Adaptive Multiresolution Mesh Refinement for the Solution of Evolution PDEs

S. Jain[*] , P. Tsiotras[†] and H.-M. Zhou[‡]
Georgia Institute of Technology, Atlanta, GA 30332-0150, USA

***Abstract*— In this paper we propose a novel multiresolution adaptive mesh refinement algorithm for solving Initial-Boundary Value Problems (IBVP) for evolution partial differential equations. The proposed algorithm dynamically adapts the grid to any existing or emerging irregularities in the solution, thus refining the grid only at 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 other wavelet- or multiresolution-based mesh refinement techniques. Several examples show the robustness, stability and savings (in terms of CPU time) of our algorithm.**

I. Introduction

It is well known that the solution of evolution partial differential equations (PDEs) is often not smooth even if the initial conditions are smooth. For instance, kinks may arise in Hamilton-Jacobi (HJ) equations and 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 based on wavelets or similar multiresolution analyses have been proposed in the literature for the solution of PDEs [1], [2], [3], [4], [5], [6], [7]. Wavelet or multiresolution-based methods take advantage of the fact that functions with localized regions of sharp transition can be very well compressed, and what makes wavelets attractive for solving PDEs is their multiresolution properties. Mallat [8] formulated the basic idea of multiresolution analysis for orthonormal wavelets in $L^2(\mathbb{R})$. Harten [9] 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. His contribution to the theory of wavelets lies mainly in his extension of wavelets to nonuniform grids.

Recently, Alves et al. [7] proposed an adaptive multiresolution mesh refinement scheme, similar to the multiresolution mesh refinement approach proposed by Harten [1], [2] and the interpolating wavelet approach proposed by Bertoluzza [3] and Holmstrom [6]. The authors in [1], [2], [6] used the interpolating subdivision scheme of [1], [10], [11] to construct the interpolating polynomials, whereas Alves et al. [7] used the high-resolution schemes SMART [12] and MINMOD [13] for constructing the interpolating polynomials. These approaches share similar underlying ideas which will be discussed shortly. Henceforth, we call this approach the traditional mesh refinement approach.

In this paper, we propose a new multiresolution mesh refinement technique for solving initial-boundary value problems (IBVP) for evolution PDEs. The key feature of our algorithm is that it is a “top-down” approach, in which we use the most updated information to make predictions as opposed to the traditional approach in which function values at all points belonging to a particular resolution level are interpolated only from the corresponding points at the coarser level. Unlike the traditional approach, our interpolations are not restricted to using only the retained points at the coarser level, but we also use the retained points at the same level and even the next finer level. This allows for a more accurate interpolation which, in turn, leads to fewer points in the final grid.

The paper is organized as follows. In the first part of the paper we start by formulating the problem and we present the algorithm for mesh refinement. Next, we compare the proposed mesh refinement approach with the traditional approach, and we show that the proposed algorithm results, in general, in a fewer number of grid points compared to the traditional approach. 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 a major speed up in terms of computational time compared to the uniform mesh case and increase in the speed-up factor compared to traditional approach.

II. Problem Statement

Many problems in engineering and physics can be written in the form of an initial-boundary value problem (IBVP) for an evolution equation:

$$\text{(IBVP)}: \quad \begin{cases} u_t + f(u_{xx}, u_x, u, x) = 0 & \text{in } \mathcal{D} \times (0,\infty), \\ u = g & \text{on } \overline{\mathcal{D}} \times \{t = 0\}, \end{cases} \tag{1}$$

[*]Ph.D. candidate, School of Aerospace Engineering, Email: sachin.jain@gatech.edu.

[†]Professor, School of Aerospace Engineering, Email: tsiotras@gatech.edu.

[‡]Assistant Professor, School of Mathematics, Email: hmzhou@math.gatech.edu.

where $\overline{\mathcal{D}} = \mathcal{D} \cup \partial\mathcal{D}$, with $\mathcal{D} \subset \mathbb{R}$ bounded. The function $f : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathcal{D} \to \mathbb{R}$, and the initial function $g : \overline{\mathcal{D}} \to \mathbb{R}$ are given. The unknown is the function $u : \overline{\mathcal{D}} \times [0, \infty) \to \mathbb{R}$. 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 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.

III. Adaptive Gridding

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

$$\mathcal{V}_j = \{x_{j,k} \in [0, 1] : x_{j,k} = k/2^j,\ 0 \le k \le 2^j\}, \qquad (2)$$

where $J_{\min} \le j \le J_{\max}$, 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,

$$\mathcal{W}_j = \{y_{j,k} \in [0, 1] : y_{j,k} = (2k + 1)/2^{j+1},\ 0 \le k \le 2^j - 1\}, \qquad (3)$$

$J_{\min} \le j \le J_{\max} - 1$. Hence, $x_{j+1,k} \in \mathcal{V}_{j+1}$ is given by

$$x_{j+1,k} = \begin{cases} x_{j,k/2}, & \text{if } k \text{ is even,} \\ y_{j,(k-1)/2}, & \text{otherwise.} \end{cases} \qquad (4)$$

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 compliment 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 $\lim_{J_{\max}\to\infty} \overline{\mathcal{V}_{J_{\max}}} = \overline{\mathcal{D}}$. The sequence of subspaces $\mathcal{W}_j$ satisfy $\mathcal{W}_j \cap \mathcal{W}_\ell = \varnothing$ for $j \ne \ell$.

A. Grid Adaptation

Assume we are given a nonuniform grid of the form

$$\begin{aligned} Grid \;=\; & \{x_{j_i,k_i} : x_{j_i,k_i} \in [0, 1],\ 0 \le k_i \le 2^{j_i}, \\ & J_{\min} \le j_i \le J_{\max},\ \text{for } i = 1, \ldots, N,\ \text{and} \\ & x_{j_i,k_i} < x_{j_{i+1},k_{i+1}},\ \text{for } i = 0 \ldots N - 1\}. \end{aligned} \qquad (5)$$

For simplicity, we denote $u(x, t)$ evaluated at $x = x_{j,k}$ and $t = t_n$ by $u^n_{j,k}$, 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 condition [14] for hyperbolic equations and the von Neumann condition [14] for all other evolution equations.

For grid adaptation, we first need a procedure for computing the function value at any point $x_{\text{interp}} \in (0, 1)$ from the function values in $U = \{u^n_{j,k} : x_{j,k} \in Grid\}$, using an interpolating polynomial of degree p. To this end, the first step is to find the $p+1$ nearest points $X_{\text{near}} = \{x_{j_\ell,k_\ell}\}_{\ell=i}^{i+p} \in Grid$ to x_{interp}, where $0 \le i \le N - p$. By $p + 1$ nearest points here we mean one neighboring point on the left of x_{interp}, one neighboring point on the right of x_{interp} and the remaining $p - 1$ points are the points nearest to x_{interp} in the set $Grid$. In case two points are at the same distance, that is, if a point on the left and a point on the right are equidistant to x_{interp}, then we choose a point so as to equalize the number of points on both sides.

Once we have found the $p + 1$ nearest points of the set X_{near}, we construct an interpolating polynomial $\hat{u}(x)$ of order p passing through these $p + 1$ points. One may use Neville's algorithm to construct the respective interpolating polynomials on the fly. The interpolating polynomials can also be constructed using high-resolution schemes such as the essentially non-oscillatory (ENO) scheme [15], [16]. The only difference in that case would be that we would take one neighboring point on the left and one neighboring point on the right of x_{interp}, and choose the remaining $p - 1$ points that give the least oscillatory polynomial.

The "top-down" approach of our algorithm allows one to add and remove points using the most updated information. Now suppose $u(x, t_n)$ is specified on a grid $Grid_{\text{old}}$, with corresponding solution values $U_{\text{old}} = \{u^n_{j,k} : x_{j,k} \in Grid_{\text{old}}\}$, where $Grid_{\text{old}}$ can be either regular or irregular[1]. We assume $Grid_{\text{old}} \supseteq \mathcal{V}_{J_{\min}}$. Our aim is to find a new grid $Grid_{\text{new}}$, by adding or removing points from $Grid_{\text{old}}$, reflecting local changes in the solution. To this end, we initialize an intermediate grid $Grid_{\text{int}} = \mathcal{V}_{J_{\min}}$, with function values $U_{\text{int}} = \{u^n_{J_{\min},k} : u^n_{J_{\min},k} \in U_{\text{old}},\ 0 \le k \le 2^{J_{\min}}\}$, and set $j = J_{\min}$. The mesh refinement algorithm proceeds as follows:

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

$$Y = \{y_{j,k_i} : y_{j,k_i} \in \mathcal{W}_j \cap Grid_{\text{old}},\ \text{for } i = 1, \ldots, M\}. \qquad (6)$$

2. Set $i = 1$.
   a) Compute the interpolated function value at point $y_{j,k_i} \in Y$, $\hat{u}(y_{j,k_i})$, from the points in $Grid_{\text{int}}$ and their function values in the set U_{int} using the procedure mentioned above.
   b) Calculate the interpolative error coefficient d_{j,k_i} at the point y_{j,k_i}[2], $d_{j,k_i} = |u(y_{j,k_i}, t_n) - \hat{u}(y_{j,k_i})|$. The interpolative error coefficient is a measure of the local smoothness of the function. If the value of d_{j,k_i} is below the prescribed threshold ϵ, then reject y_{j,k_i} since the function value at this point can be reconstructed from the points in $Grid_{\text{int}}$ with an error no larger than ϵ and goto Step 2(f), otherwise add y_{j,k_i} to the intermediate grid $Grid_{\text{int}}$ and move on to the next step[3].

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

[2]Note that $u(y_{j,k}, t_n) \in U_{\text{old}}$ for all $y_{j,k} \in Y$.

[3]A function with isolated singularities but otherwise regular, will have most of the interpolative error coefficients close to zero. This allows a compact approximation of the function without significant loss of accuracy.

c) Add to $Grid_{\rm int}$ 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 $Grid_{\rm int}$ $2N_2$ neighboring points at the next finer level $\{y_{j+1,2k_i+\ell}\}_{\ell=-N_2+1}^{N_2}$. This step accounts for the possibility of the solution becoming steeper in this region and the value of N_2 is chosen based on the problem.

e) Add the function values at all the newly added points to $U_{\rm int}$. 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 $Grid_{\rm old}$ and their function values in $U_{\rm old}$ using the procedure described above.

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

3. Increment j by 1. If $j < J_{\max}$ goto Step 1, otherwise move on to the next step.
4. Terminate the algorithm. The final nonuniform grid is $Grid_{\rm new} = Grid_{\rm int}$ and their corresponding function values are in the set $U_{\rm new} = U_{\rm int}$.

The adaptive grid generated using the proposed algorithm depends on how we select points along the grid, that is, whether we move from left to right or from right to left across each level. Note also that it is not mandatory to traverse across a level 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.

B. Comparison Between the Traditional and the Proposed Approach

The proposed grid adaptation algorithm results, in general, in a fewer number of grid points when compared to the traditional approach. First, we explain why this is so and then we give several examples to demonstrate this fact. For details on the traditional grid adaptation approach the reader is referred to [1], [2], [6], [7].

In the traditional approach, 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 $0 \leq k \leq 2^j-1$ and $J_{\min} \leq j \leq J_{\max}-1$, is greater than the prescribed threshold, we add $y_{j,k}$ to the adaptive grid, and at the same time we 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 these newly added points also for interpolating the remaining points at level $\mathcal{W}_j$ and the levels below it. In other words, in the proposed approach $\{g(y_{j,k})\}_{k=0}^{2^j-1}$ are interpolated from the function values at the points in $\mathcal{V}_j \oplus \mathcal{W}_j \oplus \mathcal{W}_{j+1}$ for $J_{\min} \leq j \leq J_{\max} - 2$ 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. Moreover, in the traditional approach, when a point $y_{j,k}$, where $0 \leq k \leq 2^j - 1$ and $J_{\min} \leq j \leq J_{\max} - 1$, is added to the grid then we also include its parents, which were used to predict the function value at that point. The parents are not needed for approximating g 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 which already exist 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 separate track of the parents from the rest of the points as in the traditional approach.

Example 1: Consider a dyadic grid $\mathcal{V}_4$ and the function $g(x)$, which is zero for all $x \in [0,1] \setminus x_{4,k}$, and an impulse located at $x = k/2^4$, where $0 \leq k \leq 16$. Let $J_{\min} = 0$, $J_{\max} = 4$, $p = 1$, $\epsilon = 0.1$, and $N_1 = N_2 = 1$. Table I shows the number of grid points in the adaptive grid using both the traditional method ($N_{\rm g_t}$) and the proposed method ($N_{\rm g_p}$) for $k = 0, \dots, 16$. 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 traditional approach by up to 33%.

TABLE I
EXAMPLE 1. TRADITIONAL METHOD VS. PROPOSED METHOD.

k	$N_{\rm g_t}$	$N_{\rm g_p}$	Ratio	k	$N_{\rm g_t}$	$N_{\rm g_p}$	Ratio
0	9	9	1	9	9	6	0.67
1	6	5	0.83	10	12	9	0.75
2	9	7	0.78	11	9	6	0.67
3	8	6	0.75	12	12	11	0.92
4	12	11	0.92	13	7	5	0.71
5	9	6	0.67	14	9	7	0.78
6	12	9	0.75	15	6	4	0.67
7	9	6	0.67	16	9	9	1
8	13	13	1				

Example 2: Consider the function, $g(x) = \sin(2\pi x) + e^{-\beta(x-0.5)^2}$, which is smooth everywhere except for the peak

at $x = 0.5$. Here β is a parameter that controls the width of the Gaussian. We let $\beta = 20,000$. For this example, we let $J_{\min} = 2$, $J_{\max} = 10$, $p = 3$, and $N_1 = N_2 = 1$. The results for different thresholds are summarized in Table II. We observe once again that the proposed algorithm outperforms the traditional approach by up to 33%.

TABLE II
EXAMPLE 2. TRADITIONAL METHOD VS. PROPOSED METHOD.

ϵ	N_{gt}	N_{gp}	Ratio	ϵ	N_{gt}	N_{gp}	Ratio
10^{-2}	69	49	0.71	10^{-7}	429	304	0.71
10^{-3}	89	79	0.89	10^{-8}	545	530	0.97
10^{-4}	125	116	0.93	10^{-9}	1005	957	0.95
10^{-5}	181	150	0.83	10^{-10}	1025	1021	0.99
10^{-6}	277	185	0.67				

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

IV. Numerical Solution of the IBVP for Evolution Equations

Based on the problem and the desired accuracy, 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 proposed grid adaptation algorithm. The final time t_f is assumed to be given.

To solve (IBVP) on an adaptive grid, we first initialize $Grid_{\mathrm{old}} = \mathcal{V}_{J_{\max}}$, $U_{\mathrm{old}} = \{g(x_{J_{\max},k})\}_{k=0}^{2^{J_{\max}}}$, and set $t = 0$, $n = 0$. Then the algorithm proceeds as follows:

1. Given $Grid_{\mathrm{old}}$, U_{old} find the new grid $Grid_{\mathrm{new}}$ and the function values at all the points in $Grid_{\mathrm{new}}$, $U_{\mathrm{new}} = \{u_{j,k}^n : x_{j,k} \in Grid_{\mathrm{new}}\}$, using the proposed grid adaptation algorithm described in Section III-A. The new grid $Grid_{\mathrm{new}}$ is the grid on which we will propagate the solution from time t to time $t + \Delta t_{\mathrm{adapt}}$, where $\Delta t_{\mathrm{adapt}}$ is the time after which the grid should be adapted again, calculated from the approximate time the solution will take to move N_1 grid points.
2. Compute the solution at time $t = t_{n+1}$ at all the points belonging to $Grid_{\mathrm{new}}$, $U_{\mathrm{new}} = \{u_{j,k}^{n+1} : x_{j,k} \in Grid_{\mathrm{new}}\}$, using any numerically stable scheme, where the numerical scheme for discretizing (IBVP) depends on $f(u_{xx}, u_x, u, x)$, and increment n by 1. Keep on repeating this step while $t < \Delta t_{\mathrm{adapt}}$. If $t \geq t_f$ terminate the algorithm.
3. Reassign the sets: $Grid_{\mathrm{old}} \leftarrow Grid_{\mathrm{new}}$, $U_{\mathrm{old}} \leftarrow U_{\mathrm{new}}$. 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 only check the retained points in $Grid_{\mathrm{old}}$ to further add and remove points in the grid. The interpolative error coefficients are computed only at the points $y_{j,k} \in Grid_{\mathrm{old}}$, and the solution U_{old} for all $y_{j,k} \in Grid_{\mathrm{old}}$ is known from the previous step.
4. Calculate the new time at which the next mesh adaptation should take place $t_{\mathrm{adapt}} = t + \Delta t_{\mathrm{adapt}}$. Goto Step 1.

V. Numerical Examples

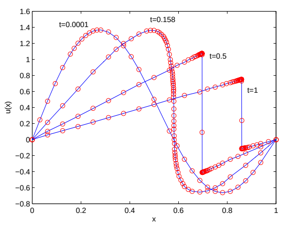
In this section, we present several examples to demonstrate the stability and robustness of our algorithm. These examples illustrate the algorithm's ability to automatically capture and follow any existing or self-sharpenning features of the solution that develop in time.

Example 3: First, we consider a nonlinear conservation law: $u_t + \left(\frac{1}{2}u^2\right)_x = 0$ (Burgers' equation). We use the same smooth initial condition and the Dirichlet boundary condition as in [7], that is, $g(x) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x)$, $u(0,t) = u(1,t) = 0$, to check the ability of the proposed algorithm to capture the shock and at the same time to compare the performance of the proposed algorithm with the traditional approach. 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.

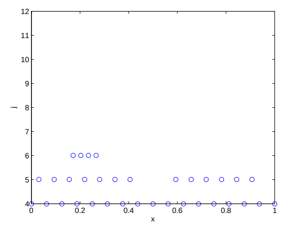
For solving the above mentioned problem, we use the ENO-Roe scheme proposed by Shu and Osher [16] along with a third-order total variation diminishing (TVD) Runge-Kutta (RK) scheme [16]. The numerical solution at times $t = 0\,\mathrm{s}$, $t = 0.158\,\mathrm{s}$, $t = 0.5\,\mathrm{s}$ and $t = 1\,\mathrm{s}$ using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ are shown in Fig. 1(a). Fig. 1 also shows the grid point distribution in the adaptive mesh at times $t = 0\,\mathrm{s}$, $t = 0.14\,\mathrm{s}$, $t = 0.158\,\mathrm{s}$ and $t = 1\,\mathrm{s}$. 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 and smoother. Similar conclusions can be drawn by looking at the time evolution of the number of grid points (N_{g}) in Fig. 1(f). This clearly 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 average number of points ($N_{\mathrm{g_{ave}}}$) used by the proposed algorithm for different $J_{\max} = 8, 9, 10, 11, 12$ are summarized in Table III. We observe a major speed up (S) in the computational time compared to the uniform mesh. The speed-up factors increase significantly with mesh refinement at an approximate rate of two. For the case $J_{\max} = 12$, the proposed algorithm used an average of 57 points for most part of the computations as opposed to the average of about 64 points used by the algorithm of Alves et al. [7] for most part of the computations, which shows an 11% reduction in the number of points used by the proposed algorithm to compute the solution with comparable accuracy. Correspondingly, the proposed algorithm resulted in a 10% higher speed-up factor, which is 55.3 as opposed to 50.2 reported in [7]. This clearly demonstrates the superiority of the proposed approach compared to the traditional approach.

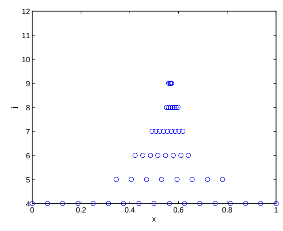
Example 4: Consider the scalar reaction-diffusion problem that appears in combustion problems [17]: $u_t - u_{xx} - \frac{Re^{\delta}}{a\delta}(1 + a - u)e^{-\delta/u} = 0$, $u_x(0,t) = 0$, $u(1,t) = 1$, $u(x,0) = 1$. The solution u represents the temperature of



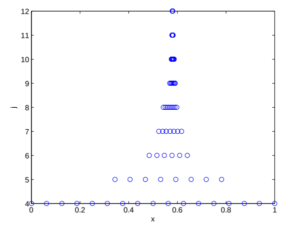
(a) Solution $u(x,t)$.



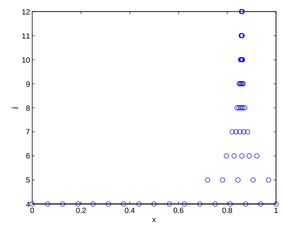
(b) Grid point distribution at $t = 0.0001$ s.



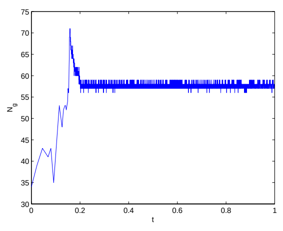
(c) Grid point distribution at $t = 0.14$ s.



(d) Grid point distribution at $t = 0.158$ s.



(e) Grid point distribution at $t = 1$ s.



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

Fig. 1. Example 3. Parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 12,\ p = 1, \epsilon = 0.02,\ N_1 = N_2 = 2$.

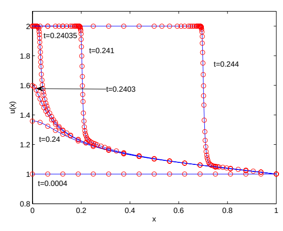
TABLE III

EXAMPLE 3. COMPUTATIONAL TIMES FOR UNIFORM VS. ADAPTIVE MESH.

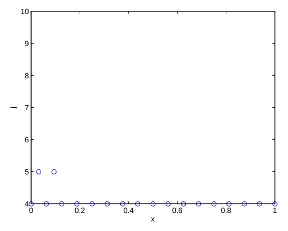
	Uniform Mesh		Adaptive Mesh		
$J_{\max}$	N_g in $\mathcal{V}_{J_{\max}}$	t_{cpu} (s)	$N_{g_{\text{ave}}}$	t_{cpu} (s)	S
8	$2^8 + 1 = 257$	2.7106	39	0.6932	3.9
9	$2^9 + 1 = 513$	9.3851	43	1.4026	6.7
10	$2^{10} + 1 = 1025$	36.6631	48	2.8098	13.0
11	$2^{11} + 1 = 2049$	223.8606	53	7.8836	28.4
12	$2^{12} + 1 = 4097$	804.9415	58	14.5679	55.3

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 $x = 0$. At 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 speed proportional to $e^{a\delta}/2(1 + a)$. We use the same problem parameters as in [17] namely, $a = 1$, $R = 5$, and $\delta = 30$.

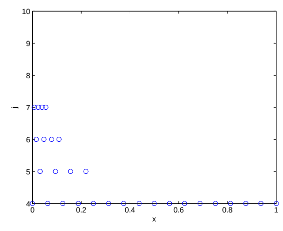
We use the centered second difference scheme [14] to discretize u_{xx} and use a second-order TVD RK scheme [16] for temporal integration. The numerical solutions at times $t = 0$ s, $t = 0.24$ s, $t = 0.2403$ s, $t = 0.24035$ s, $t = 0.241$ s and $t = 0.244$ s using a grid with $J_{\min} = 4$ and $J_{\max} = 10$



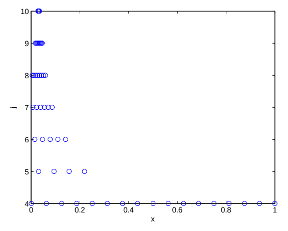
(a) Solution $u(x,t)$.



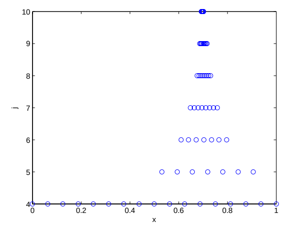
(b) Grid point distribution at $t = 0.24$ s.



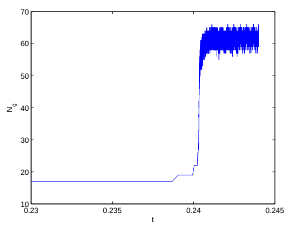
(c) Grid point distribution at $t = 0.2403$ s.



(d) Grid point distribution at $t = 0.24035$ s.



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



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

Fig. 2. Example 4. Parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 10,\ p = 3, \epsilon = 0.001,\ N_1 = 2$, and $N_2 = 1$.

are shown in Fig. 2(a). 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, Δt_n in the procedure for the numerical solution of IBVP for evolution equations (Section IV) also decreases. In Fig. 2(f) we show the time evolution of grid points starting at $t = 0.23$ s since the number of points for $t \leq 0.23$ s remains constant. We see from Fig. 2(f) that for time $t < 0.239$ s the proposed algorithm found the solution at grid $\mathcal{V}_4$ and efficiently added points at finer levels starting at $t = 0.239$ s. As the points from finer grid levels are being added, the algorithm automatically decreases the time step and is able to accurately capture the transition layer during the time of ignition.

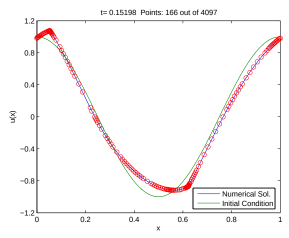
Example 5: Next, we consider the Hamilton-Jacobi (HJ) equation: $u_t + f(u_x) = 0$. Let $f(u_x) = -\cos(\alpha u_x + 1)$, where α is a constant, $g(x) = -\cos \pi(2x-1)$, and $u(0,t) = u(1,t)$. The choice of $\alpha = 0.5$ results in the commonly used test problem for 1-D HJ equations given in [18]. 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 $t = 1.5/\pi^2$ s.

For solving the above mentioned problem, we use the Lax-Friedrich's (LF) scheme [16], [18] for discretizing $f(u_x)$ along with WENO to approximate the spatial derivatives in the LF discretization. The temporal integration was performed using a third-order TVD RK scheme. The numerical solutions for all the cases at time $t = 1.5/\pi^2$ s using a grid with $J_{\min} = 4$ and $J_{\max} = 12$ along with the corresponding grid point distributions are shown in Figure 3. The solutions at $t = 1.5/\pi^2$ 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. Because of the space constraints, we only show the time evolution of the number of grid points for the case $\alpha = 1$ and $\alpha = 1.5$ in Figures 3(g) and 3(h) respectively.

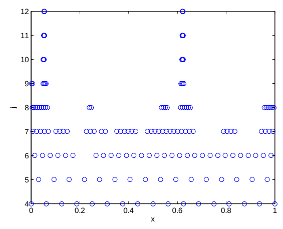
VI. Conclusions

In this paper, we have proposed a novel grid adaptation algorithm for solving evolution PDEs, which is shown to outperform similar grid adaptation schemes. Specifically, we have shown that the proposed approach results in fewer grid points in the grid compared to a common adaptive grid approach, thereby reducing the overall computation time. Several examples have demonstrated the efficiency, stability, and robustness of the proposed algorithm.

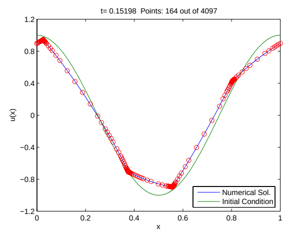
Acknowledgement: This work was partially supported by NSF award no. CMS-0510259 and DMS-0410062.



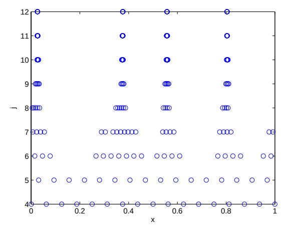
(a) Solution $u(x, 1.5/\pi^2)$ for $\alpha = 0.5$.



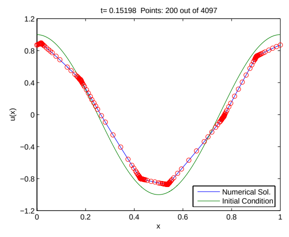
(b) Grid point distribution at $t = 1.5/\pi^2$ s for $\alpha = 0.5$.



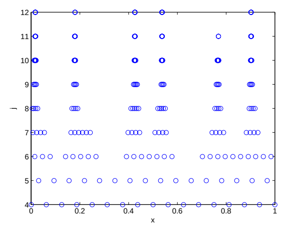
(c) Solution $u(x, 1.5/\pi^2)$ for $\alpha = 1$.



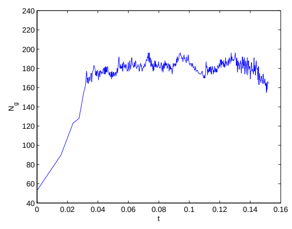
(d) Grid point distribution at $t = 1.5/\pi^2$ s for $\alpha = 1$.



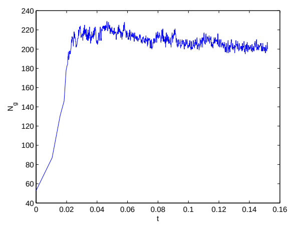
(e) Solution $u(x, 1.5/\pi^2)$ for $\alpha = 1.5$.



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



(g) Time evolution of the number of grid points for $\alpha = 1$.



(h) Time evolution of the number of grid points for $\alpha = 1.5$.

Fig. 3. Example 5. The parameters used in the simulation are $J_{\min} = 4,\ J_{\max} = 12,\ p = 3, \epsilon = 0.0001,\ N_1 = N_2 = 1$.

References

[1] A. Harten, "Adaptive multiresolution schemes for shock computations," *Journal of Computational Physics*, vol. 115, pp. 319–338, 1994.

[2] ——, "Multiresolution algorithms for the numerical solution of hyperbolic conservation laws," *Comm. Pure Appl. Math.*, vol. 48, pp. 1305–1342, 1995.

[3] S. Bertoluzza, "Adaptive wavelet collocation method for the solution of Burger's equation," *Transport Theory and Statistical Physics*, vol. 25, pp. 339–352, 1996.

[4] L. Jameson, "A wavelet-optimized, very high order adaptive grid and order numerical method," *SIAM Journal on Scientific Computing*, vol. 19, pp. 1980–2013, 1998.

[5] M. Holmström and J. Waldén, "Adaptive wavelet methods for hyperbolic PDEs," *Journal of Scientific Computing*, vol. 13, no. 1, pp. 19–49, 1998.

[6] M. Holmström, "Solving hyperbolic PDEs using interpolating wavelets," *SIAM Journal on Scientific Computing*, vol. 21, pp. 405–420, 1999.

[7] 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," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, pp. 3909–3928, 2002.

[8] S. G. Mallat, "Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$," in *Transactions of the American Mathematical Society*, vol. 315, 1989, pp. 69–87.

[9] A. Harten, "Multiresolution representation of data: A general framework," *SIAM Journal on Numerical Analysis*, vol. 33, no. 3, pp. 1205–1256, 1996.

[10] G. Deslauriers and S. Dubuc, "Symmetric iterative interpolation processes," *Constructive Approximation*, vol. 5, pp. 49–68, 1989.

[11] D. L. Donoho, "Interpolating wavelet transforms," Department of Statistics, Stanford University, Stanford, CA, Tech. Rep., 1992.

[12] P. H. Gaskell and A. K. C. Lau, "Curvature compensated convective transport: SMART, a new boundedness preserving transport algorithm," *International Journal for Numerical Methods in Fluids*, vol. 8, pp. 617–641, 1988.

[13] A. Harten, "High resolution schemes for hyperbolic conservation laws," *Journal of Computational Physics*, vol. 49, pp. 357–393, 1983.

[14] J. W. Thomas, *Numerical Partial Differential Equations*. NY: Springer, 1995.

[15] A. Harten, B. Enquist, S. Osher, and S. Chakravarthy, "Uniformly high-order accurate essentially non-oscillatory schemes III," *Journal of Computational Physics*, vol. 71, pp. 231–303, 1987.

[16] S. Osher and R. P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*. New York: Springer, 2003.

[17] W. Huang, Y. Ren, and R. D. Russell, "Moving mesh methods based on moving mesh partial differential equations," *Journal of Computational Physics*, vol. 113, pp. 279–290, 1994.

[18] S. Osher and C.-W. Shu, "High order essentially non-oscillatory schemes for Hamilton-Jacobi equations," *SIAM Journal on Numerical Analysis*, vol. 28, pp. 902–921, 1991.