1-D) HJ equations given in [40]. In order to make the problem more interesting and challenging, in this work, we consider two more choices for α, namely, $\alpha = 1$ and $\alpha = 1.5$. The choices $\alpha = 1$ and $\alpha = 1.5$ result in more kinks in the solution at time $t = 1.5/\pi^2\,\mathrm{s}$. The numerical solutions for all the cases at time $t = 1.5/\pi^2\,\mathrm{s}$ using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ along with the corresponding grid point distributions are shown in Figure 5.3. The other parameters used in the grid adaptation algorithm are $p = 3$, $\epsilon = 0.001$, and $N_1 = N_2 = 2$. The solutions at $t = 1.5/\pi^2\,\mathrm{s}$ for $\alpha = 0.5$, 0.1, and 1.5 have two, four, and six kinks, respectively. We once again observe that the proposed algorithm is able to capture all the kinks in the solutions accurately and efficiently by adding points at the finer resolution levels in the region of kinks, while resolving the smoother regions using only the points at the coarse resolution levels. The L_1 errors ($E_1(u)$) along with the number of grid points used by the proposed algorithm at time $t = 1.5/\pi^2\,\mathrm{s}$ for $\alpha = 0.5,\ 1,\ 1.5$, and $J_{\max} = 12$ have been summarized in Table 5.4.



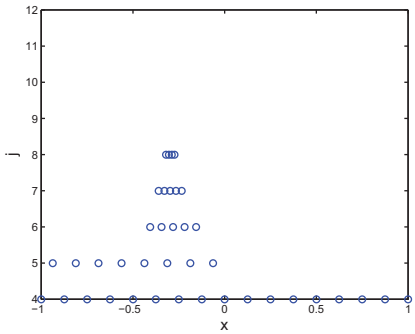
(a) *Solution* $u(x,t)$.



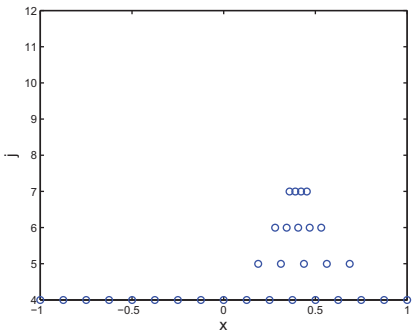
(b) *Grid point distribution at* $t = 0\,\mathrm{s}$.



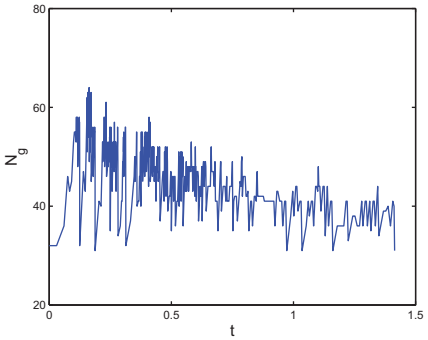
(c) *Grid point distribution at* $t = 3.5/\pi^2\,\mathrm{s}$.



(d) *Grid point distribution at* $t = 7/\pi^2\,\mathrm{s}$.



(e) *Grid point distribution at* $t = 14/\pi^2\,\mathrm{s}$.

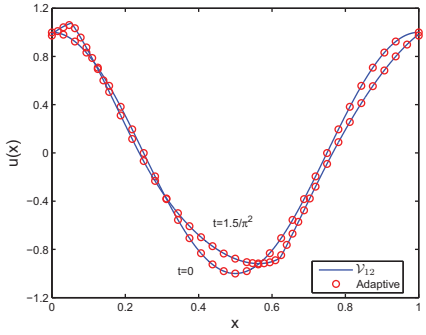


(f) *Time evolution of the grid points.*

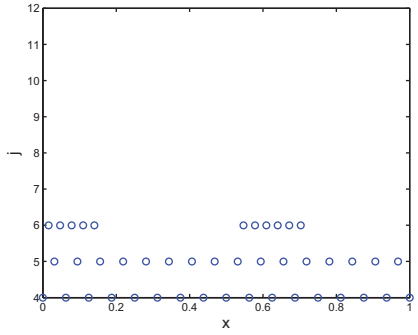
FIG. 5.2. *Example 7. Parameters used in the simulation are* $J_{\min} = 4$, $J_{\max} = 12$, $p = 3, \epsilon = 0.001$, *and* $N_1 = N_2 = 2$.

TABLE 5.3
Example 7. L_1 *errors at different times for* $J_{\max} = 12$.

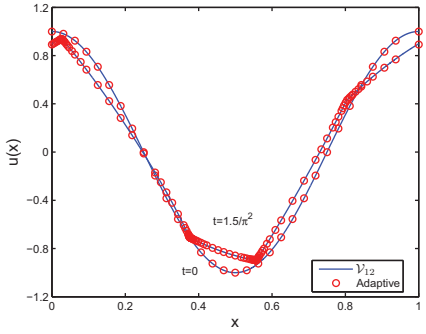
t	N_g in $\mathsf{Grid}_{\mathrm{new}}$	C (%)	$E_1(u)$
$1.5/\pi^2$	52	98.73	1.7603×10^{-3}
$3.5/\pi^2$	50	98.78	4.2828×10^{-3}
$7/\pi^2$	39	99.05	5.4194×10^{-3}
$10/\pi^2$	44	98.93	6.0113×10^{-3}
$14/\pi^2$	31	99.24	6.5118×10^{-3}



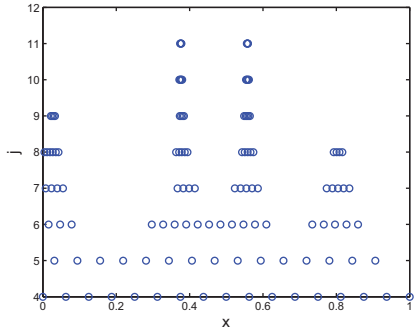
(a) *Solution* $u(x, 1.5/\pi^2)$ *for* $\alpha = 0.5$.



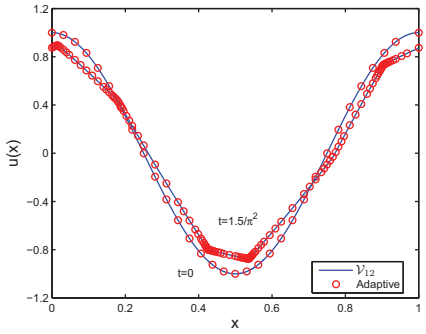
(b) *Grid point distribution at* $t = 1.5/\pi^2$ s *for* $\alpha = 0.5$.



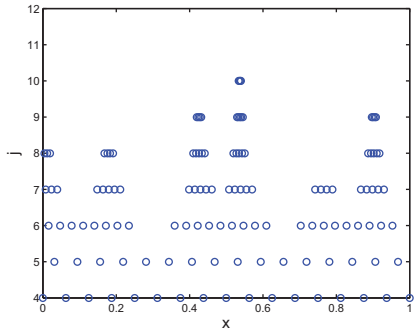
(c) *Solution* $u(x, 1.5/\pi^2)$ *for* $\alpha = 1$.



(d) *Grid point distribution at* $t = 1.5/\pi^2$ s *for* $\alpha = 1$.



(e) *Solution* $u(x, 1.5/\pi^2)$ *for* $\alpha = 1.5$.



(f) *Grid point distribution at* $t = 1.5/\pi^2$ s *for* $\alpha = 1.5$.

FIG. 5.3. *Example* 8. *Parameters used in the simulation are* $J_{\min} = 4$, $J_{\max} = 12$, $p = 3, \epsilon = 0.001$, *and* $N_1 = N_2 = 2$.

TABLE 5.4
Example 8. L_1 *errors at different times for* $J_{\max} = 12$.

α	N_g in $\mathsf{Grid}_{\text{new}}$	C (%)	$E_1(u)$
0.5	44	98.93	1.1259×10^{-3}
1	120	97.07	8.8733×10^{-4}
1.5	125	96.95	5.6106×10^{-4}

Example 9. Consider the scalar reaction-diffusion problem that appears in combustion problems [25, 41]:

$$u_t - u_{xx} - \frac{Re^{\delta}}{a\delta}(1 + a - u)e^{-\delta/u} = 0, \tag{5.13}$$

$$u_x(0,t) = 0, \quad u(1,t) = 1, \quad u(x,0) = 1. \tag{5.14}$$

The solution u represents the temperature of a reactant in a chemical system, a is the heat release, δ is the activation energy, and R is the reaction rate. For small times, the temperature gradually increases from unity with a "hot spot" forming at $x = 0$. After some finite time, ignition occurs, and the temperature at $x = 0$ jumps rapidly from near unity to near $1+a$. A flame front then forms and propagates towards $x = 1$ with a speed proportional to $e^{a\delta}/2(1 + a)$. In real problems, a is close to unity and δ is large, thus the flame front moves exponentially fast after the ignition. We use the same problem parameters as in [25, 41], namely, $a = 1$, $R = 5$, and $\delta = 30$. This is the same problem as the one given in [2], except for the value of the parameter δ, which in [2] is taken to be 20. Instead, we consider $\delta = 30$ as in [25, 41], since the flame layer in this case is much thinner, and higher mesh adaptation is required. We use (4.11) to discretize u_{xx} and use a third-order TVD RK scheme for temporal integration. To illustrate how we apply the Neumann boundary condition on a nonuniform grid, we again consider a grid of the form (3.5). To apply the Neumann boundary condition $u_x(0,t) = 0$, we introduce a fictitious node $x_{j_{-1},k_{-1}} = -x_{j_1,k_1}$, which lies outside the physical domain,[5] and approximate the boundary condition by

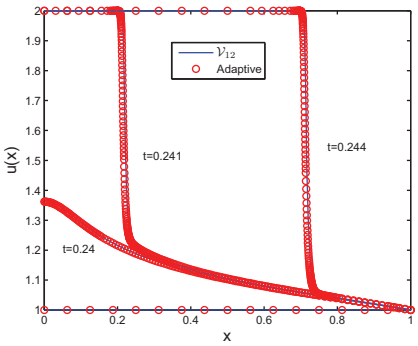
$$(u_x)^n_{j_0,k_0} = \frac{u^n_{j_1,k_1} - u^n_{j_{-1},k_{-1}}}{x_{j_1,k_1} - x_{j_{-1},k_{-1}}} = 0, \tag{5.15}$$

which implies $u^n_{j_{-1},k_{-1}} = u^n_{j_1,k_1}$. Hence, at the boundary $x = 0$, (4.11) reduces to

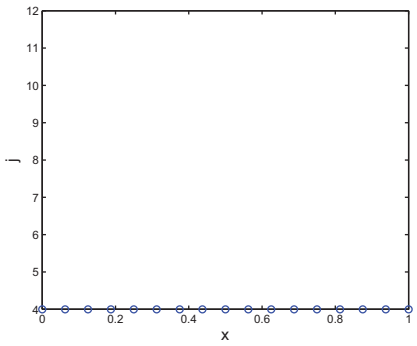
$$(u_{xx})^n_{j_0,k_0} = \frac{2\left(\frac{u^n_{j_1,k_1} - u^n_{j_0,k_0}}{x_{j_1,k_1} - 0} - \frac{u^n_{j_0,k_0} - u^n_{j_{-1},k_{-1}}}{0 - x_{j_{-1},k_{-1}}}\right)}{x_{j_1,k_1} - x_{j_{-1},k_{-1}}} = \frac{2(u^n_{j_1,k_1} - u^n_{j_0,k_0})}{(x_{j_1,k_1})^2}. \tag{5.16}$$

The numerical solutions at times $t = 0\,\mathrm{s}$, $t = 0.24\,\mathrm{s}$, $t = 0.241\,\mathrm{s}$, and $t = 0.244\,\mathrm{s}$ using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ are shown in Figure 5.4a. The other parameters used in the grid adaptation algorithm are $p = 3$, $\epsilon = 10^{-5}/2^{J_{\max}-j}$, and $N_1 = N_2 = 1$. One of the main challenges of this problem is the fact that one needs to use a very small time step to capture the transition layer during the time of ignition. This is achieved automatically by the proposed algorithm, since the algorithm is adaptive both in time and space. As the mesh gets refined, the Δt_n in the proposed algorithm for the solution of evolution PDEs given in section 4.2 also decreases. We see from Figure 5.4f that for time $t < 0.195\,\mathrm{s}$, the proposed algorithm found the solution using only about 50 to 65 points. Starting from $t = 0.195\,\mathrm{s}$ to $t = 0.2385\,\mathrm{s}$, the algorithm slowly increased the number of points to around 95 points and thereafter added points at finer levels starting at $t = 0.2385\,\mathrm{s}$. As points from finer grid levels are being added, the algorithm automatically decreases the time step and is able to capture the transition layer during the time of ignition. The L_1 errors ($E_1(u)$) along with the number of grid points used by the proposed algorithm at times $t = 0.24\,\mathrm{s}$, $t = 0.241\,\mathrm{s}$, and $t = 0.244\,\mathrm{s}$ for $J_{\max} = 12$ have been summarized in Table 5.5.

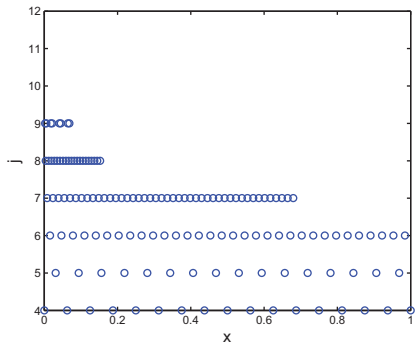
[5] Note that $x_{j_0,k_0} = 0$.



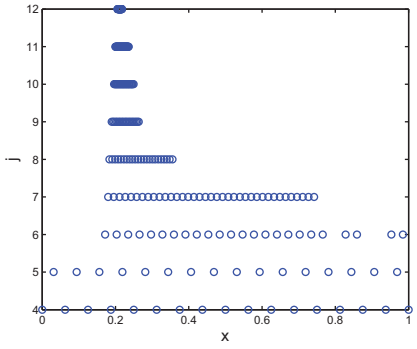
(a) *Solution* $u(x,t)$.



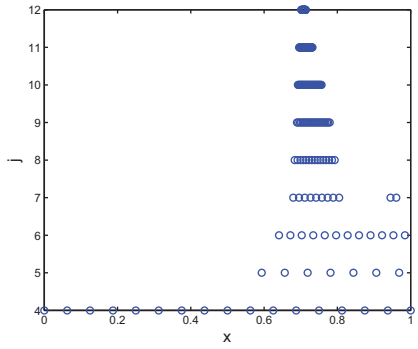
(b) *Grid point distribution at* $t = 0$ s.



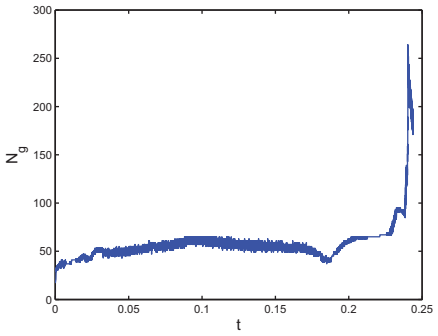
(c) *Grid point distribution at* $t = 0.24$ s.



(d) *Grid point distribution at* $t = 0.241$ s.



(e) *Grid point distribution at* $t = 0.244$ s.



(f) *Time evolution of the number of grid points.*

FIG. 5.4. *Example* 9. *Parameters used in the simulation are* $J_{\min} = 4$, $J_{\max} = 12$, $p = 3, \epsilon = 10^{-5}/2^{J_{\max}-j}$, *and* $N_1 = N_2 = 1$.

TABLE 5.5
Example 9. L_1 *errors at different times for* $J_{\max} = 12$.

t	N_g in $\mathsf{Grid}_{\text{new}}$	C (%)	$E_1(u)$
0.24	137	96.66	4.6714×10^{-4}
0.241	227	94.46	3.4834×10^{-3}
0.244	180	95.61	3.2098×10^{-3}

Example 10. Finally, we consider a Riemann initial value problem (shock tube) for the Euler equations of gas dynamics as follows:

$$u_t + f(u)_x = 0, \tag{5.17}$$

$$u(x,0) = \begin{cases} u_{\mathrm{L}}, & x < 0.5, \\ u_{\mathrm{R}}, & x > 0.5, \end{cases} \tag{5.18}$$

where

$$u = [\rho\ m\ E]^{\mathrm{T}}, \tag{5.19}$$

$$f(u) = \nu u + [0\ p\ p\nu]^{\mathrm{T}}, \tag{5.20}$$

ρ, m, E are the gas density, momentum, total energy per unit volume, respectively, $\nu = m/\rho$ is the velocity, and

$$p = (\gamma - 1)\left(E - \frac{\rho\nu^2}{2}\right) \tag{5.21}$$

is the pressure. In (5.21), γ is the ratio of specific heat, which takes the usual value of 1.4 (for air). We consider the two well-known problems, namely, Sod's problem [44], the initial data for which is given by

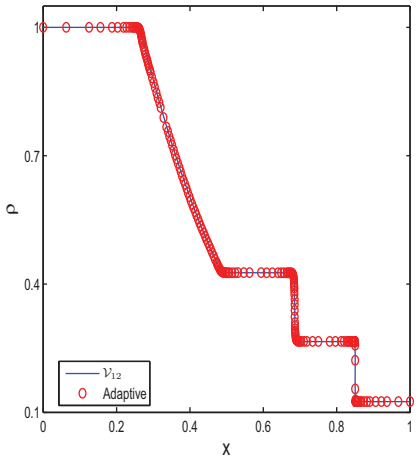
$$u_{\mathrm{L}} = [1\ 0\ 2.5]^{\mathrm{T}}, \quad u_{\mathrm{R}} = [0.125\ 0\ 0.25]^{\mathrm{T}}, \tag{5.22}$$

and Lax's problem [30], the initial data for which is given by

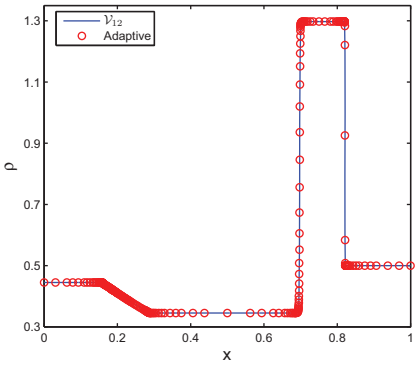
$$u_{\mathrm{L}} = [0.445\ 0.698\ 8.82]^{\mathrm{T}}, \quad u_{\mathrm{R}} = [0.5\ 0\ 1.4275]^{\mathrm{T}}. \tag{5.23}$$

We use the characteristic numerical scheme given in [43, 39] for solving this problem. The basic idea behind the characteristic scheme is to transform the nonlinear system (5.17) to a system of (nearly) independent scalar conservation laws and discretize each scalar conservation law independently in an upwind biased fashion. Then we transform the discretized system back to the original variables. We use the ENO–Roe fix scheme [43] on a nonuniform grid for obtaining the numerical flux function $\mathcal{F}^n_{j_{i\pm1/2},k_{i\pm1/2}}$ in the scalar field, and we use a third-order TVD RK scheme for temporal integration.

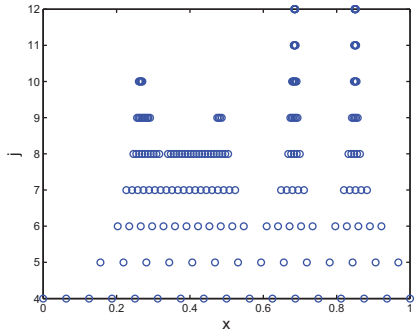
The numerical solution of the density $\rho(x,t)$ along with the grid point distributions in the adaptive mesh for Sod's problem and Lax's problem at times $t = 0.2\,\mathrm{s}$, $t = 0.13\,\mathrm{s}$, respectively, using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ are shown in Figure 5.5. The other parameters used in the grid adaptation procedure are $p = 3$, $\epsilon = 0.001$, and $N_1 = N_2 = 2$. Figure 5.5 also shows the time evolution of the number of grid points for both Sod's and Lax's problems. The L_1 errors ($E_1(\rho)$, $E_1(m)$, $E_1(E)$) along with the number of grid points used by the proposed algorithm for solving Sod's problem at times $t = 0.05\,\mathrm{s}$, $t = 0.1\,\mathrm{s}$, $t = 0.15\,\mathrm{s}$, and $t = 0.2\,\mathrm{s}$ and Lax's problem at times $t = 0.05\,\mathrm{s}$, $t = 0.1\,\mathrm{s}$, and $t = 0.13\,\mathrm{s}$ for $J_{\max} = 12$ are summarized in Table 5.6 and Table 5.7, respectively.



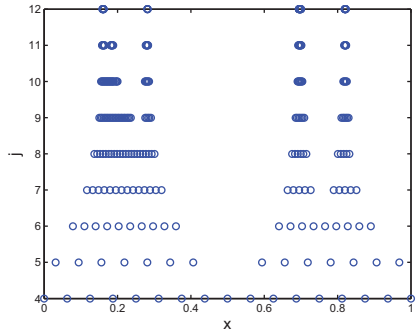
(a) *Sod's problem. Solution* $\rho(x, 0.2)$.



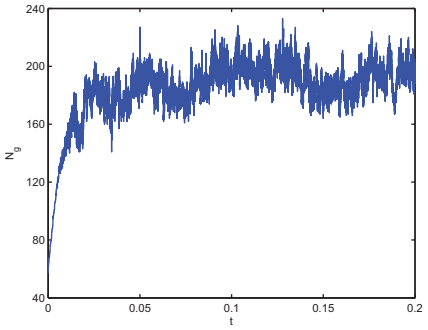
(b) *Lax's problem. Solution* $\rho(x, 0.13)$.



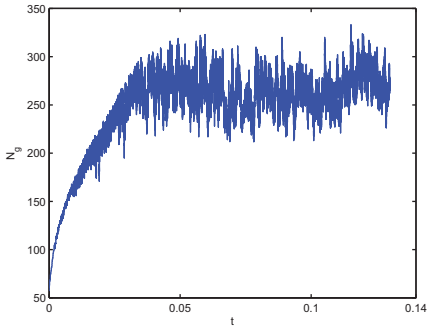
(c) *Sod's problem. Grid point distribution at* $t = 0.2$,s.



(d) *Lax's problem. Grid point distribution at* $t = 0.13$,s.



(e) *Sod's problem. Time evolution of grid points.*



(f) *Lax's problem. Time evolution of grid points.*

FIG. 5.5. *Example* 10. *Parameters used in the simulation are* $J_{\min} = 4$, $J_{\max} = 12$, $p = 3, \epsilon = 0.001$, *and* $N_1 = N_2 = 2$.

TABLE 5.6
Example 10. *Sod's problem.* L_1 *errors at different times for* $J_{\max} = 12$.

t	N_g in $\mathsf{Grid}_{\text{new}}$	C (%)	$E_1(\rho)$	$E_1(m)$	$E_1(E)$
0.05	212	94.83	1.0300×10^{-4}	1.1859×10^{-4}	2.9885×10^{-4}
0.1	189	95.39	2.8712×10^{-4}	3.2164×10^{-4}	8.3684×10^{-4}
0.15	173	95.78	4.9362×10^{-4}	5.4437×10^{-4}	1.4215×10^{-3}
0.2	195	95.24	7.8443×10^{-4}	8.1571×10^{-4}	2.1954×10^{-3}

TABLE 5.7
Example 10. *Lax's problem.* L_1 *errors at different times for* $J_{\max} = 12$.

t	N_g in $\mathsf{Grid}_{\text{new}}$	C (%)	$E_1(\rho)$	$E_1(m)$	$E_1(E)$
0.05	272	93.36	5.6092×10^{-5}	1.2380×10^{-4}	7.7312×10^{-4}
0.1	270	93.41	1.8641×10^{-4}	4.1361×10^{-4}	3.3487×10^{-3}
0.13	267	93.48	2.7005×10^{-4}	5.9612×10^{-4}	4.9735×10^{-3}

6. Conclusions. In this paper, we have proposed a novel multiresolution scheme for data compression along with a new grid adaptation algorithm for solving evolution equations. Both are shown to outperform similar data compression and grid adaptation schemes in the literature. Specifically, we have shown that the proposed approach results in fewer grid points when compared to a common adaptive grid approach. For the examples considered here, we have observed savings of up to 43% in terms of the number of grid points and a gain of about 27% in terms of speedup factors. Several examples have demonstrated the stability and robustness of the proposed algorithm. In all examples considered, the algorithm adapted dynamically to any existing or emerging irregularities in the solution by automatically allocating more grid points to the region where the solution exhibited sharp features and fewer points to the region where the solution was smooth. As a result, the computational time and memory usage can be reduced, while maintaining an accuracy equivalent to the one obtained using a fine uniform mesh.

For the solution of evolution equations in which the mesh refinement is performed at each time step, that is, $\Delta t_{\text{adapt}} = \Delta t$, choosing a value of $N_1 > 1$ will only result in more points in the grid; it will not decrease the frequency of mesh adaptation. For all other cases, the larger the N_1, the smaller the frequency of mesh adaptation will be, at the expense of a larger number of grid points in the adaptive grid. Hence, there is a trade-off between the frequency of mesh adaptation and the number of grid points. Our practical experience has shown that $N_1 = N_2 = 2$ is a good choice for solving hyperbolic equations. Future work should investigate the issue of how to choose the parameters N_1, N_2 in an optimal way, including the adaptivity of these parameters across the spatial domain.

Followup work should also concentrate on extending the proposed approach to multiple dimensions. It is expected that the savings in terms of CPU time and the number of grid points observed for the single spatial dimension case will be greater in multiple dimensions. One approach in this direction is to work directly with interpolating functions in higher dimensions and then follow the same approach as for the 1-D case. That is, use the error between the actual and interpolated values from neighboring points to determine which points to retain in the grid and which to remove. The challenge is to find a consistent way of selecting neighboring points. Another idea is to proceed in a dimension-by-dimension fashion. That is, to compute the interpola-

tive error coefficients at a particular point using an interpolation operator based on function values in the intermediate grid along one direction, while keeping the other coordinates fixed and repeating the same for all directions. These issues are left for future investigation.

Acknowledgments. We would like to thank the anonymous reviewers for their insightful suggestions.

REFERENCES

[1] S. Adjerid and J. Flaherty, *A moving mesh finite element method with local refinement for parabolic partial differential equations*, Comput. Methods Appl. Mech. Engrg., 56 (1986), pp. 3–26.

[2] S. Adjerid and J. E. Flaherty, *A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations*, SIAM J. Numer. Anal., 23 (1986), pp. 778–796.

[3] M. A. Alves, P. Cruz, A. Mendes, F. D. Magalhães, F. T. Pinho, and P. J. Oliveira, *Adaptive multiresolution approach for solution of hyperbolic PDEs*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 3909–3928.

[4] D. C. Arney and J. E. Flaherty, *A two-dimensional mesh moving technique for time dependent partial differential equations*, J. Comput. Phys., 67 (1986), pp. 124–144.

[5] I. Babuška, J. Chandra, and J. E. Flaherty, eds., *Adaptive Computational Methods for Partial Differential Equations*, SIAM, Philadelphia, 1983.

[6] M. Baines and A. Wathen, *Moving finite element methods for evolutionary problems.* I. *Theory*, J. Comput. Phys., 79 (1988), pp. 245–269.

[7] M. J. Baines, *Moving Finite Elements*, Oxford University Press, New York, 1994.

[8] J. Bell, M. Berger, J. Saltzman, and M. Welcome, *Three-dimensional adaptive mesh refinement for hyperbolic conservation laws*, SIAM J. Sci. Comput., 15 (1994), pp. 127–138.

[9] M. J. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 82 (1989), pp. 64–84.

[10] M. J. Berger and R. J. Leveque, *Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems*, SIAM J. Numer. Anal., 35 (1998), pp. 2298–2316.

[11] M. J. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[12] S. Bertoluzza, *Adaptive wavelet collocation method for the solution of Burger's equation*, Transport Theory Statist. Phys., 25 (1996), pp. 339–352.

[13] N. N. Carlson and K. Miller, *Design and application of a gradient-weighted moving finite element code* I: *In one dimension*, SIAM J. Sci. Comput., 19 (1998), pp. 728–765.

[14] H. D. Ceniceros and T. Y. Hou, *An efficient dynamically adaptive mesh for potentially singular solutions*, J. Comput. Phys., 172 (2001), pp. 609–639.

[15] A. Cohen, S. M. Kaber, S. Müller, and M. Postel, *Fully adaptive multiresolution finite volume schemes for conservation laws*, Math. Comp., 72 (2003), pp. 183–225.

[16] M. Crandall and P. Lions, *Two approximations of solutions of Hamilton-Jacobi equations*, Math. Comp., 43 (1984), pp. 1–19.

[17] E. A. Dorfi and L. O. Drury, *Simple adaptive grids for* 1-*d initial value problems*, J. Comput. Phys., 69 (1987), pp. 175–195.

[18] B. Gottschlich-Müller and S. Müller, *Adaptive finite volume schemes for conservation laws based on local multiresolution techniques*, in Hyperbolic Problems: Theory, Numerics, Applications, M. Fey and R. Jeltsch, eds., Birkhäuser, Basel, 1999, pp. 385–394.

[19] A. Harten, *Adaptive multiresolution schemes for shock computations*, J. Comput. Phys., 115 (1994), pp. 319–338.

[20] A. Harten, *Multiresolution algorithms for the numerical solution of hyperbolic conservation laws*, Comm. Pure Appl. Math., 48 (1995), pp. 1305–1342.

[21] A. Harten, *Multiresolution representation of data: A general framework*, SIAM J. Numer. Anal., 33 (1996), pp. 1205–1256.

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

[23] M. Holmström, *Solving hyperbolic PDEs using interpolating wavelets*, SIAM J. Sci. Comput., 21 (1999), pp. 405–420.

[24] M. Holmström and J. Waldén, *Adaptive wavelet methods for hyperbolic PDEs*, J. Sci. Comput., 13 (1998), pp. 19–49.
[25] W. Huang, Y. Ren, and R. D. Russell, *Moving mesh methods based on moving mesh partial differential equations*, J. Comput. Phys., 113 (1994), pp. 279–290.
[26] L. Jameson, *A wavelet-optimized, very high order adaptive grid and order numerical method*, SIAM J. Sci. Comput., 19 (1998), pp. 1980–2013.
[27] G. S. Jiang and D. Peng, *Weighted ENO schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2126–2143.
[28] G. S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.
[29] I. W. Johnson, A. J. Wathen, and M. J. Wathen, *Moving finite element methods for evolutionary problems.* II. *Applications*, J. Comput. Phys., 79 (1988), pp. 270–297.
[30] P. D. Lax, *Weak solutions of nonlinear hyperbolic equations and their numerical computation*, Commun. Pure Appl. Math., 7 (1954), pp. 195–206.
[31] R. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser, Boston, 1992.
[32] R. Li and T. Tang, *Moving mesh discontinuous Galerkin method for hyperbolic conservation laws*, J. Sci. Comput., 27 (2006), pp. 347–363.
[33] X. D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes*, J. Comput. Phys., 115 (1994), pp. 200–212.
[34] S. G. Mallat, *Multiresolution approximations and wavelet orthonormal bases of* $L^2(\mathbb{R})$, Trans. Amer. Math. Soc., 315 (1989), pp. 69–87.
[35] K. Miller, *Moving finite elements* II, SIAM J. Numer. Anal., 18 (1981), pp. 1033–1057.
[36] K. Miller and R. N. Miller, *Moving finite elements* I, SIAM J. Numer. Anal., 18 (1981), pp. 1019–1032.
[37] S. Müller, *Adaptive Multiscale Schemes for Conservation Laws*, Lect. Notes Comput. Sci. Eng. 27, Springer, New York, 2003.
[38] S. Müller and Y. Stiriba, *Fully adaptive multiscale schemes for conservation laws employing locally varying time stepping*, J. Sci. Comput., 30 (2007), pp. 493–531.
[39] S. Osher and R. P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer, New York, 2003.
[40] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 902–922.
[41] L. R. Petzold, *Observations on an adaptive moving grid method for one-dimensional systems for partial differential equations*, Appl. Numer. Math., 3 (1987), pp. 347–360.
[42] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.
[43] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes* II, J. Comput. Phys., 83 (1989), pp. 32–78.
[44] G. A. Sod, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Comput. Phys., 27 (1978), pp. 1–31.
[45] H. Tang and T. Tang, *Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws*, SIAM J. Numer. Anal., 41 (2003), pp. 487–515.
[46] J. W. Thomas, *Numerical Partial Differential Equations*, Springer, New York, 1995.
[47] J. F. Thompson, *A survey of dynamically-adaptive grids in the numerical solution of partial differential equations*, Appl. Numer. Math., 1 (1985), pp. 3–27.
[48] O. V. Vasilyev, *Solving multi-dimensional evolution problems with localized structures using second generation wavelets*, Int. J. Comput. Fluid Dyn., 17 (2003), pp. 151–168.
[49] O. V. Vasilyev and C. Bowman, *Second-generation wavelet collocation method for the solution of partial differential equations*, J. Comput. Phys., 165 (2000), pp. 660–693.
[50] J. Waldén, *Filter bank methods for hyperbolic PDEs*, J. Numer. Anal., 36 (1999), pp. 1183–1233.