Adaptive Mesh Refinement and its Application to Magneto-Hydrodynamics

Shengtai Li

ABSTRACT. In this paper we present three improvements over the Berger-Colella's adaptive mesh refinement (AMR) approach: adaptive clustering algorithm, conservative prolongation and restriction for cylindrical and spherical grid, and divergence-free reconstruction for AMR.

1. Introduction

The adaptive mesh refinement (AMR) method [**4, 5**] refines the numerical approximation of partial differential equations (PDEs) in space and time. It has become one of the most popular adaptive methods to solve the time-dependent PDEs with multiple time and length scales.

AMR uses block-structure and each level of grid can contain several logical rectangular grids (called *patch*) which can be integrated independently. It refines the time as well as space. More but smaller time steps are taken on fine grids than on coarse grids. The hierarchical nature of the grid system allows the different sized time steps to be interleaved such that the simulation remains time accurate. One of most intriguing components in AMR algorithm is the *clustering algorithm*, which is used to group the cells that require refinement into several patches. It is sensitive, complex and largely determines the efficiency of the AMR method. In this paper, we propose an adaptive clustering algorithm that can greatly reduce the number of patches generated without introducing too many extra cells.

It is critical to conserve the properties of solutions when the mesh resolution changes. Berger and Colella [**4**] proposed an adaptive mesh refinement (AMR) scheme for hydrodynamics (HD) to conserve scalar quantities (e.g., mass, energy) and numerical fluxes. First the conservative interpolation is used whenever it is needed. Then a flux-correction scheme is performed at the fine-coarse interface during the synchronization of different levels of grid. The conservative interpolation is quite different when applied to the cylindrical and spherical grid. The

1991 *Mathematics Subject Classification.* Primary 65M06, 65M50; Secondary 76M12, 76W05.

Key words and phrases. adaptive mesh refinement, divergence-free, magnetohydrodynamics (MHD), cylindrical geometry, spherical geometry, finite volume method.

This research was performed under the auspices of the Department of Energy. It was supported by the Laboratory Directed Research and Development Program at Los Alamos.

volume-based interpolation is required. In this paper we propose a second-order conservative interpolation scheme for cylindrical and spherical geometry.

Additional challenges are presented for AMR in physical systems satisfying the Stokes's law type of equation with the divergence-free evolution of vector fields, such as velocity fields in incompressible HD and magnetic fields in magnetohydrodynamics (MHD). The divergence-free evolution of magnetic fields is an important issue in developing an MHD code even for a single non-adaptive grid. Brackbill and Barnes [**7**] have shown that the discretization error with respect to the divergence of the magnetic field ($\nabla \cdot \mathbf{B}$) usually grows exponentially during the computations, causing an artificial force parallel to the magnetic field and destroying the correctness of the solutions.

Several recent studies by Balsara [**1**] and Toth and Roe [**12**] have addressed the question of preserving the divergence-free condition when the mesh is adaptive. The basic idea is to construct a divergence-free interpolation formula for the coarse cell which is being refined. This approach works well for AMR with a refinement ratio of two. However the interpolation polynomial can become quite complicated if the refinement ratio is larger than 2. As suggested in [**1**], the procedure should be applied recursively for refinement ratio of 4 or 8, and new formulae have to be derived for other refinement ratios due to the additional degrees of freedom needed to match the increased number of fine-grid faces. In this paper, we propose a new approach that maintains the divergence-free condition with AMR. This new method is simple, can be implemented efficiently for arbitrary refinement ratios, and can be generalized to different orthogonal and curvilinear grids.

2. AMR with Hierarchical Block Structure

Our AMR method makes use of the AMR data structure and refinement strategy proposed in [**8**]. To be read independently, we outline the method here.

2.1. Hierarchical Grid Structure. The basics of the AMR algorithm can be illustrated by a 1-D example in Fig. 2.1. The grid has three refinement levels (G_0, G_1, G_2) at time t_m. The integration starts from the coarsest level G_0, which advances one time step to t_{m+1}. Suppose the solution requires that the time stepsize for G_1 is half of that for G_0, and for G_2 is one-fourth of that for G_0. Then in the second step G_1 advances one time step to $t_{m+\frac{1}{2}}$, where the boundary conditions can be obtained from G_0. In the third step, G_2 advances two time step to $t_{m+\frac{1}{2}}$. After G_1 and G_2 reaches the same time, we update the solution of G_1 with the solution of G_2, because G_2 contains more accurate solution than G_1. In the fourth step, G_1 advances another step to t_{m+1}. In the fifth step, G_2 advances another two steps to t_{m+1}. Now G_0, G_1, G_2 reaches the same time level. We first update the solution of G_1 with that of G_2, and then update the solution of G_0 with that of G_1. The updating occurs only when the finest level grid reaches the same time level with the coarser grid. In step 6, we readapt the mesh and generate a new hierarchical grid. This readaptation can also be taken after each updating (e.g., at step 3).

2.2. Refinement strategy. The core of the AMR algorithm is in choosing how to cover the regions that need refinement with a finer grid. The *Remesh* stage to cover subdomains with higher resolution patches is the most algorithmically complex AMR operation in the refinement process. The remesh stage is divided

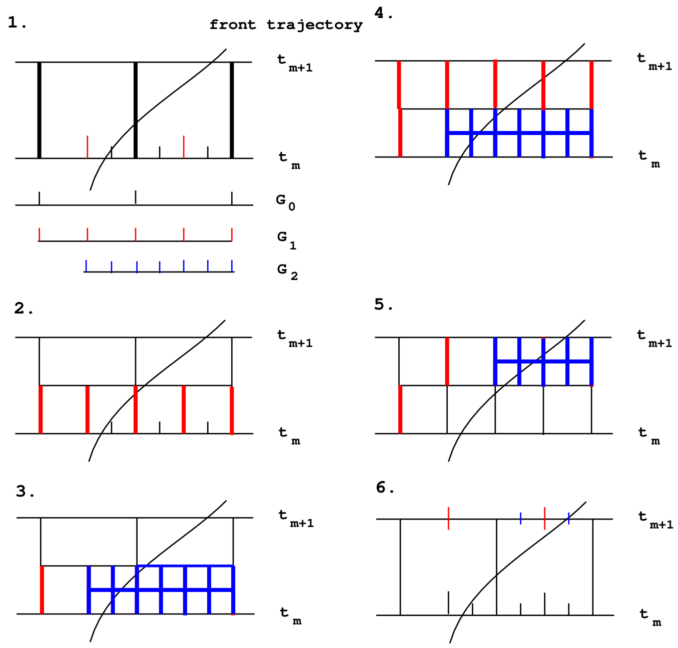

FIGURE 2.1. AMR processing: 1) Advance G_0 one step to t_{m+1}; 2) Advance G_1 one step to $t_{m+\frac{1}{2}}$; 3) Advance G_2 two steps to $t_{m+\frac{1}{2}}$, update the solution of G_1 at $t_{m+\frac{1}{2}}$ with that of G_2; 4) Advance G_1 one step to t_{m+1}; 5) Advance G_2 two steps to t_{m+1}, update the solution of G_1 (G_0) with that of G_2 (G_1); 6) Readapt the mesh to generate a new G_1 and G_2, goto 1).

into two processes: *readapt* (including refine and coarsen) the current grid and *refine* to generate a new finer grid.

The readaptation must be designed to capture the features that appear in the finer levels but would not be identified if the process started with the solution on the coarsest grid and then adapted the grid to the finer levels. Therefore, we initiate the mesh readaptation on the finest level possible. This grid is then coarsened or refined based on the selection algorithm. This process continues until all of the indicated levels have been readapted. The regridding step (defining the solution values for the readapted grid) is done in reverse. It starts from the coarsest level possible. The whole regridding procedure from the old mesh solution to the new mesh solution is called *prolongation*. After the first process, if the finest level available does not reach the maximum level allowable and needs further refinement, we start the second process to refine and generate finer level patches.

We incorporate several options in our software for user control of the refinement. The user can force a refinement through a grid file and modify the refinement parameters at any time [**8**].

During the time integration, the coarse grid is integrated first, and then the fine grid. The boundary conditions of the fine grid are defined by either external boundary conditions, or adjacent sibling patches, or interpolations from the coarse parent grid. After the integration of the fine grid is done, we transfer the more accurate fine grid solutions to the parent coarse grid. This procedure is called *restriction*. The restriction could cause a loss of conservation at the interfaces between a fine and a coarse grid. A flux correction step is proposed in [**4**] to maintain that the consistent fluxes are used at the fine-coarse interface.

3. Dynamic Clustering Algorithm

We first give a brief review on the previous clustering algorithm. The *nearest neighbor clustering* algorithm originally used by Berger et al.[**5, 4**] is highly heuristic and time-consuming. First a cluster is defined consisting of a single (arbitrary) tagged node. Other tagged nodes are added to the cluster if they are within a specified minimum inter-cluster distance from the nearest node in the cluster. Finally, clusters are merged when a node is determined to belong to more than one of them. It costs about $O(n^2) + O(p^2)$ on n nodes and p clusters requiring refinement.

The *signature and cut clustering* algorithm proposed by Berger and Rigoutsos [**6**] modifies the definition of the cutting index in the mean-cut algorithm by computing a signature list along each grid axis. The signature s_i is the number of flagged entries along the i^{th} grid surface. The zero entry in the signature list is used to indicate the best index to subdivide the domain. If there is no zero entry, the second difference of the signatures $\Delta_i = s_{i+1} - 2s_i + s_{i-1}$ is used to define $z_{i+\frac{1}{2}} = |\Delta_{i+1} - \Delta_i|$. The index with the largest $z_{i+\frac{1}{2}}$ is chosen as the cutting index. The recursion stops when either the ratio for the cluster is less than some threshold or no cutting index is found (index is equal to 0).

To ensure the flux correction algorithm [**4**], which is performed at the fine-coarse interface, works correctly and easily, it is required that the grids is *properly nested* in the strict sense that, except at physical boundaries, the level l grids are large enough to guarantee that there is a border at least one level l cell wide surrounding each level $l + 1$ grid. Although all the flagged points are inside the coarse grids, the new finer grids after clustering may not be properly nested in the coarse grids. This is because some unflagged points are added to the patches during clustering, and those points may not be inside the coarse grids. Since the clustering procedure can start from any level that is less than maximum level. If the clustering starts from level $l > 1$, the new grids generated at level $l + 1$ may not be properly nested in level l in a strict sense, because the flagged cells may reach the boundaries of the level l after a few time steps when level l grids remain the same during that period. To avoid this problem, the coarse grids must have enough buffer zones. In our refinement, we add different number of buffer zones for different levels: the coarser levels have more buffer zones than the finer levels.

For memory allocation and parallel problems, we may wish to produce clusters of no greater than a specified size. Therefore any cluster larger than that is bisected along the longest axis (the mean-cut is performed here). Therefore, if a patch already produced satisfies the threshold condition, it may need further cutting to ensure the proper nesting and/or proper size.

The threshold value does affect the efficiency of the algorithm. The bigger the threshold, the fewer unwanted points are involved, but usually more clusters result. More clusters introduce more artificial internal boundaries which may increase the storage requirement and computation time. It is a trade-off between a large threshold and high computational efficiency. How to choose an optimal threshold is application-dependent. We have tested several problems and found that a value in a range of 0.5∼0.7 can be adopted.

Almost all of the AMR implementations we have known so far use a fixed threshold (≥ 0.5) during the clustering procedure. We find that a constant threshold usually results in small patches when the depth of the recursion is large. Every patch formed during the clustering requires both internal and ghost boundary cells. In many small patches, the number of internal cells is even smaller than the number of ghost cells. These small patches increases the communication time between different patches and reduce the computational efficiency, especially on computers with vector processing units.

These small patches can be eliminated by post processing the selection decisions after the clustering. However, we have been unable to devise a robust post processing optimization that is not problem-specific. We observed that dynamically reducing the threshold on the recursion could improve performance. Since a small patch is produced only when the depth of the recursion is large, if we decrease the threshold when the depth of the recursion increases, the recursion will stop and a relatively big patch will be produced. In our implementation, we decrease the threshold by 10% for each depth of the recursion. The threshold at the start of the recursion can be large, such as 0.9 or 0.8. This approach does not add many unflagged points to the flagged regions, because the patch becomes smaller when the recursive depth increases.

To further reduce the number of the patches in a level, we also merge any two patches that have a common edge. The new patches must have proper size that is no great than the maximum size in any direction.

Our clustering algorithm is optimized for serial computation or parallel computation with distributed memory and does not take into consideration the advantages of long vector operations when there are vector processors. On vector machines there may be significant advantages to modifying the shape of the patches to maximize the vector length.

4. Conservative AMR on Spherical and Cylindrical grid

The conservative interpolation includes prolongation and restriction. In this section, we describe the conservative prolongation and restriction operations for cell-centered variables. The operations for face-centered variables can be done in a similar way. We assume that the cell-center variables have the cell-average values over the whole cell. We should pointed out that our approach is different from that in [**10**].

4.1. Restriction. We first describe the restriction operation. The restriction should make a volume weighted average of a variable and assign it to the corresponding cell of the coarse mesh. For Cartesian grid, it can be done by simple arithmetic averaging over all of the fine cells because each fine cell has the same volume. For cylindrical and spherical grid, it may not be true that each fine cell has the same volume. The volume of a cylindrical cell is

$$\int_{z_1}^{z_2} dz \int_{r_1}^{r_2} dr \int_{\theta_1}^{\theta_2} r d\theta = \frac{1}{2}(r_2^2 - r_1^2)\Delta\theta\Delta z.$$

The volume of a spherical cell is

$$\int_{r_1}^{r_2} dr \int_{\theta_1}^{\theta_2} r d\theta \int_{\phi_1}^{\phi_2} r \sin\theta d\phi = \frac{1}{3}(r_2^3 - r_1^3)(\cos\theta_1 - \cos\theta_2)\Delta\phi.$$

It is easy to see that the volume depends on r for a cylindrical grid, and depends on r and θ for a spherical grid. The volume-weighted average value is no longer a simple arithmetic average.

Consider a reduced 2-D (r, θ) problem. We can drop Δz or $\Delta\phi$ in the volume formula. If the refinement ratio is 2 and the values on the fine grid is $u_{f_{i,j}}$, the volume-weighted average value for the cylindrical grid is then

$$u_c = \frac{(u_{f_{0,0}} + u_{f_{0,1}})dA_0 + (u_{f_{1,0}} + u_{f_{1,1}})dA_1}{dA_c},$$

where $dA_0 = (r_1^2 - r_0^2)$, $dA_1 = (r_2^2 - r_1^2)$, and $dA_c = (r_2^2 - r_0^2)$.

The volume average value for the spherical grid is

$$u_c = \frac{u_{f_{0,0}} dA_{0,0} + u_{f_{0,1}} dA_{0,1} + u_{f_{1,0}} dA_{1,0} + u_{f_{1,1}} dA_{1,1}}{dA_c},$$

where

$$\begin{aligned}
dA_{0,0} &= (r_1^3 - r_0^3)(\cos\theta_0 - \cos\theta_1),\\
dA_{0,1} &= (r_1^3 - r_0^3)(\cos\theta_1 - \cos\theta_2),\\
dA_{1,0} &= (r_2^3 - r_1^3)(\cos\theta_0 - \cos\theta_1),\\
dA_{1,1} &= (r_2^3 - r_1^3)(\cos\theta_1 - \cos\theta_2),\\
dA_c &= (r_2^3 - r_0^3)(\cos\theta_0 - \cos\theta_2).
\end{aligned}$$

4.2. Prolongation. The prolongation is far more intricate than the restriction problem. The difficulty is to find a conservative interpolation. We here study only the monotone-preserved linear interpolation. For a reduced 2-D (r, θ) problem. The interpolation on a coarse cell $[r_1, r_2] \times [\theta_1, \theta_2]$ can be formulated as

$$u(r, \theta) = u_c + \Delta_r u_c (r - r_c) + \Delta_\theta u_c (\theta - \theta_c), \tag{4.1}$$

where Δ_r and Δ_θ are limited slopes in corresponding r and θ direction, (r_c, θ_c) is the location of the cell center.

It is naive to think that the cell center locates at $\frac{1}{2}(r_1 + r_2), \frac{1}{2}(\theta_1 + \theta_2)$ for cylindrical and spherical grid. By definition, the cell center should satisfy the following condition

$$\int_{r_1}^{r_2} dr \int_{\theta_1}^{\theta_2} u(r, \theta) r dr d\theta = u_c \cdot dA_c, \tag{4.2}$$

for the cylindrical grid, and

$$\int_{r_1}^{r_2} \int_{\theta_1}^{\theta_2} u(r, \theta) r^2 \sin\theta dr d\theta = u_c \cdot dA_c, \tag{4.3}$$

for the spherical grid.

For a cylindrical grid, we can obtain the location of the cell center by inserting Eq. (4.1) into (4.2), which yields after simple manipulation,

$$r_c = \frac{2}{3}\frac{(r_2^3 - r_1^3)}{(r_2^2 - r_1^2)}, \tag{4.4}$$

$$\theta_c = \frac{1}{2}(\theta_2 + \theta_1). \tag{4.5}$$

For a spherical grid, the location of the cell center can be obtained by inserting Eq. (4.1) into (4.3),

$$r_c = \frac{3}{4}\frac{(r_2^4 - r_1^4)}{(r_2^3 - r_1^3)} \tag{4.6}$$

$$\theta_c = \frac{\theta_1 \cos\theta_1 - \theta_2 \cos\theta_2 + \sin\theta_2 - \sin\theta_1}{\cos\theta_1 - \cos\theta_2}. \tag{4.7}$$

The locations of the cell-center for the fine grid can be calculated in the same way. Then the cell-average values for the fine grid are calculated by (4.1). We have verified that the cell-centered values obtained by (4.1) with r_c and θ_c calculated by (4.4)–(4.7) for find grid cells satisfy the conservation laws.

5. Maintaining Divergence Free Conditions for AMR

We first illustrate our algorithm using a simple 2D Cartesian grid, (The generalization to 3D in Cartesian is straightforward.) then move on to implementations in cylindrical and spherical grids. To simplify our description, we assume that, for all geometries considered, each cell edge is split into equal-distance pieces according to the refinement ratio.

5.1. Cartesian Grid. For the Cartesian grid, the face area is proportional to the local grid spacing in each direction. This is an important feature that can greatly simplify our algorithm. Later we will see that other types of orthogonal grids do not have this feature. For the ease of description, we will use a 2D example. Similar procedure in 3D can be deduced relatively easily.

Our approach is illustrated in sequence by plots (a), (b), (c), (d), and (e) in Fig. 5.2 with the following steps:

Step 1: We use a directional splitting approach to treat multi-dimensional problems. We first consider the refinement along the x-direction as shown in subplot (b) of Fig. 5.2. Obtaining boundary values $v_1, ..., v_6$ is the same as in Balsara's approach described above.

Step 2: Given the quantities on the edges in (b), we can now partially construct the internal fine face values based on the divergence-free condition. For example, in subplot (c), u_{c2} is uniquely defined by u_{c0}, v_1, v_2 as

$$u_{c2} = u_{c0} + (v_2 - v_1)\frac{\delta x}{\Delta y},$$

where δx and Δy are the spacings for the fine and coarse cells respectively. The quantity u_{c3} can be calculated in the similar manner. Then it is straight forward to show that the divergence-free condition is also preserved in the last cell by noting that

$$\frac{u_{c1} - u_{c3}}{\delta x} + \frac{v_6 - v_5}{\Delta y} = 0 \ ,$$

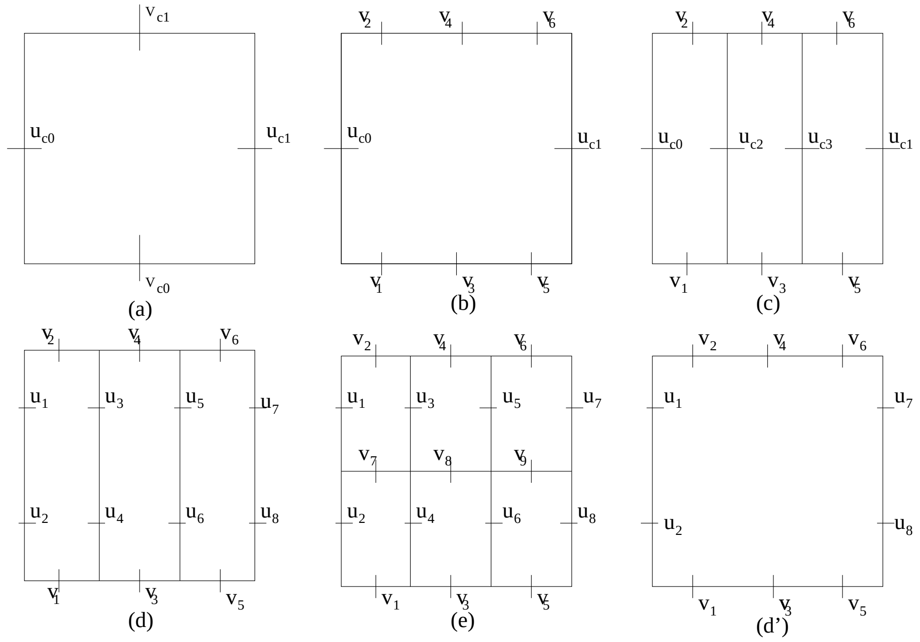

FIGURE 5.2. Directional-splitting approach for reconstruction of the divergence-free fields on a refined grid.

if

$$\frac{u_{c1} - u_{c0}}{\Delta x} + \frac{\frac{1}{3}(v_2 + v_4 + v_6) - \frac{1}{3}(v_1 + v_3 + v_5)}{\Delta y} = 0 \ .$$

Step 3: We now consider the refinement along the y-direction, which is taken as two in this example. As shown in (d), quantities u_1, u_2, u_7 and u_8 can be obtained similar to Step (1). In order to obtain quantities u_3 and u_4, we first obtain the slope along the $y-$direction at u_{c2}. This limited slope can be obtained by interpolation from the known values at the left and right faces of the coarse cell u_{c0} and u_{c1}, which yields,

$$\frac{du_{c2}}{dy} = \frac{1}{3}\frac{du_{c1}}{dy} + \frac{2}{3}\frac{du_{c0}}{dy} \ . \tag{5.1}$$

Quantities u_3 and u_4 can then be calculated from u_{c2} and du_{c2}/dy. This step can be repeated to obtain u_5 and u_6.

Step 4: Quantities v_7, v_8 and v_9 can be calculated using the divergence-free condition within its respective, refined cell. This is shown in (e).

Even though we have only demonstrated our algorithm in 2D, its extension to 3D should not be difficult. The interpolation on the coarse faces can be done using higher order reconstruction as well, though it should be conservative in a finite-volume sense.

We have made detailed comparisons between our method and that by Balsara for using a refinement ratio of 2 in each direction. If the fine face values on the coarse faces are constructed in the same way and the limited slope for the intermediate coarse face is obtained via Eq. (5.1), we obtain exactly the same values for the internal fine faces for both 2D and 3D cases as those given by Balsara's method.

This also implies that our reconstruction method has the same TVD preserving property as his method, which was discussed in [**2**].

The divergence-free prolongation for 3D is different from the reconstruction in Balsara [**1**]. The reason is that the face values of the new fine grid may come from the old fine grid, and a linear profile is not enough. Balsara [**1**] introduced complicated interpolation polynomials to match the known fine face values. Our results are different from Balsara's approach [**1**]. We may not have a closed-form expressions for the prolongation operation on a whole coarse cell. However, whenever the interpolation is needed, monotonicity-preserving linear or high-order interpolation in 1-D is used. It is efficient and TVD-preserving. We acknowledge that our approach may not be as smooth as Balsara's approach within a coarse cell.

For finite-difference and finite-volume methods on an AMR grid, a number of ghost cells are usually required at the boundary. When the number of ghost cells is not divisible by the refinement ratio, special care is needed in order to maintain the divergence-free condition in those cells. Instead, we use a virtual extended grid, where several additional zones are added. For the virtual fine grid, the number of ghost cells is divisible by the refinement ratio so that if a coarse face is next to a fine grid, it is covered wholly by the fine faces. This virtual fine grid is used only to obtain the values of fine grid whenever the grid is re-adapted. It is not used in integration.

5.2. Cylindrical and Spherical Grids. We now discuss how to implement our method in other orthogonal grids such as cylindrical and spherical coordinates with AMR. Balsara [**2**] has studied this issue recently as well by extending his divergence-free reconstruction of [**1**] to the cylindrical and spherical meshes. As described in [**2**], a variable substitution and coordinate transformation approach is used to transform the divergence formula of the cylindrical and spherical geometry to the standard divergence formula of the Cartesian grid. Then the reconstruction for the Cartesian grid is applied to the new variables on the new coordinates.

We use (r, z, ϕ) and (r, θ, ϕ) to represent the cylindrical and spherical coordinates, respectively. The divergence of a vector field $\mathbf{v}$ becomes

$$\nabla \cdot \mathbf{v} = \frac{1}{r}\left(\frac{\partial(rv_r)}{\partial r} + \frac{r\partial v_z}{\partial z} + \frac{\partial v_\phi}{\partial \phi}\right), \tag{5.2}$$

for the cylindrical grid, and

$$\nabla \cdot \mathbf{v} = \frac{\partial(r^2 v_r)}{r^2 \partial r} + \frac{\partial(\sin\theta v_\theta)}{r\partial\theta} + \frac{\partial v_\phi}{r\sin\theta\partial\phi}, \tag{5.3}$$

for the spherical grid.

The main complication in implementing AMR in these geometries is to determine where the cell centers or face centers are, because they are all weighted by additional factors such as $1/r$ and $\sin\theta$. In our method we assume that the edges of a grid in any direction are split into equal-distance pieces according to the refinement ratio. This typically means that the coarse cell is not divided into equal area/volume pieces.

We will describe the algorithm only for the cylindrical grid. The implementation for spherical grid can be done accordingly. The finite-volume discretization of

Eq. (5.2) is

$$(\nabla \cdot \mathbf{v})_{i,j,k} = \frac{(r_i v_{r_i} - r_{i-1} v_{r_{i-1}})}{r_{i-\frac{1}{2}} dr} + \frac{(v_{z_j} - v_{z_{j-1}})}{dz} + \frac{(v_{\phi_k} - v_{\phi_{k-1}})}{r_{i-\frac{1}{2}} d\phi},$$

where $r_{i-\frac{1}{2}} = \frac{1}{2}(r_i + r_{i-1})$.

As described in section 5.1, the reconstruction for fine cells inside a whole coarse cell begins with reconstruction on each coarse face. It can be verified that the interpolation scheme for the Cartesian grid can be applied to faces (z, ϕ) for the v_r component and (r, z) for the v_ϕ component, but not to face (r, ϕ) for the v_z component. To construct a conservative interpolation on the face of (r, ϕ), we need to find the slopes at various intermediate points. To do this, we use a linear profile defined by

$$v_z = v_{zc} + v_{z,\phi}(\phi - \phi_0) + v_{z,r}(r - r_0), \tag{5.4}$$

where $\ldots_{,\phi,r}$ stands for $\partial/\partial\phi$ or $\partial/\partial r$, $v_{z,\phi}$ and $v_{z,r}$ are the limited slopes, and (r_0, ϕ_0) is the face center. The quantity v_{zc} is the area-averaged value of v_z on the coarse face $[r_{i-1}, r_i] \times [\phi_{k-1}, \theta_k]$, which is defined by

$$v_{zc} = \frac{1}{\int\int r dr d\phi} \int_{r_{i-1}}^{r_i} \int_{\phi_{k-1}}^{\phi_k} v_z r dr d\phi. \tag{5.5}$$

Inserting (5.4) into (5.5) and simplifying it, we obtain

$$v_{z,\phi}(\phi_{k-\frac{1}{2}} - \phi_0) + v_{z,r} \left(\frac{2(r_i^3 - r_{i-1}^3)}{3(r_i^2 - r_{i-1}^2)} - r_0 \right) = 0. \tag{5.6}$$

To satisfy Eq. (5.6), we let

$$\phi_0 = \phi_{k-\frac{1}{2}} = \frac{1}{2}(\phi_k + \phi_{k-1}) \tag{5.7}$$

$$r_0 = \frac{2(r_i^3 - r_{i-1}^3)}{3(r_i^2 - r_{i-1}^2)}. \tag{5.8}$$

We should mention that the expression for ϕ_0 and r_0 are also valid for faces on a fine grid. Therefore, when the mesh is refined with ratio m, face centers for the fine mesh can be calculated in the same fashion. Let $\delta r = (r_i - r_{i-1})/m$ and $r = r_i$, then the face centers for the fine grid will be

$$r_{0l} = \frac{2[(r - l\delta r)^3 - (r - (l+1)\delta r)^3]}{3[(r - l\delta r)^2 - (r - (l+1)\delta r)^2]}, \quad l = 0, 1, \ldots, m-1.$$

After all the face centers are calculated, the vector field components can be calculated by the linear profile (5.4), for example, in r-direction,

$$v_{z_{0l,k}} = v_{zc} + v_{z,r}(r_{0l} - r_0).$$

If we denote the area for the fine and coarse faces by

$$dA_{0l} = \frac{1}{2}(r_{0(l+l)}^2 - r_{0l}^2) d\phi, \qquad dA_c = \frac{1}{2}(r_k^2 - r_{k-1}^2) d\phi,$$

then it can be verified that

$$\sum_{l=0}^{m-1} v_{z_{0l,k}} \cdot dA_{0l} = v_{zc} \cdot dA_c,$$

which means the flux is conserved after interpolation.

After the face reconstruction is finished, the algorithm described in section 5.1 can be used to reconstruct the internal faces step by step.

6. Numerical Experiment

In this section, we present several 2D numerical examples in Cartesian, cylindrical and spherical geometries. For the MHD examples, magnetic field components are defined at the edge/face centers while all other fluid quantities are defined at the cell centers. A second order Roe's Riemann solver is used to advance the conservative variables. The detail of our implementation can be found in [**9**]).

6.1. Clustering Examples. The first example is a scalar version of 2D Burgers' equation

$$u_t + uu_x + uu_y = 0.005(u_{xx} + u_{yy}), \tag{6.1}$$

with an analytic solution

$$u(x, y, t) = \frac{1}{1 + \exp(x + y - t)/(2r)}.$$

The solution is a steep wave front propagating to the right top corner. The flagged points are distributed along the diagonal. We compare the results for a variable threshold (start with 0.8) and a fixed threshold (0.7) in our clustering algorithm. We use two refinement levels. The total number of points for the finest level with the variable threshold increases about 10% compared with the fixed threshold (0.7) method. However, the total number of patches for the level decreases from 78 to 18.

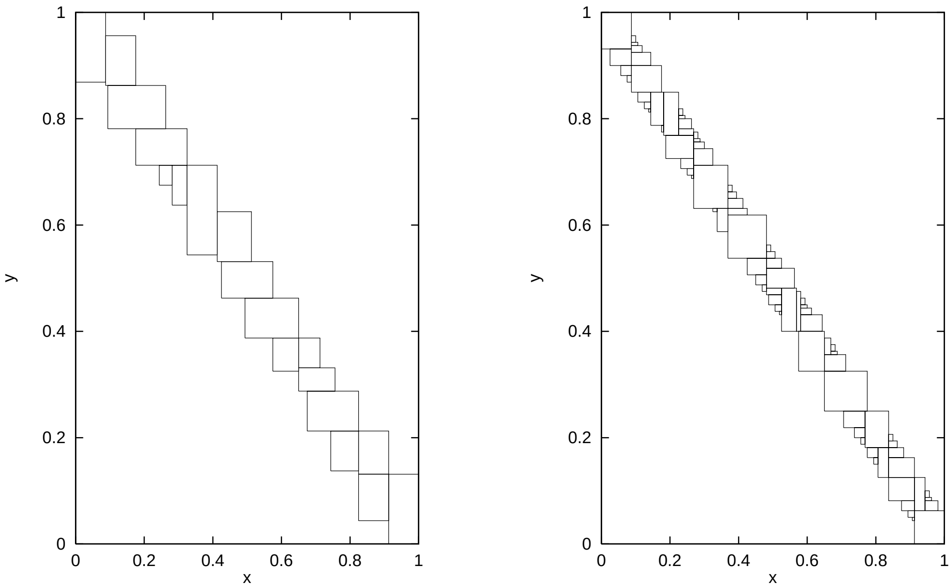

FIGURE 6.1. Clusters of the finest level at t=1.0. (Left): with a variable threshold (start with 0.8), a total of 18 clusters are generated. (Right): with a fixed threshold (0.7); a total of 78 clusters are generated.

6.2. MHD Examples for cylindrical geometry. We tested our algorithm for cylindrical geometry with two examples. The first one is taken from [**3**]. There is a dense rotating disk of fluid with density 10, angular velocity 20, and radius 0.1. The ambient fluid is at rest with density 1. The initial magnetic field is uniform with $B_x = 5/\sqrt{4\pi}$ and $B_y = 0$.

We first solved this example in a Cartesian grid with domain $[0, 1] \times [0, 1]$, base grid 100×100 and three-level refinement with ratios of 3 and 2. The results are shown in Fig.6.2. The divergence-free condition is preserved very well.

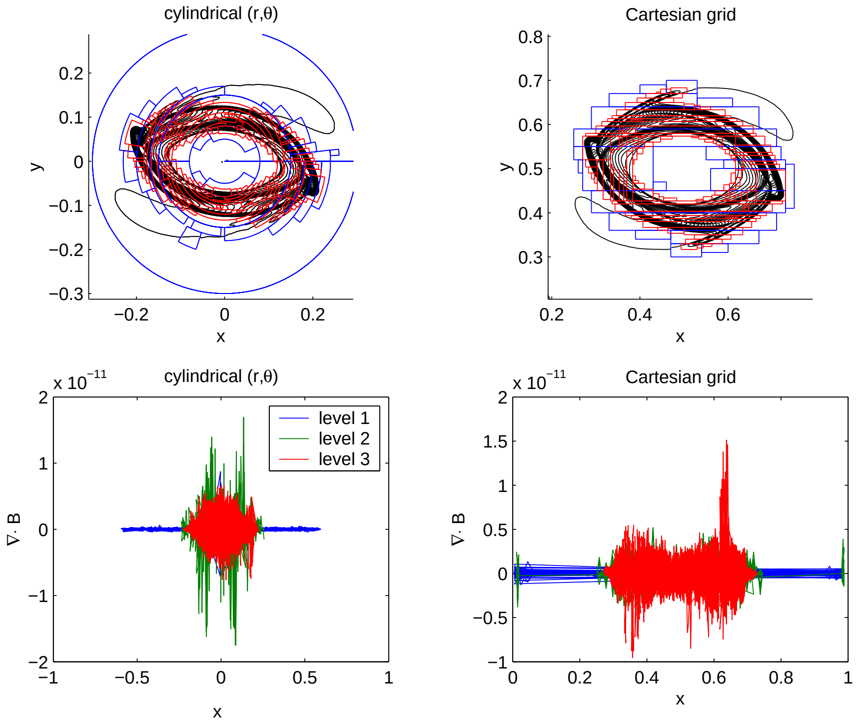

FIGURE 6.2. The results for rotor problem at $t = 0.295$. The refinement ratios are 3 and 2 respectively. 30 contour lines between 0.532 and 10.83 are used.

We also solved this example as a a reduced 2-D problem in (r, ϕ) with a disk domain $[0, 0.6] \times [0, 2\pi]$, where x is along $\phi = 0$ direction. Converting B_x and B_y into cylindrical coordinates, we have

$$B_r = B_x \cos\phi, \quad B_\phi = -B_x \sin\phi, \tag{6.2}$$

which is not divergence-free in our finite-volume discretization. Therefore, we adopted an approximate initialization for B_r, which is

$$B_r = B_x \frac{\sin\phi_{j+1} - \sin\phi_j}{d\phi}.$$

We use three refinement levels for this example. The refinement ratio is 3 and 2, respectively. The base grid is 60×60. As suggested in [**3**], we output the solution at time $t = 0.295$. Fig. 6.2 shows the results. The divergence of the magnetic field can be kept around the machine round-off error. We also solved this example with a uniform grid that is corresponding to the finest-level of the AMR mesh. The CPU time for the uniform grid is 12 times more than that for the AMR grid.

The second example is in the (r, z) coordinates. It was introduced in [**11**]. The problem is simulating a light cylindrical MHD jet with a top-hat velocity profile propagating through a background medium. We used a computation domain $(r, z) = [0, 1] \times [0, 2]$. The base grid is 200×200. The jet has a radius $r = 0.125$ initially, which is about 25 base grid cells. The jet has an initial Mach number of 20, a gas density contrast between the jet and ambient medium of $\rho_{jet}/\rho_{\mathrm{amb}} = 0.1$. The ambient medium has a sound speed of 1, and an initial ambient magnetic field of $B_\phi = B_r = 0$, and $B_z = B_{\mathrm{amb}} = 0.1$. The jet carries a helical magnetic field with $B_r = 0, B_\phi = 2B_{\mathrm{amb}}(r/r_{jet})$, and $B_z = B_{\mathrm{amb}}$.

We again used three levels of refinement with refinement ratios of 3 and 2. We ran our simulation until $t = 0.1$. Fig. 6.3 shows the density contour plot with refinement and the results of $\nabla \cdot \mathbf{B}$.

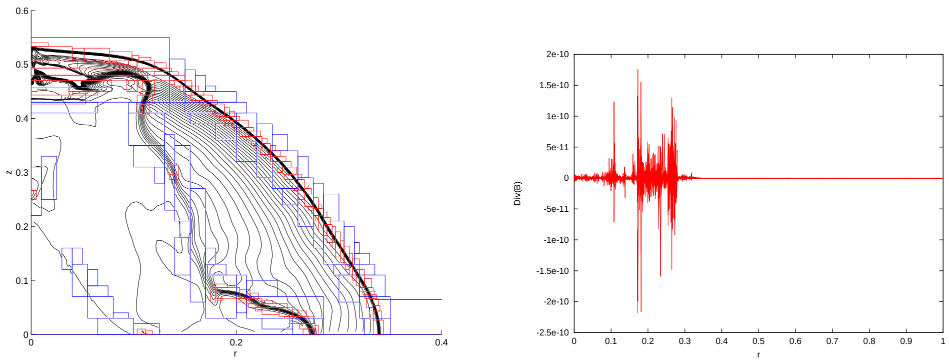

FIGURE 6.3. The "zoomed" version of refinement and density contour plot for the jet problem on cylindrical (r, z) plane, and $\nabla \cdot \mathbf{B}$ at each r. Values of $\nabla \cdot \mathbf{B}$ at different z are plotted.

6.3. HD and MHD Example for spherical geometry. The last example is on simulating a rising buoyant bubble in an isothermal background gas, which is initially in a pressure equilibrium under an external gravitational field. Its density (and pressure) profile is described as $\rho = p = \exp(-r)$. A spherical bubble with a radius 0.5 is initially located at $(1.1, 0)$. At $t = 0$, the density inside the bubble is taken as $\rho = 0.1 \exp(-r)$, so that the bubble will rise due to the pressure imbalance. (Some details are further described in [**9**]).

The computational domain in (r, θ) is $[0.2, 3.8] \times [0, \pi/3]$. We first simulated this problem without any external magnetic field, i.e., $B_r = B_\theta = B_\phi = 0$, and observed the Rayleigh-Taylor instability at the contact interface (see Fig. 6.4). We then used a uniform magnetic field along the $z(= r\cos\theta)$ direction. It is a potential field with magnitude $F = -b \cdot r \cos\theta$. The magnetic field is defined by

$$\mathbf{B} = -\nabla F = (b \cos\theta, -b \sin\theta),$$

where the constant b controls the magnitude of the magnetic field. We simulated this problem with $b = 0.4$. The Rayleigh-Taylor instability at the contact interface is suppressed by the strong magnetic field (see Fig. 6.4).

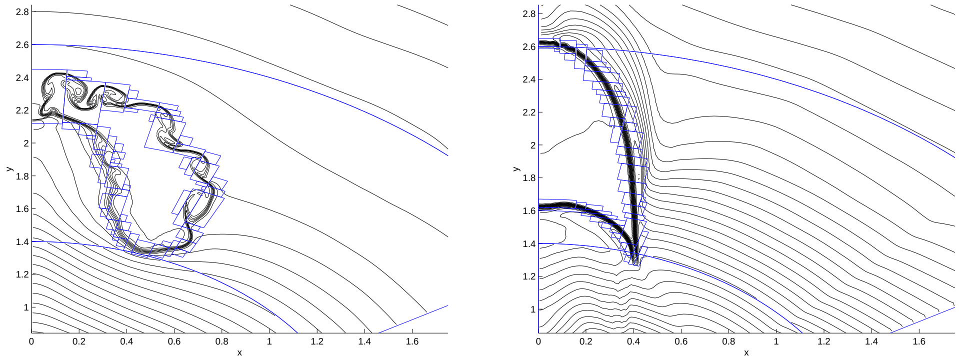

FIGURE 6.4. The density contour plot with refinement for bubble problem with no magnetic field (left) and with strong magnetic field $b = 0.4$ (right).

We used a base grid of 360×120 and a refinement ratio of 3. The refinements and divergence of the magnetic field at different times are shown in Fig. 6.5.

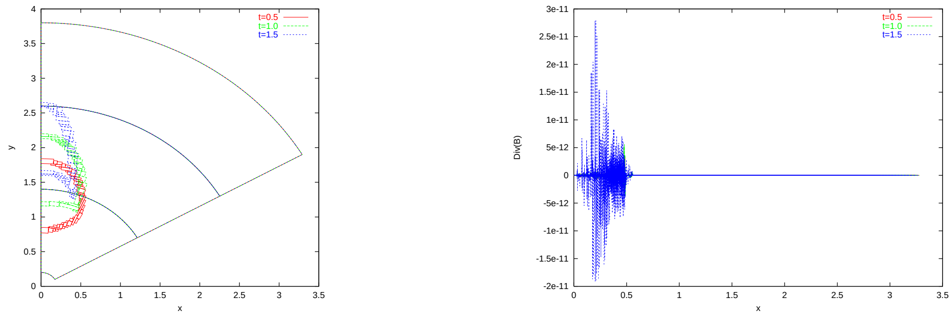

FIGURE 6.5. The refinement and $\nabla \cdot \mathbf{B}$ at different times in an axis-symmetric spherical grid. Values of $\nabla\cdot\mathbf{B}$ at all cells and three different times are plotted.

References

1. D. S. Balsara, *Divergence-free adaptive mesh refinement for magneto-hydrodynamics*, J. Comput. Phys. **174** (2001), 614–648.
2. ______, *Second order accurate schemes for magnetohydrodynamics with divergence-free reconstruction*, preprint, astro-ph/0308249, 2003.
3. D. S. Balsara and D. S. Spicer, *A staggered mesh algorithm using high order Godunov fluxes to ensure solenoidal magnetic fields in magnetohydrodynamics simulations*, J. Comput. Phys. **149** (1999), 270–292.
4. M. J. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys. **82** (1989), 64–84.
5. M. J. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys. **53** (1984), 484–512.

6. M. J. Berger and I. Rigoutsos, *An algorithm for point clustering and grid generation*, IEEE Trans. on Systems, Man, and Cybernetics **21** (1991).
7. J. U. Brackbill and D. C. Barnes, *The effect of nonzero* $\nabla \cdot \mathbf{b}$ *on the numerical solution of the magnetohydrodynamic equations*, J. Comput. Phys. **35** (1980), 426.
8. J. M. Hyman and S. Li, *Solution adapted nested grid refinement for 2-D PDEs*, Tech. Report LA-UR-98-5463, Los Alamos National Lab., 1998.
9. S. Li and H. Li, *A modern code for solving magnetohydrodynamics or hydrodynamic equations*, Tech. report, Los Alamos National Laboratory, 2003.
10. R.B. Pember, J.A. Greenough, and P. Colella, *An adaptive higher-order godunov method for gas dynamics in three dimensional orthogonal curvilinear coordinates*, Tech. Report UCRL-JC-123351, Lawrence Livermore National Laboratory, 1996.
11. D. Ryu, F. Miniati, T. W. Jones, and A. Frank, *A divergence-free upwinding code for multi-dimensional MHD flows*, Astrophysics. J. **509** (1998), 244–255.
12. G. Tóth and P. L. Roe, *Divergence- and curl-preserving prolongation and restriction formulas*, J. Comt. Phys. **180** (2002), 736–750.

Theoretical Division, MS B284, Los Alamos National Laboratory, Los Alamos, NM 87545

E-mail address: `sli@lanl.gov`