Journal of Computational Physics 262 (2014) 131–152

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Adaptive multi-resolution method for compressible multi-phase flows with sharp interface model and pyramid data structure

L.H. Han *, X.Y. Hu, N.A. Adams

Institute of Aerodynamics and Fluid Mechanics, Technische Universität München, 85748 Garching, Germany

ARTICLE INFO

Article history:
Received 24 April 2013
Received in revised form 31 October 2013
Accepted 31 December 2013
Available online 9 January 2014

Keywords:
Multi-resolution
Multi-phase flows
Sharp interface methods
Data structure
Parallel computing

ABSTRACT

In this work, we present a block-based multi-resolution method coupled with a sharp interface model (MR-SIM) for the high-resolution simulation of multi-phase flows, where data structure and adaptive multi-resolution approach are tailored to achieve high computational efficiency. The method updates the dynamic topological data structure according to two separate procedures: (i) tracking of the interface position, and (ii) MR analysis on each individual phase. High efficiency is achieved by employing a storage-and-operation-splitting pyramid data structure, in which any two adjacent blocks partially overlap while the overlapping parts share the same data in memory. The non-overlapping data are distributed into fine-grained data packages and stored within a memory pool. The proposed narrow-band technique for the level-set-based interface method also increases the computational and memory efficiency greatly by restricting all interface-relevant data and operations to a neighborhood of the interface. A broad set of test simulations is carried out to demonstrate the potential and performance of the MR-SIM approach.

1. Introduction

To study the dynamics of compressible flows, various numerical methods have been developed to solve the Euler equations or the Navier–Stokes equations. For the solution of these equations straightforward discretization schemes were employed over large parts of the computational domain, except for regions with steep gradients, shocks or discontinuities. Capturing sharp changes in the solution requires the use of nonlinear discretization schemes, such as WENO5 [1] or WENO-CU6 [2], that do not introduce spurious oscillations. Since this kind of schemes often contain specific stencils consisting of many evenly-distributed points, a common and simple way to apply them is to solve the governing equations on uniform grids. In general, the solution exhibits a wide range of spatial and temporal scales, and typically these scales are not uniformly distributed in the space–time domain. As their distribution is not known a priori, adaptive approaches, such as adaptive mesh refinement [3–6] and wavelet-based multi-resolution methods [7–11], have been introduced for adjusting grid resolution locally and dynamically according to error estimates from different resolution levels. The main differences between adaptive multi-resolution (MR) methods and adaptive mesh refinement (AMR) methods are in the way the adaptive grids are stored and in the error-estimation approaches. Deiterding et al. [12] found that multi-resolution techniques yield improved memory compression and CPU-time reduction when compared to adaptive mesh refinement methods. Within the framework of MR methods, wavelets have been implemented successfully for the simulation of conservation laws in computational fluid dynamics [13]. Based on the details, or wavelet coefficients, between two consecutive grid-refinement

* Corresponding author.
E-mail addresses: Luhui.Han@tum.de (L.H. Han), Xiangyu.Hu@tum.de (X.Y. Hu), Nikolaus.Adams@tum.de (N.A. Adams).

0021-9991/$ – see front matter
http://dx.doi.org/10.1016/j.jcp.2013.12.061

levels, MR methods provide a rigorous regularity analysis [14,15], while for AMR methods rigorous error estimators are not available [12]. By matching spatial and temporal scales for each discrete cell, Osher and Sanders [16] and Dawson and Kirby [17] introduced local scale-dependent time-stepping integration schemes (LTS) to obtain additional speed-up. In such space–time adaptive methods [18,19], the size of each cell and its time step are adjusted dynamically. However, such a data-compression strategy leads to a large amount of sequential logical operations, which is incompatible with efficient parallel computing requirements.

To find the balance between adaptive algorithms and parallel computing, a new method based on wavelet-adapted blocks with local time-stepping on multi-core architectures has been proposed in [20]. Instead of using a single grid cell in the dynamic graded tree data structure, grid blocks with a predefined number of computational cells were introduced similarly as in [6]. Although the granularity is increased at the expense of compression rate, it significantly decreases the amount of sequential operations and improves parallelism. To make sure that high-order interpolation schemes can be employed for flux computations at the block boundaries where the grid resolution varies, auxiliary boundary cell data are temporarily reconstructed during each evolution. To avoid the temporary memory allocation and deallocation in every block at each time step, in [21] memory was permanently allocated to store boundary cell data for all grid blocks, which finally increased the parallelism to an ever higher level with the help of a multi-step boundary rebuilding approach. However, partially overlapping blocks result in a severe memory inefficiency, especially for three dimensional simulations.

Until recently, multi-resolution methods are mainly employed for single-phase problems (e.g. Cohen et al. [9], Han et al. [21]). Multi-phase flows have been recently addressed with smooth-interface models (Hejazialhosseini et al. [20], Rossinelli et al. [22]). For problems with immiscible material interfaces a smooth-interface model can result in an inaccurate interface evolution that may lead to numerical instabilities. Accurate interface evolution and numerically stable solution can be obtained by sharp-interface models, such as those based on level-set [23–25] and front-tracking methods (Glimm et al. [26]). Since ghost cell data are required for such approaches, duplication of memory storage in the whole computational domain is common, although only the ghost-cell data near the interface are useful.

The objective of the present paper is to develop a generalized block-based multi-resolution method with a sharp interface model (MR-SIM) for multi-phase simulations. Previous MR methods organize their data with tree structures to achieve high rates of data compression. However, frequent indirect searches required to access a particular node except for parent and child seriously limits parallel efficiency, especially for a tree with many levels. In this work, we employ a pyramid data structure instead, which is inspired by Google's mapping system [27] and allows direct search among all blocks. However, unlike their choice of non-overlapping blocks (tiles in [27]) as the basic parallel units, we use partially overlapping ones to obtain better parallelism. For remedying limited memory efficiency of approaches in [6,21], our present data structure adopts storage-and-operation-splitting techniques to separate a block from its associated computational cell data. Relative to blocks, cells are finer-grained into data packages and stored in a memory pool [28]. Since packages do not overlap with each other, there is no extra memory required for the data associated with overlapping regions among adjacent blocks, and inter-block communications [21] are achieved implicitly by shared memory. To be able to track the interface as accurately as possible, we update the adaptive data structure first by the actual interface position to ensure all near-interface regions required by the sharp-interface model are refined to the finest grid level. Second, a standard multi-resolution analysis is performed successively from the coarsest to the finest level to complete the final adapted block distribution. As from one time step to the next the interface needs to be tracked only in a narrow band near the actual interface at the finest level, it is not necessary to save ghost cell data at the other levels. Therefore, we employ a narrow-band technique to restrict all interface-relevant data and operations to a neighborhood of the material interface. As a result, MR-SIM requires only an amount of memory which is similar to smooth-interface models and increases computational efficiency largely.

The paper is structured as follows: In Section 2 we review the conservative sharp interface model for multi-phase compressible flows, and adaptive MR schemes. A storage-and-operation-splitting pyramid data structure is proposed in Section 3 to improve the computational efficiency. Section 4 introduces an extension of the current MR approach with sharp-interface models. A number of tests are carried out in Section 5, which demonstrate accuracy and high performance of the new MR-SIM approach.

2. Preliminaries

2.1. Conservative sharp interface method

In this work, we employ the conservative sharp interface method to solve multi-phase flows. Assuming the fluid to be inviscid and compressible, the governing equations can be written as a system of conservation laws

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = 0. \tag{1}$$

Here, $\mathbf{U} = (\rho, \rho\mathbf{v}, E)^T$ is the density of conserved quantities of mass, momentum and total energy with relation $E = \rho e + \frac{1}{2}\rho|\mathbf{v}|^2$, where e is internal energy per unit mass. $\mathbf{F}(\mathbf{U})$ denotes the corresponding flux functions. To close this set of equations it is necessary to specify equations of state (EOS) for each phase.

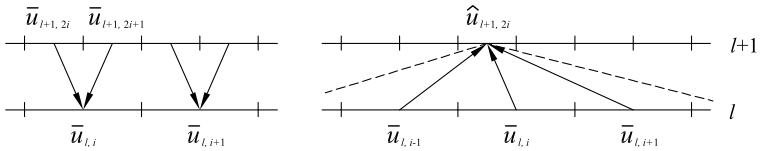

Fig. 1. Projection operator $P_{l+1\to l}$ (left) and prediction operator $P_{l\to l+1}$ (right) in 1D. $\bar{u}_{l,i}$ and $\hat{u}_{l,i}$ represent the exact and predicted cell-average of cell i at level l, respectively.

Considering computational cells being cut by an interface, a typical discretization with explicit first-order forward time difference (as basic building block for the higher-order Runge–Kutta method) of Eq. (1) for each considered phase, on a two-dimensional Cartesian grid with grid spacings Δx and Δy can be written as

$$\alpha_{i,j}^{n+1}\mathbf{U}_{i,j}^{n+1} = \alpha_{i,j}^{n}\mathbf{U}_{i,j}^{n} + \frac{\Delta t}{\Delta x\Delta y}\hat{\mathbf{X}}(\Delta\Gamma_{i,j}) + \frac{\Delta t}{\Delta x}[A_{i-1/2,j}\hat{\mathbf{F}}_{i-1/2,j} - A_{i+1/2,j}\hat{\mathbf{F}}_{i+1/2,j}] + \frac{\Delta t}{\Delta y}[A_{i,j-1/2}\hat{\mathbf{F}}_{i,j-1/2} - A_{i,j+1/2}\hat{\mathbf{F}}_{i,j+1/2}], \tag{2}$$

where Δt is the time step size, $\alpha_{i,j}\mathbf{U}_{i,j}$, where $\alpha_{i,j}$ is volume fraction, and $\mathbf{U}_{i,j}$ are the conservative quantities in the cut cell and the cell-averaged density of conservative quantities, respectively. $A_{i-1/2,j}$, $A_{i+1/2,j}$, $A_{i,j-1/2}$ and $A_{i,j+1/2}$ are cell-face apertures. $\hat{\mathbf{F}}$ is the cell-face flux, which is obtained by high-order shock-capturing schemes, e.g., WENO schemes [1,2,29], and $\hat{\mathbf{X}}(\Delta\Gamma_{i,j})$, where $\Delta\Gamma_{i,j}$ is the interface segment within the cut cell, is the momentum and energy flux across the interface segment determined by the interface interaction. For a small cut cell or an empty cell a stable fluid state may be unobtainable based on the time step calculated according to the full grid size CFL condition. Therefore, the fluid in those cells is mixed with that of the neighboring cells with large volume fractions. The exchanges of the conservative quantities are calculated according to the averaged values in mixing operations. Then the conservative quantities for one fluid in the near interface cells are updated by

$$\alpha_{i,j}^{n+1}\mathbf{U}_{i,j}^{n+1} \leftarrow \alpha_{i,j}^{n+1}\mathbf{U}_{i,j}^{n+1} + \sum_{k}\mathbf{M}, \tag{3}$$

where the second term on the right hand side represents the sum of all mixing operations on cell $(i,\ j)$.

In order to obtain the volume fraction, interface segment and cell-aperture in Eq. (2), a signed distance function $\phi(\mathbf{x},t)$, i.e., the level set function [30,31], is employed to describe the interface. The location of the interface is given as the zero level set implicitly, and its evolution is governed by a linear advection equation of the form,

$$\frac{\partial\phi}{\partial t} + \mathbf{v}\cdot\nabla\phi = 0. \tag{4}$$

Instead of using the actual fluid velocity $\mathbf{v}$ in Eq. (4), for the conservative sharp-interface method [24], the level-set advection velocity is obtained by solving the real and ghost interface interactions [32].

The signed distance property of the level set function is often violated during the evolution of the interface, leading to inaccurate approximations of the geometrical properties of the interface. In order to maintain this regularity, we solve the re-initialization equation,

$$\frac{\partial\phi}{\partial\tau} + \mathrm{sgn}(\phi)\big(|\nabla\phi| - 1\big) = 0, \tag{5}$$

as proposed by Sussman et al. [33] until steady-state is reached.

We note that an extension of this method to 3D has already been developed by Lauer et al. [34].

2.2. Overview of MR algorithms

An adaptive MR algorithm consists of a dynamic data structure and the process of its adaptive updating. To our knowledge, almost all currently used MR methods for fluid simulations employ tree-type or hash table based [15,35] data structures. Adaptivity implies that the data structure can change dynamically. Achieving this goal relies on a multi-resolution analysis. The general principle is to represent a set of data given on a fine grid as values on a coarser grid plus a series of differences at different levels of nested dyadic grids [7]. In most Refs. [9,10,18,19,21] the differences (called details) at a level $l+1$ were described as prediction errors between exact values at level $l+1$ and predicted values from level l. For this purpose, first, exact cell-averages at the level l were calculated from the ones of level $l+1$ by using a projection operator $P_{l+1\to l}$ (Fig. 1, left). Second, a prediction operator $P_{l\to l+1}$ [10] (Fig. 1, right) was employed to predict cell-average values of level $l+1$ by using the values obtained above.

In order to maintain conservation, a straightforward choice is the cell-average multi-resolution representation by Bihari and Harten [8]. For a regular grid structure in 1D, the two operators are defined individually as

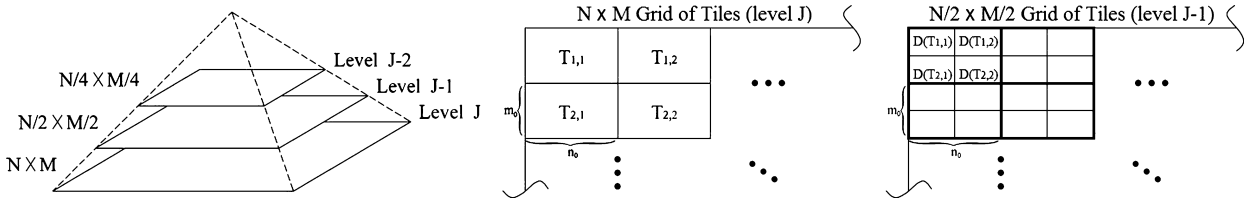

Fig. 2. Schematic of a multi-resolution pyramid data structure (left), an $N \times M$ grid of tiles at level J (middle) and an $N/2 \times M/2$ grid of tiles at level $J-1$ (right). Each tile has $n_0 \times m_0$ pixels (or computational cells in our approach). $D(\cdot)$ is a projection operator.

$$P_{l+1\to l}: \quad \bar{u}_{l,i} = \frac{1}{2}(\bar{u}_{l+1,2i} + \bar{u}_{l+1,2i+1}), \tag{6}$$

and

$$P_{l\to l+1}: \begin{cases} \hat{u}_{l+1,2i} = \bar{u}_{l,i} + \sum_{k=1}^{m} \gamma_k(\bar{u}_{l,i+k} + \bar{u}_{l,i-k}), \\ \hat{u}_{l+1,2i+1} = \bar{u}_{l,i} - \sum_{k=1}^{m} \gamma_k(\bar{u}_{l,i+k} + \bar{u}_{l,i-k}). \end{cases} \tag{7}$$

Hence, the prediction errors of cell i at level l can be estimated as

$$\bar{d}_{l,i} = \bar{u}_{l,i} - \hat{u}_{l,i}. \tag{8}$$

They are small in regions where the solution is smooth and significant close to irregularities.

In this work we adopt fifth-order interpolation ($m = 2$) and the corresponding coefficients in Eq. (7) are

$$\gamma_1 = -\frac{22}{128}, \qquad \gamma_2 = \frac{3}{128}. \tag{9}$$

Extensions of these two operators to 2D and 3D have already been developed by Bihari and Harten [8] and Domingues et al. [18] respectively.

To evolve the solution of discrete equations of Eq. (2) from $\mathbf{U}^n$ to $\mathbf{U}^{n+1}$, three basic steps are necessary: (i) refinement, (ii) evolution, and (iii) coarsening. The refinement operator **R** accounts for possible translation or creation of smaller scales in the solution between two subsequent time steps. With a tree data structure a multi-resolution analysis (MRA) is carried out on all leaves to obtain their details (a leaf denotes a node without children in an incomplete tree structure [10]). When the detail is larger in absolute value than a certain level-dependent threshold, the corresponding cell or block should be refined. We choose the level-dependent threshold

$$\epsilon_l = 2^{D_0(l-L_{max})}\epsilon, \tag{10}$$

where D_0 is the space dimension and L_{max} is the maximum level of adaptive data structure [9,10]. In order to obtain additional speed-up and accuracy from the time integration, during the evolution stage (**E**) a multi-step Runge–Kutta local time stepping scheme is often employed which applies a special rule to the leaves, see e.g. [18]. To ensure strict conservation on the full domain a conservative flux computation is adopted between cells of different levels [10]. The coarsening operator **T** is employed to remove unnecessary leaves where details have smaller magnitude than prescribed thresholds.

Note that in order to adopt high-order interpolations in the multi-resolution analysis and flux computations, before each step auxiliary boundary cell data of a node need to be temporarily constructed [20,21] from data of other nodes to ensure that all stencil elements required by the high-order schemes are available. As a node in a tree can only know its predecessor (parent node) and successors (child nodes), it needs several indirect searches to find its neighbors or other required nodes, which seriously limits parallel efficiency, especially for a tree with many levels.

3. Efficient data structure

In this section we briefly introduce the pyramid data structure of Google's mapping system [27]. Subsequently it is modified and enhanced by coupling with a new storage-and-operation-splitting approach to be suitable for our parallel MR method.

3.1. Pyramid data structure

As shown in Fig. 2, to leverage a massive amount of parallelization techniques in map systems, an original image is divided into $N \times M$ tiles, where each tile is of size $n_0 \times m_0$ and J is the depth of the multi-resolution pyramid. The tiles

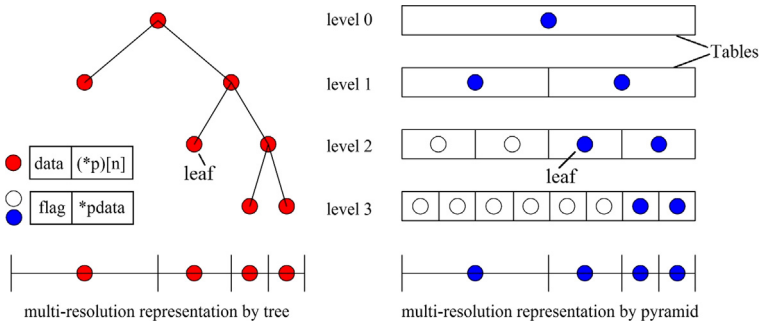

Fig. 3. Multi-resolution representations of the same case in 1D using different data structure, tree structure (left) and pyramid structure (right). In a tree a node (red circle) consists of a value (of some data type) and a list of other nodes, while a node (empty or blue circle) in the pyramid structure has two elements: a flag indicating the type of current node (empty or occupied) and a pointer to associated data (null if the node is empty). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

can be projected from level J to level $J-1$ by using a projection operator, such as Eq. (6), producing an $N \times M$ grid of fragments where each fragment includes $n_0/2 \times m_0/2$ pixels. Each 2×2 group of fragments can be merged, producing an $N/2 \times M/2$ grid of $n_0 \times m_0$ tiles containing the downsampled image at level $J-1$ of the multi-resolution image pyramid. The same process can be extended to other levels. Note that the reason why a massive amount of parallelization techniques can be used in pyramid structures is that each tile can be directly and exactly located according to its position (row and column) in the tile grid at the corresponding level. In other words, each tile knows exactly which ancestor tile it contributes to, which descendant tiles it covers and which neighboring tiles it connects to.

In order to take full advantage of both data compression and parallelization techniques in CFD, we build a table-based pyramid structure as shown in Fig. 3. It is defined as a set of tables, in which each element consisting of a *flag* and a *pointer* is called a node. All nodes are organized in the same way as tiles of the map systems. Considering data compression, the nodes can change between two states, "empty" and "occupied", which are indicated by the value of *flag*. An empty node means there is no computational data, while an occupied node implies that it is necessary to allocate memory for corresponding computational cells, and the *pointer* should point to this memory. From each table we can easily find the exact distribution of occupied nodes at the corresponding level, which is difficult in a tree. Furthermore, as shown in Fig. 3, by comparing the two data structures for the same case one can notice that they represent the same data and data relationship. Following the terminology of the tree structure, in the table-based pyramid structure a node is also called a leaf, when it has no substructures.

Note that the memory usage of all empty nodes can be neglected, since an empty node only contains an integer variable and an empty pointer. Also, there is only one pointer in a node of the pyramid structure, while a node in a tree can have 2^{D_0} pointers to its children (D_0 is the dimensions). All these properties imply that the memory efficiency of the table-based pyramid structure is similar to that of the tree structure.

When used within an adaptive MR environment it is obvious that data structures allocate and release memory frequently during run time. As all useful pointers point to fixed-size memory blocks, we employ a memory pool (also called fixed-size-blocks allocation) to manage all related implementations. The memory pool allocates a big amount of memory on startup and separates it into smaller chunks. Each chunk is available as the target of a pointer of a node to store its computational data. Every time we request memory from the pool, it is taken from the previously allocated chunks, not from the system. It causes very little memory fragmentation and is much faster than the "normal" memory allocation.

3.2. Storage-and-operation-splitting approach

In the table-based pyramid structure, each occupied node contains a block of $n_0 \times m_0$ computational cells. In the interior region of a block, because there are complete interpolation stencil elements for each grid point, the desired high-order interpolation schemes can be employed directly. However, one has to introduce auxiliary grid points (ghost points) at block boundaries (Fig. 4). Ghost points need to be constructed from other blocks at the ancestor level or copied from neighboring blocks at the same level. Although the efficient search method within the pyramid data structure can improve the speed of ghost-point construction, temporary memory allocation and deallocation for ghost points of every block at each time step still adversely affect the final performance. Moreover, the construction approach depends on the resolution jumps among adjacent blocks, which impedes parallel computing. That is why the maximum resolution jump often is restricted in other work, e.g. [20].

In our previous work [21], we refer to a concept of overlapping blocks to remedy such limitations, where memory is permanently allocated for storing boundary cell data (ghost points) for all blocks to avoid frequent memory allocation so that a restriction of the maximum resolution jump can be avoided. Consider Fig. 4(bottom), let w_b be the width of boundary layers of a block, w_i be the number of interior cells of a block and I_{max} be the number of blocks at level $l-1$, the cell set of B_I at level $l-1$ can be represented by

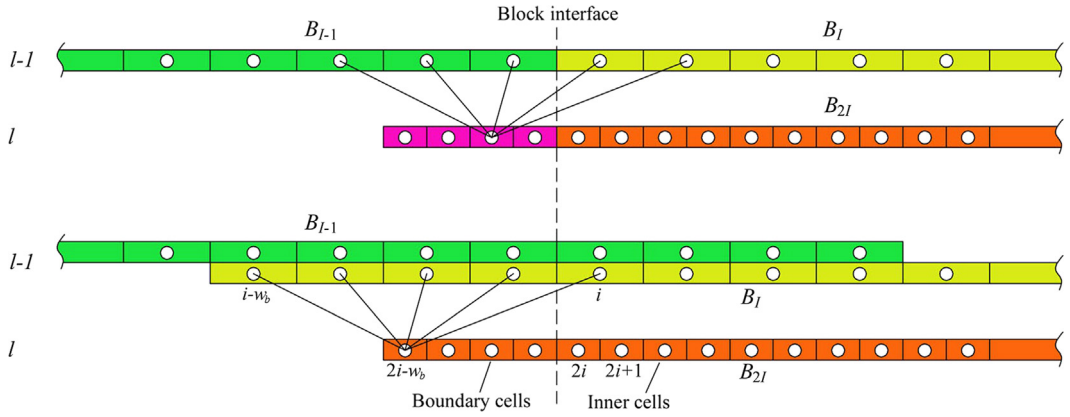

Fig. 4. Comparison of non-overlapping blocks (top) and partially overlapping blocks (bottom). The system consisting of non-overlapping blocks has high memory efficiency. Ghost cells (purple) of a block (orange) are constructed temporarily form the parent (yellow) and uncle (green) blocks at block interfaces with resolution jump. In partially overlapping blocks (orange, yellow, green regions at the bottom), permanent boundary cells are included. Boundary cells of B_{2l} are updated by interpolation only from its parent B_l, while double storage for the overlapping parts adversely affects the memory efficiency. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$C_{B_l,l-1} = [i - w_b, i + w_i + w_b - 1], \tag{11}$$

where $i = I \cdot w_i$, and $I = 0, 1, \ldots, I_{max} - 1$. Based on the relationship between block B_l at level $l-1$ and block B_{2l} at level l, the cell set of B_{2l} is

$$C_{B_{2l},l} = [2i - w_b, 2i + w_i + w_b - 1]. \tag{12}$$

To reconstruct the data of all cells in block B_{2l} by Eq. (7), one observes that all required stencil elements will be in

$$S_{B_{2l},l-1} = \left[\left\lfloor \frac{2i - w_b}{2} \right\rfloor - m, \left\lfloor \frac{2i + w_i + w_b - 1}{2} \right\rfloor + m\right], \tag{13}$$

where m is decided by the order of the adopted interpolation scheme as in Eq. (7). To ensure that all these stencil elements belong to block B_l, it is required that

$$S_{B_{2l},l-1} \subset C_{B_l,l-1}, \tag{14}$$

i.e.

$$i - w_b \leqslant \left\lfloor \frac{2i - w_b}{2} \right\rfloor - m, \tag{15}$$

and

$$i + w_i + w_b - 1 \geqslant \left\lfloor \frac{2i + w_i + w_b - 1}{2} \right\rfloor + m. \tag{16}$$

From Eq. (15) follows $w_b \geqslant 2m$. For the fifth-order interpolation, $m = 2$, we select $w_b = 4$. These parameters are used throughout our computational test cases. Substituting them into Eq. (16), one can find that Eq. (16) holds for every positive integer w_i. Accordingly, the construction of boundary cells for every block becomes very easy. For a block its corresponding boundary cells are updated by interpolation from its parent block at boundaries with resolution jump, such as B_{2l} in Fig. 4(bottom), or by copying from its neighboring blocks at boundaries without resolution jump, such as B_l and B_{l-1} in Fig. 4(bottom). When a large resolution jump occurs between two blocks, in order to update boundary cells for both blocks the approach can be performed level by level, starting from the level with the block with the coarser grid resolution, until the level the other block belongs to is reached. In practice, at each level this approach can be adopted on every block in parallel. Starting from the coarsest level, one can easily update boundary cells for all blocks in the pyramid structure level by level.

However, partially overlapping blocks result in severe memory inefficiency, especially for three dimensional simulations. Assume that a block has the same number of cells in each dimension for $D_0 = 2$ and 3. Since the whole computational domain is represented only by interior cells of all blocks, memory efficiency can be described by the ratio of the number of interior cells to the number of total cells in a block. Thus it can be calculated by

$$\eta = \left(\frac{w_i}{w_i + 2w_b}\right)^{D_0}. \tag{17}$$

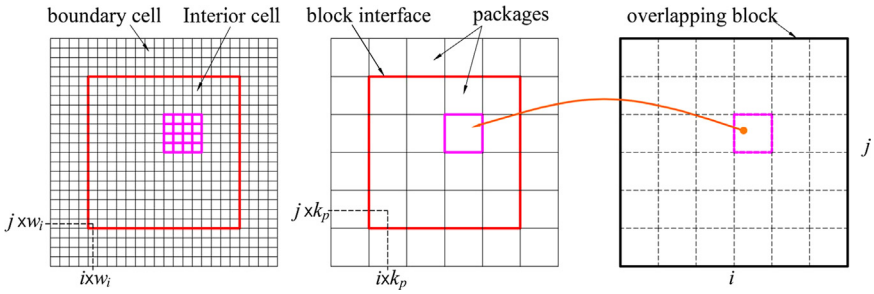

Fig. 5. Relationships among cells, packages and overlapping blocks at the same level. $w_b^{D_0}$ cells (left, magenta) are integrated into one package (middle, magenta) for storage, and then $(k_p+2)^{D_0}$ packages are mapped onto an overlapping block for evolution, where $k_p = w_i/w_b$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

When $w_i = 32$, e.g. as in [21], memory efficiency is only 64% for $D_0 = 2$, 51.2% for $D_0 = 3$. The same problem is also encountered in [6], where $\eta = 12.5\%$ for a three-dimensional case. The problem can be remedied by a storage-and-operation-splitting approach which will be described in the following.

Let the interior cell number be an integer multiple of the boundary width ($w_i = k_p \cdot w_b$). We introduce "packages" as key concept of the approach. As shown in Fig. 5, packages always include $w_b^{D_0}$ cells, so each block consists of a preset number of packages, such as $(w_i/w_b + 2)^{D_0} = (k_p + 2)^{D_0}$. All packages are also organized within a table-based pyramid structure. We will employ two sets of pyramid structures, one for blocks and the other for packages. In order to avoid confusion, we call them operation pyramid and storage pyramid, respectively, in the remaining part of this paper.

Note that the relationship is a unique mapping between packages in the storage pyramid and blocks in the operation pyramid. Given a block, we can directly find which packages it contains (see Fig. 5). Since some packages may be shared by several blocks, we set a counter for each package to identify by how many blocks it is shared. When a new block needs to be created in the operation pyramid, the procedure (called creating operation) is as follows:

1. Change the state of the corresponding node in the operation pyramid from "empty" to "occupied", and associate its block pointer with a memory chunk from the memory pool in the operation pyramid.
2. Find all corresponding package nodes in the storage pyramid. Activate the *flag* of each node if it is not activated, specify a chunk for the package pointer from the memory pool embedded in the storage pyramid, and set the counter of the package to zero ("zero" means this package has not been used by any block).
3. Map each package in the storage pyramid onto the block in the operation pyramid. Increase their counters by 1.

When an existing block needs to be removed in the operation pyramid, we employ the removing operation:

1. Find all package nodes it covers in the storage pyramid. Decrease their counters by 1. Deactivate the *flag* of the corresponding package node and release the package when its counter goes back to zero.
2. Release the block and change the state of the corresponding node in the operation pyramid to "empty".

As shown in Fig. 6, by mapping all packages stored in the memory pool of the storage pyramid onto partially-overlapping blocks in the operation pyramid, we finally develop a so-called storage-and-operation-splitting approach. In order to avoid allocating too much memory for other intermediate variables such as fluxes across cell walls. We establish a "working block" for each thread. As in PARAMESH [6], to advance the solution we loop over all leaves in the operation pyramid, in turn copying each leaf into a "working block", advancing the solution of the "working block", and finally copying the solution back to the leaf.

Note that in this approach all flow-relevant data such as primary variables and cell average quantities are stored in packages without overlap. The overlapping parts among neighboring blocks actually are package pointers. Since they point to the same memory chunks, data will be shared by neighboring blocks automatically. That is to say, a block only needs to interpolate its boundary cells at boundaries with resolution jump after a flow evolution step.

4. Multi-resolution method with sharp interface model (MR-SIM)

In this section we extend the efficient multi-resolution approach of the previous section to multi-phase simulations, where a sharp interface model is employed to track the interface.

4.1. MR analysis with sharp interface model

The main issue of multi-phase simulations with immiscible materials is accurate interface tracking. In order to resolve the interface as exactly as possible, we require that all cells within the near-interface band should be identified for refinement.

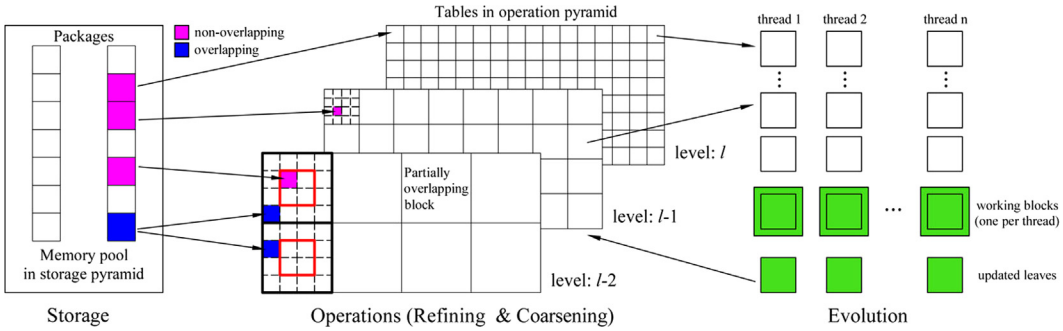

Fig. 6. Schematic of the storage-and-operation-splitting approach. Packages in the storage pyramid are mapped onto partially-overlapping blocks in the operation pyramid. By using task-based parallelism, all leaves finally are evolved in different "working blocks" in parallel.

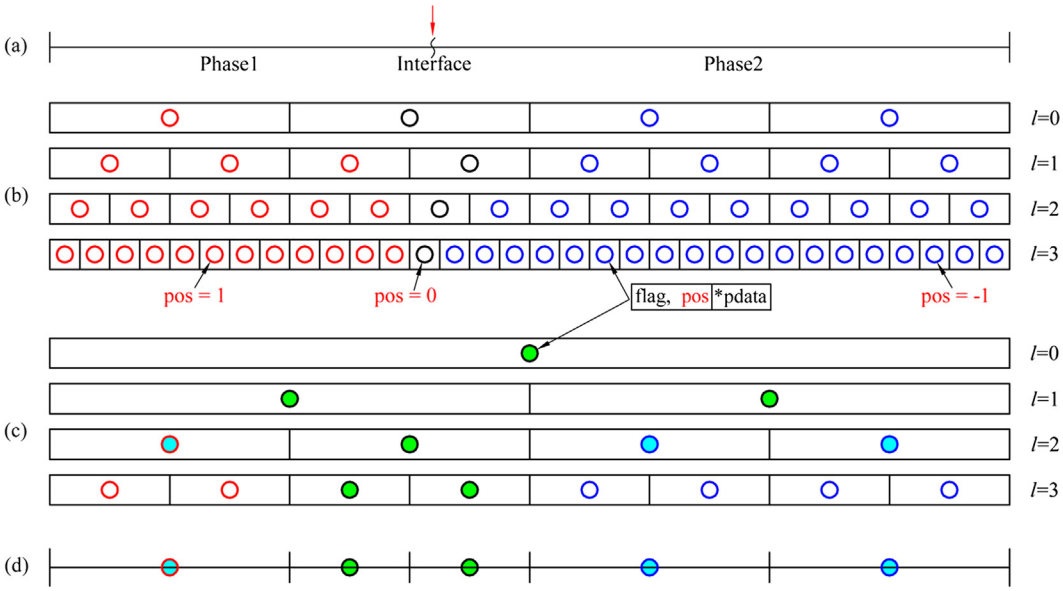

Fig. 7. Creation of a basic multi-resolution representation in 1D with $k_p = 4$ and $L_{max} = 3$. According to the level set at the finest level (a) and the relationship between packages and blocks, all position identifiers (*pos*) in the storage pyramid (b) and the operation pyramid (c) are obtained. The red, black and blue circles indicate $pos = 1, 0$ and -1, respectively. Then all nodes in the operation pyramid with $pos = 0$ (black circles) are activated (green dots) and refined. Newly created leaves (cyan dots) and all leaves cut by interface (green dots) at the finest level create the *basic multi-resolution representation* (d). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

This operation is performed recursively from the coarsest level to the finest. Eventually, all cells cut by the interface will be computed at the finest level, which has already been suggested by Sussman et al. [36]. Under this precondition, most of the tasks belonging to sharp interface models are straightforward, such as the construction of ghost fluid cells, the interface advection and reconstruction, because no resolution jump are involved. Thus, we only compute and store the level set at the finest level.

In order to generate a multi-resolution representation of a computational domain according to the current interface position, we add another identifier (see *pos* in Fig. 7) into each node of the pyramids above mentioned (both storage and operation pyramids) to distinguish the node location, i.e. to which phase it belongs or that it contains the interface. As shown in Fig. 7, we consider a 1D case as an example in the following. Assume that there are $K(l)$ packages at level l of the storage pyramid. First of all, the position identifiers at the finest level of storage pyramid are determined according to the level set by

$$pos_{k,L_{max}} = \begin{cases} 1, & \text{if } (\forall j \in \mathcal{J}\colon \phi_j > 0), \\ -1, & \text{if } (\forall j \in \mathcal{J}\colon \phi_j < 0), \\ 0, & \text{otherwise}, \end{cases} \tag{18}$$

where $k = 0, 1, \ldots, K(L_{max}) - 1$ and $\mathcal{J} = [k \cdot w_b, (k+1) \cdot w_b - 1]$.

The position identifiers at other levels are computed recursively by

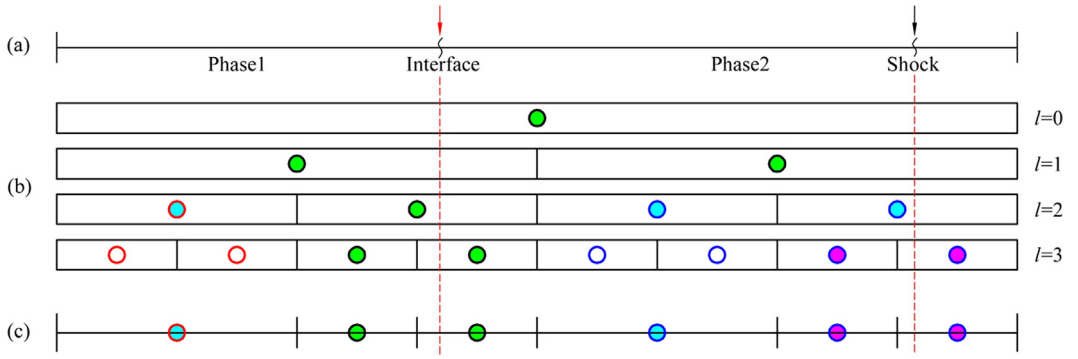

Fig. 8. Example of the final multi-resolution representation of the same case in Fig. 7. Magenta dots at level 3 denote nodes which are created due to a MR analysis not referring to the interface. (a), (b) and (c) represent the computational domain, the operation pyramid and the final multi-resolution representation, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$pos_{k,l} = \begin{cases} 1, & \text{if } (pos_{2k,l+1} = 1) \wedge (pos_{2k+1,l+1} = 1), \\ -1, & \text{if } (pos_{2k,l+1} = -1) \wedge (pos_{2k+1,l+1} = -1), \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

where $l = L_{max} - 1, \ldots, 0$ and $k = 0, 1, \ldots, K(l) - 1$.

Based on the relationship between packages and blocks, we obtain all position identifiers in operation pyramid through

$$pos^{O}_{k,l} = \begin{cases} 1, & \text{if } (\forall i \in \mathcal{I}\colon\ pos^{S}_{i,l} = 1), \\ -1, & \text{if } (\forall i \in \mathcal{I}\colon\ pos^{S}_{i,l} = -1), \\ 0, & \text{otherwise}, \end{cases} \tag{20}$$

where the O superscript denotes the operation pyramid, the S superscript denotes the storage pyramid, $\mathcal{I} = [k \cdot k_p - 1, (k+1) \cdot k_p]$ and $l = 0, 1, \ldots, L_{max}$. Note that a block consists of interior and boundary packages, the position identifier of a global boundary package, such as $pos^{S}_{-1,l}$ when $k = 0$ in Eq. (20), is determined according to physical boundary conditions, e.g. $pos^{S}_{-1,l} = pos^{S}_{0,l}$ if there is a reflecting-wall boundary condition.

Since all effective flow evolution is obtained from leaves and all ghost-fluid cells needed by sharp interface models exist only at the finest level, in order to assure that all leaves that are not at the finest level can be viewed as single-phase sub-domains and treated by single-phase discretization schemes, we refine all nodes in the operation pyramid with $pos = 0$ once again, see Fig. 7(c). Accordingly, all leaves except that at the finest level become single-phase blocks as expected. We call the current multi-resolution representation a *basic MR representation*, which is created only according to the interface position, see Fig. 7(d). Based on this basic MR representation, a standard single-phase-model multi-resolution analysis can be employed recursively on all leaves except the finest level to create the final representation as shown in Fig. 8.

Note that there are actually only two types of leaves within the final multi-resolution representation, see Fig. 8(c), single-phase leaves and multi-phase leaves, and all such multi-phase leaves are at the finest level. That means it is not essential to store the level set and ghost cell data on single-phase leaves and all other nodes which are not leaves.

4.2. Narrow band technique

According to the above analysis, all multi-phase leaves naturally form a narrow band. Unlike in [37], our narrow band technique restricts all interface-relevant data and operations to a neighborhood of the interface, not only the level set but also ghost-cell data.

In order to store these data, we insert another two tables at the finest level in the operation pyramid of Fig. 8. As shown in Fig. 9, the three tables at the finest level are used to manage blocks from two different phases and the level set, respectively. Based on the MR analysis with sharp-interface models mentioned above, when a node needs to be created at the finest level, we activate the *flag* at the appropriate position of the corresponding table according to its position identifier *pos* (1 for phase 1 and -1 for phase 2). If it is a multi-phase node, all *flags* at the appropriate position of these 3 tables need to be changed from “empty” to “occupied”. Accordingly, there are also 3 corresponding tables needed at the finest level in the storage pyramid. The procedures to create and remove a node in the operation pyramid are the same as in Section 3.2.

Since there may be 3 blocks at a position at the finest level of the pyramid data structure for two-phase flows, see Fig. 9, but only 1 block exists at the same position at their parent level, the projection operator $P_{l+1 \to l}$, Eq. (6), becomes ill defined. As the volume fraction α of phase 1 in a cell can be computed by the level set, we define another projection operator $P^{multi}_{L_{max} \to L_{max}-1}$ specially for the multi-phase nodes at the finest level,

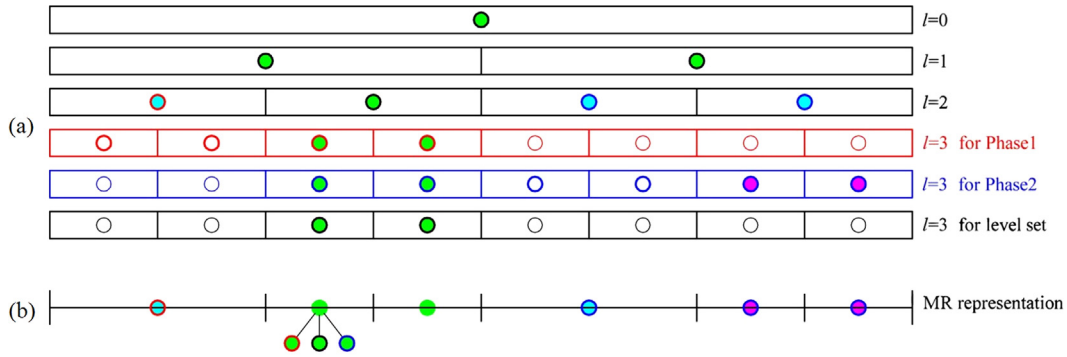

Fig. 9. Example of the operation pyramid (a) and final MR representation (b) of the same case in Fig. 8. There are 3 tables at the finest level ($l = 3$) corresponding to phase 1 (red circles), phase 2 (blue circles) and the level set (black circles), respectively. For single-phase nodes (cyan or magenta dots) at the finest level, at most one block exists at the same position. For multi-phase leaves (green dots) at the finest level, all 3 nodes are "occupied" at the same position. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

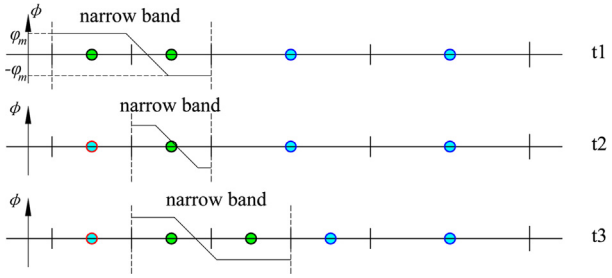

Fig. 10. Adaptive update of a MR representation at different time instances. At time $t1$ the narrow band contains two nodes (top). At time $t2$ the first node changes to single-phase node (middle). The third node are created at time $t3$ (bottom) and its initial values of level set are set to $-\phi_m$.

$$P^{multi}_{L_{max}\to L_{max}-1}:\ \bar{u}_{L_{max}-1,i} = \frac{\alpha}{2}\left(\bar{u}^1_{L_{max},2i} + \bar{u}^1_{L_{max},2i+1}\right) + \frac{1-\alpha}{2}\left(\bar{u}^2_{L_{max},2i} + \bar{u}^2_{L_{max},2i+1}\right), \tag{21}$$

where the superscripts 1 and 2 denote the two phases respectively.

Recall that before each time step all position identifiers in the pyramid data structure should be updated by Eqs. (18)–(20). As the level set is only known initially in the whole domain and restricted within the narrow band during the simulation, Eq. (18) is employed in the entire domain only once at the beginning and subsequently only within the narrow band. As shown in Fig. 10, for a moving interface some nodes can move out of the band and some can move in. After a node has moved out, its position identifier will be fixed, Fig. 10(middle). However, when a node moves into the narrow band, a corresponding level set node would be created in operation pyramid. Time advancement of the level set technique requires appropriate level set values. For this purpose, we apply a cut-off function [38] to the level set,

$$\phi = \begin{cases} \phi_m & \text{if } \phi > \phi_m, \\ -\phi_m & \text{if } \phi < -\phi_m, \\ \phi & \text{otherwise}, \end{cases} \tag{22}$$

where ϕ_m is an integer multiple of the grid size. Since a node will be created when the interface is $2w_b$ cell sizes away from it, we choose $\phi_m = 8\Delta x$ in this work. When a node moves into the narrow band, the initial level set values of all cells of the node will be set to ϕ_m or $-\phi_m$, Fig. 10(bottom).

During the evolution stage, as every leaf within the MR representation contains its own boundaries, they can be viewed as sub-domains with uniform grids. The two types of leaves can be advanced by using single-phase numerical methods [39–43] and multi-phase numerical methods with sharp interface models [24,25,44–46], respectively. In this paper, the conservative sharp interface method is adopted due to its robustness and accuracy.

Note that the above method has been implemented in 1D, 2D and 3D. The respective extensions of the discussion using 1D and 2D cases to 3D is straightforward. Numerical examples are shown for 2D as they give the purpose of demonstrating the efficiency of the method.

5. Numerical examples

The following numerical examples are provided to illustrate the potential of the present multi-resolution method with sharp interface models (MR-SIM) for multi-phase flows. Different kinds of flow problems involving gas–gas and gas–water interactions with shock waves are calculated to indicate the robustness of the present method with respect to obtaining a

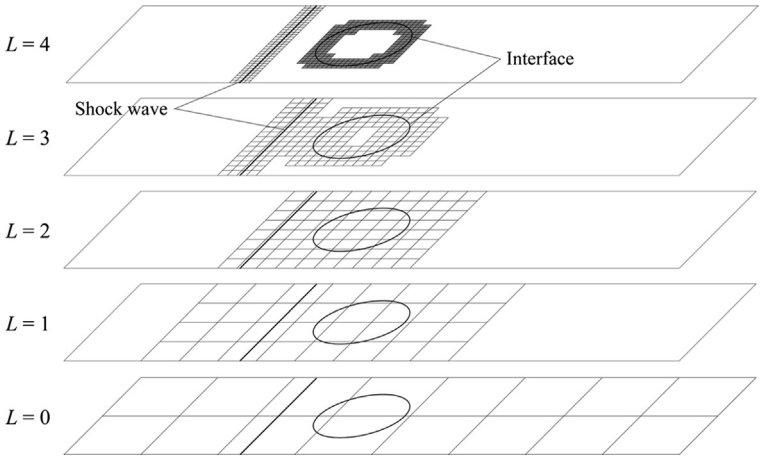

Fig. 11. The "occupied" node distribution at different grid levels for 2D air–bubble interaction at $t = 0$. Only the shaded blocks at the finest level ($L = 4$) contain both real and ghost fluids.

physically meaningful solution for very complex interface structures. For all test cases, we employ the ideal-gas equation of state for gas phases

$$p = (\gamma - 1)\rho e, \tag{23}$$

and Tait's equation for water,

$$p = B\left(\frac{\rho}{\rho_0}\right)^{\gamma} - B + A, \tag{24}$$

where $\gamma = 7.15$, $A = 10^5$ Pa, $B = 3.31 \times 10^8$ Pa and $\rho_0 = 1000$ kg/m^3.

If not mentioned otherwise, $w_i = 16$ will be employed in the operation pyramid, and the MR-analysis is performed for density with the threshold $\epsilon_{L_{max}} = 0.01$. The evolution equations of the individual phases are discretized by the fifth-order WENO-LF method [1] and the second-order TVD Runge–Kutta scheme [18]. All the computations are carried out with a CFL number of 0.6.

In order to test the parallel performance of the MR-SIM approach, all simulations are carried on a workstation with 4 quad-core Intel Xeon Processor E5620 processors (12 M Cache, 2.40 GHz) with 24 GB of RAM. The Intel Threading Building Blocks (TBB) library [28] is used to map logical tasks to physical threads.

5.1. Case I: 2D air–helium shock bubble interaction

In this problem, we compute a Mach 6 shock wave in air interacting with a cylindrical helium bubble. Numerical computations for the same problem can be found in [24] on a uniform grid. The initial conditions are

$$\begin{cases} (\rho = 1,\ u = 0,\ v = 0,\ p = 1,\ \gamma = 1.4) & \text{pre-shocked air,} \\ (\rho = 5.268,\ u = 5.752,\ v = 0,\ p = 41.83,\ \gamma = 1.4) & \text{post-shocked air,} \\ (\rho = 0.138,\ u = 0,\ v = 0,\ p = 1,\ \gamma = 1.667) & \text{helium bubble,} \end{cases} \tag{25}$$

with dimensional references given by the state of air at 1 atmosphere and 1 m. The helium bubble of initial radius 0.025 at $(0.15, 0.0445)$ is impacted by a shock wave initiated at $x = 0.1$. In order to obtain as much data compression as possible, the domain size is extended to 0.356×0.089 with an aspect ratio of 4:1, which allows to use 8×2 blocks at the coarsest level. The upper and lower boundaries are solid walls, while the left and the right boundaries are inflow and outflow, respectively. As shown in Fig. 11, with the maximum level of resolution at $L_{max} = 4$ and a large block size $w_i + 2w_b = 40$, the effective grid resolution at the finest level is 4096×1024. Based on our two-step strategies on topological structure upbuilding and updating, the near-interface regions are refined up to the maximum level according to the interface position, while the shock is captured through the MR analysis.

Fig. 12 shows the density fields at $t = 6.0 \times 10^{-3}, 1.2 \times 10^{-2}, 1.8 \times 10^{-2}$ and 2.4×10^{-2}. These results show a good agreement with that of [24] (their Fig. 10). However, due to the higher effective resolution of the present simulation, very complex wave and interface structures are produced, especially in the roll-up region. Much more of complex mixing details are captured at later times than in [24], Fig. 12(c) and (d). A large amount of air filaments are entrained into the deformed helium bubble, and finally turn into isolated droplets when their sizes are smaller than the minimum resolved scale.

As shown in Fig. 12(b), because strong activities occur near the interface and a small error threshold 0.01 is chosen, all blocks near the interface are calculated with the finest resolution. On the other hand the total simulation time is not too

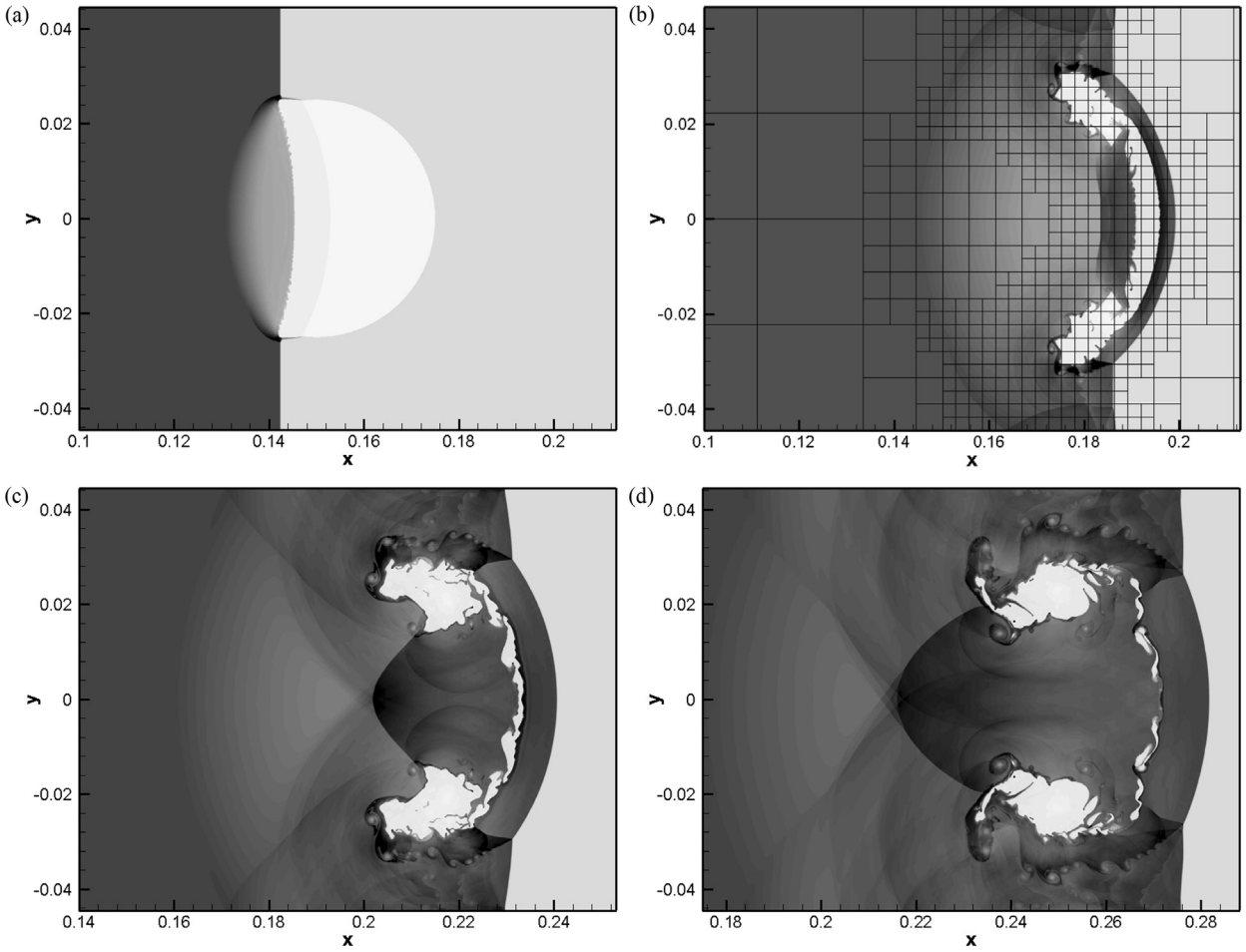

Fig. 12. Density fields for 2D air–helium interaction case with Mach number of 6.0 at different time instances. A multi-resolution representation is outlined at time $t = 1.2 \times 10^{-2}$.

Table 1
Memory usage and data compression rates for simulations of Case I with different data structures on grids with the same (effective) resolution (4096×1024).

Methods	L_{max}	Memory usage (GB)	Data compression rate
Hu et al. [24]	–	6.38	–
Han et al. [21]	0	4.49	0.296
	4	1.47	0.770
MR-SIM (present)	0	2.54	0.602
	4	0.81	0.873

long to allow waves transmit far enough, which leads to that the regions far away from the interface have to be handled on the coarser grids. Finally, the maximum memory usage is 813 MB corresponding to a data compression rate of 87.3%. Table 1 shows comparison of memory usage and data compression rates for simulations with different data structures. All these simulations employ the conservative sharp interface method [24] for calculating multi-phase flows, so they store the same variables. The memory usage differences here result from different data structures. Compared with the non-adaptive method [24], both of the adaptive MR methods reduce the memory usage by avoiding redundant ghost cells for both fluid and level set in regions far away from the material interface. The present storage technique based on non-overlapping data packages reduces the memory usage significantly.

5.2. Case II: 2D air–R22 shock bubble interaction

Within the same simulation domain as in Section 5.1, an analogous problem is considered by Nourgaliev et al. [37] and So et al. [47], which involves a weak shock wave of Mach 1.22 in air interacting with a cylindrical bubble of Refrigerant-22

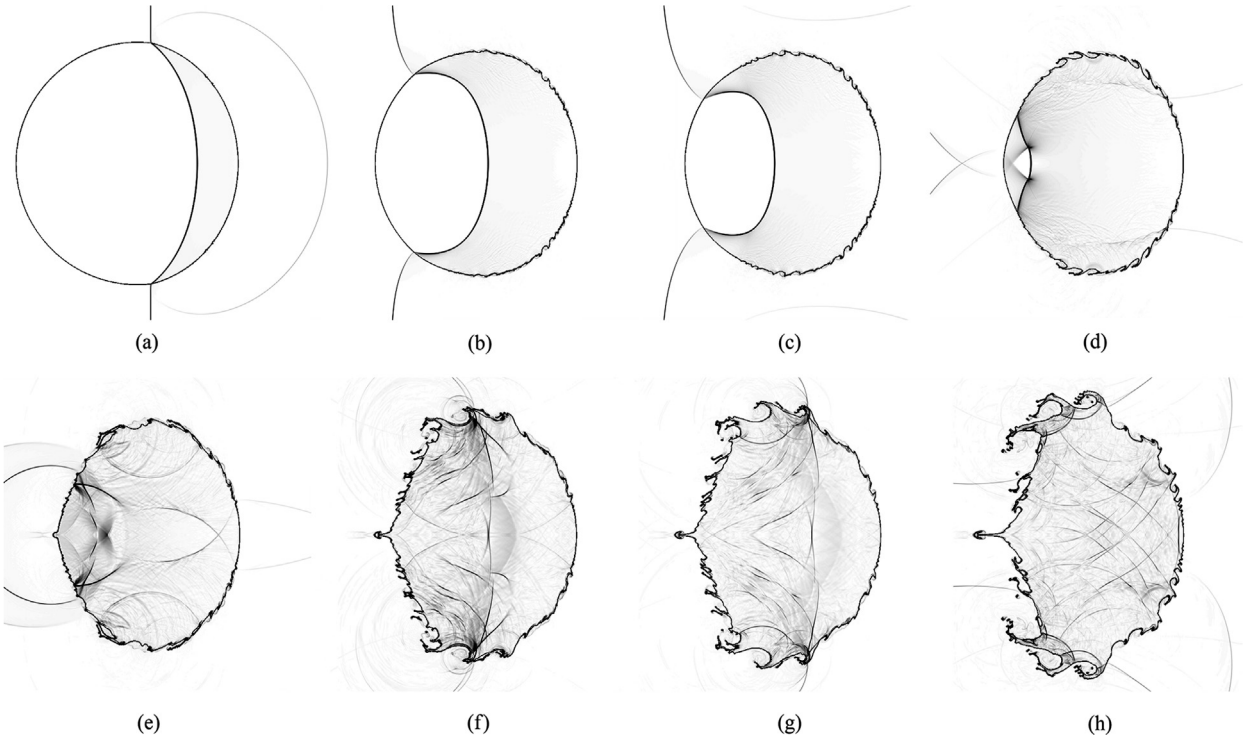

Fig. 13. Schlieren-type images, $|\nabla\rho|$, of the air–R22 shock-bubble interaction. Time recording begins from the shock impact on R22 bubble, (a) $t = 55$ μs, (b) 115 μs, (c) 135 μs, (d) 187 μs, (e) 247 μs, (f) 318 μs, (g) 342 μs, (h) 417 μs. Effective resolution is about 1150 cells per initial bubble diameter.

(R22). Corresponding experiments were carried out by Haas and Sturtevant [48]. Dimensional reference values given by the state of air at 1 atmosphere and 1 m, the initial conditions are expressed non-dimensionally as:

$$\begin{cases} (\rho = 1,\ u = 0,\ v = 0,\ p = 1,\ \gamma = 1.4) & \text{pre-shocked air,} \\ (\rho = 1.3764,\ u = 0.3947,\ v = 0,\ p = 1.5698,\ \gamma = 1.4) & \text{post-shocked air,} \\ (\rho = 3.154,\ u = 0,\ v = 0,\ p = 1,\ \gamma = 1.249) & \text{R22 bubble.} \end{cases} \tag{26}$$

In order to compare the current results with that in the reference literature, the coordinate system is defined with the origin at the lower right corner of the domain. Solid walls boundary conditions are imposed at the upper and the lower boundaries, while the left and the right boundaries are outflow and inflow, respectively. Assuming that flow field is symmetric about the streamwise center axis, only the top half domain is computed. We set 8×1 blocks at the coarsest level, and with five levels of adaptation and block size of $w_i + 2w_b = 40$, the effective grid resolution at the finest level is 8192×1024, which corresponds to about 1150 computational points per initial bubble diameter.

Schlieren-type images of density gradient $|\nabla\rho|$ at the same time instants as the experiment [48] are shown in Fig. 13, which are in good agreement not only with the experiments but also with previous simulations [37,47]. Again, our method shows much more details than previous simulations. Compared to the reference results of Nourgaliev et al. [37], employing adaptive mesh refinement (AMR) algorithms with an effective grid resolution of approximately 900 cells across the bubble diameter, the level of details of the interface evolutions obtained here is much higher. Note that the MR-SIM method is sufficiently sensitive with respect to small-scale structures, so that the interface instability develops earlier than in [47]. A clear spike is found on the symmetric axis at later time steps.

As can be seen from Fig. 14, the MR-analyzer also identifies wave propagation and reflection within the bubble and thus refines the entire bubble interior. Nevertheless, the obtained data compression outside the bubble is high as nearly only the wave fronts are handled on the finest resolution grid. At late stages exist 3899 blocks, which corresponds to a data compression rate of 52.4%, whereas it is 94.04% at early stages. The entire simulation uses less than 739 MB of memory. Note that in the remainder of this paper the data compression rate is obtained by

$$\zeta = 1 - \frac{n_{ol}}{n_{bf}}, \tag{27}$$

where n_{ol} and n_{bf} denote number of "occupied" leaves and number of blocks at the finest level, respectively.

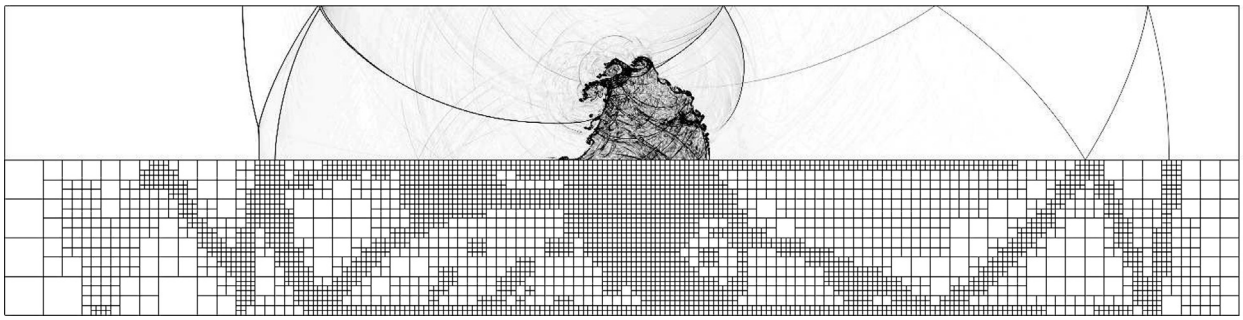

Fig. 14. Numerical Schlieren of density gradient and the MR representation of the whole field for Case II at time $t = 417$ μs.

5.3. Case III: Collapse of 2D air bubble under strong shock in water

The same problem has already been considered by Hu and Khoo [32] and Nourgaliev et al. [37]. A 6 mm cylinder air bubble in water is impacted by a strong shock under the following non-dimensional initial conditions:

$$\begin{cases} (\rho = 1,\ u = 0,\ v = 0,\ p = 1,\ \gamma = 7.15) & \text{pre-shocked water,} \\ (\rho = 1.31,\ u = 67.32,\ v = 0,\ p = 19\,000,\ \gamma = 7.15) & \text{post-shocked water,} \\ (\rho = 1.2 \times 10^{-3},\ u = 0,\ v = 0,\ p = 1,\ \gamma = 1.4) & \text{air bubble.} \end{cases} \tag{28}$$

Reference state is the property of water at 1 atmosphere and a length scale 1 mm. The center of the bubble is located at $(6, 6)$ in the computational domain of size 12×12, and the initial position of shock wave is $x = 2.4$. Boundary conditions are all outflow boundaries with zero gradient. Simulations are performed with only one block at the coarsest level and $L_{max} = 7$, which indicates an effective resolution of 2048×2048 on the finest level and 1024 cells per initial bubble diameter.

In order to directly apply the numerical methods [39–43] developed for compressible gas flow to the water medium, an expression for the internal energy associated with the Tait's equation of state was proposed in [49]. A set of Roe-averaged values is defined by

$$\tilde{f} = \mu(f) = \frac{\sqrt{\rho_l} f_l + \sqrt{\rho_r} f_r}{\sqrt{\rho_l} + \sqrt{\rho_r}}, \quad f = u, v, H \tag{29}$$

where $H = \frac{E+p}{\rho}$ represents enthalpy. A corresponding averaged sound speed is

$$\tilde{c}^2 = (\gamma - 1)\left[\tilde{H} - \frac{1}{2}\left(\tilde{u}^2 + \tilde{v}^2\right)\right]. \tag{30}$$

Since Tait's equation of state is independent of entropy, the energy equation is not required for solving water flows. Hence, we employ a generalized Roe average form of the sound speed $\tilde{c}$ developed by Hu et al. [25] instead as computing $\tilde{c}$ for Eq. (30). With Tait's equation of state, it can be further simplified to

$$\tilde{c}^2 = \mu\left(c^2\right) = \frac{\sqrt{\rho_l} c_l^2 + \sqrt{\rho_r} c_r^2}{\sqrt{\rho_l} + \sqrt{\rho_r}}. \tag{31}$$

In all the following simulations, Eq. (31) is employed to replace Eq. (30) for the water phase.

Fig. 15 shows the evolution of the bubble collapse, which is in good agreement with the computations of Hu and Khoo [32]. After the high speed jet with velocity of about 2800 m/s divides the bubble into two parts, pressure quickly decreases and finally results in strong cavitation. We employ the simplest cavitation model, cut-off model [50], in this simulation for convenience. More sophisticated models can be employed in a straightforward way.

Although with a different EOS for water, we obtain almost the same solution as Nourgaliev et al. [37], in terms of interface evolution and density and pressure distribution. However, the higher resolution and the sharp-interface technique make us observe much more details than in [32,37], such as tertiary jets in Fig. 16. The MR representation in Fig. 16 indicates that multi-resolution schemes are more efficient than AMR. Especially, block-based MR schemes can enhance parallelism. Again, the simulation shows efficient memory usage of 355 MBytes corresponding to a high data compression rate of 91.30%.

5.4. Case IV: Shock and water-column interaction problem

In this section, we employ two cases to study phenomena caused by a planar shock wave impinging upon cylindrical water columns, which have been simulated previously by Chang and Liou [51], Hu et al. [25] and Chang et al. [52]. In the first case, as shown in Fig. 17, a shock wave of Mach 3 hits a single cylindrical water column with initial conditions

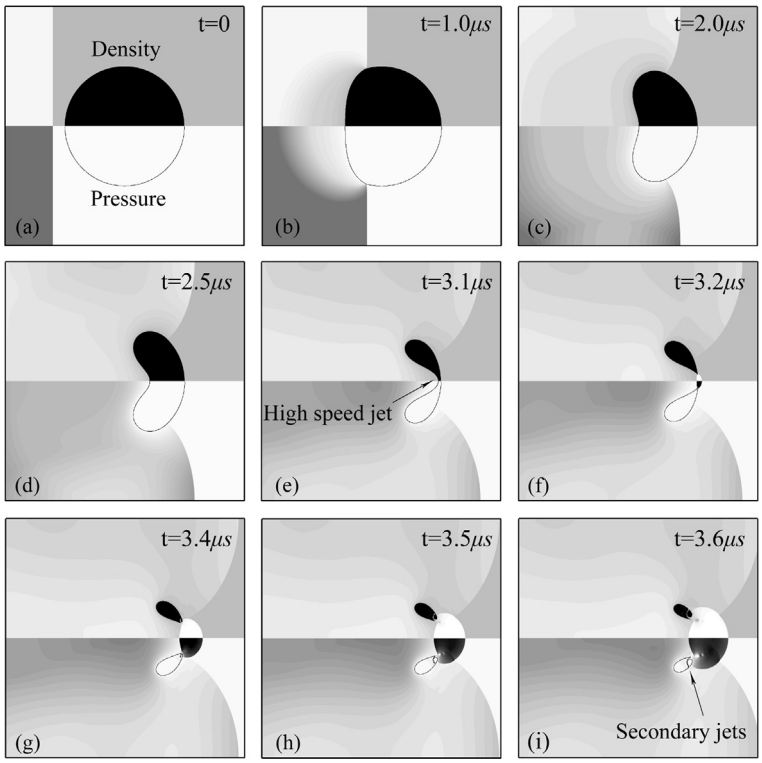

Fig. 15. Dynamics of the density and pressure at different time instances for the case of the collapse of a 2D air bubble under a strong shock in water. Effective resolution is 1024 grid cells per initial bubble diameter.

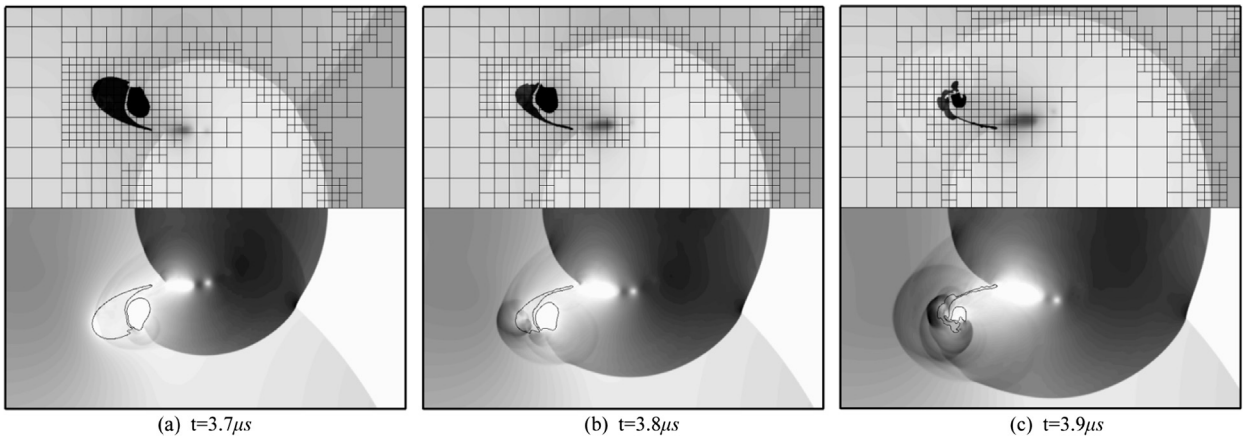

Fig. 16. Density, pressure and MR representations at last time instances for the case of the collapse of a 2D air bubble under a strong shock in water. The upper half shows the density filed, and the lower half represents the pressure.

$$\begin{cases} (\rho = 1.2,\ u = 0,\ v = 0,\ p = 1,\ \gamma = 1.4) & \text{pre-shocked air,} \\ (\rho = 4.628,\ u = 2.4,\ v = 0,\ p = 10.33,\ \gamma = 1.4) & \text{post-shocked air,} \\ (\rho = 1000.0,\ u = 0,\ v = 0,\ p = 1,\ \gamma = 7.15) & \text{water column,} \\ \phi = -0.175 + \sqrt{(x - 0.55)^2 + y^2} & \text{level set,} \end{cases} \tag{32}$$

where $\phi < 0$ indicates the region occupied by water, and $\phi > 0$ represents the air phase. The computational domain is extended to a square region $[0, 1.4] \times [-0.7, 0.7]$, in which a 2D water column with diameter of 0.35 is located at $(0.55, 0.0)$ and an air shock wave is initialized at $x = 0.35$. Reference data for non-dimensionalization are based on $p_0 = 1.0$ atm, $\rho_0 = 1.0$ kg/m^3 and $L = 1.0$ cm. Boundary conditions employed are a solid wall at the top and bottom sides, an inflow boundary at the left side and an outflow boundary with zero gradient at the right side. Beginning from one block at the coarsest level and refining the regions with large flow gradients to level $L_{max} = 7$, finally we obtain an effective grid of 2048×2048 cells at the finest level.

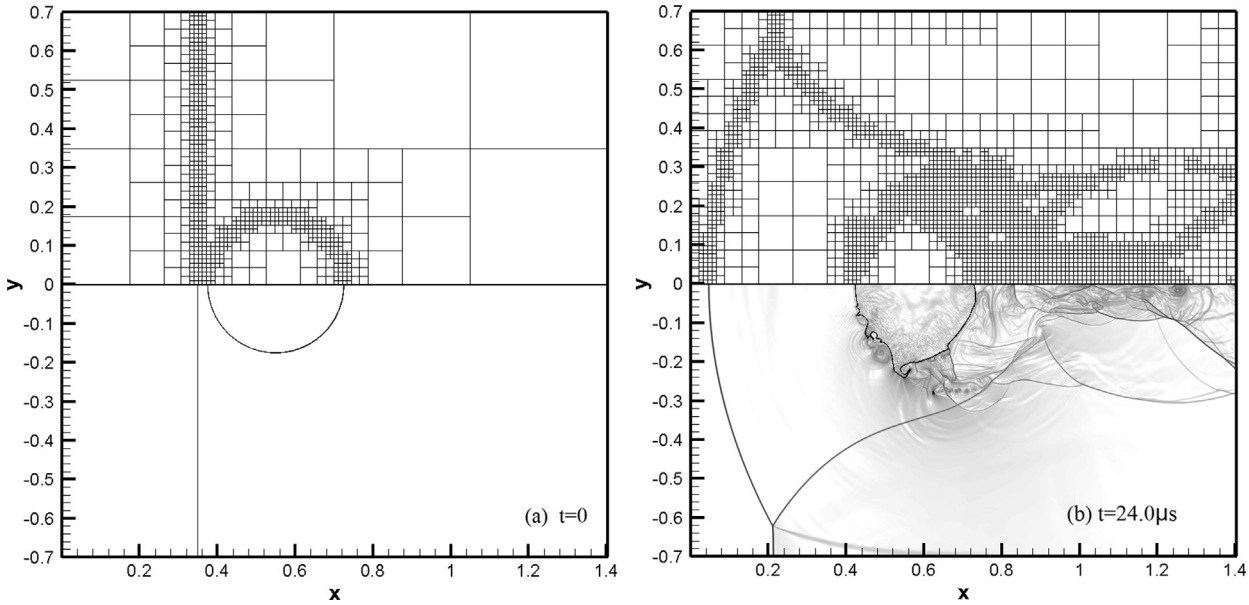

Fig. 17. MR representation of blocks (the upper half) and Schlieren-type image of density (the lower half) at $t = 0$ (left) and $t = 24.0$ μs (right) for the case: shock interacting with one water bubble. Date compression rates are 91.7% and 69.1%, respectively.

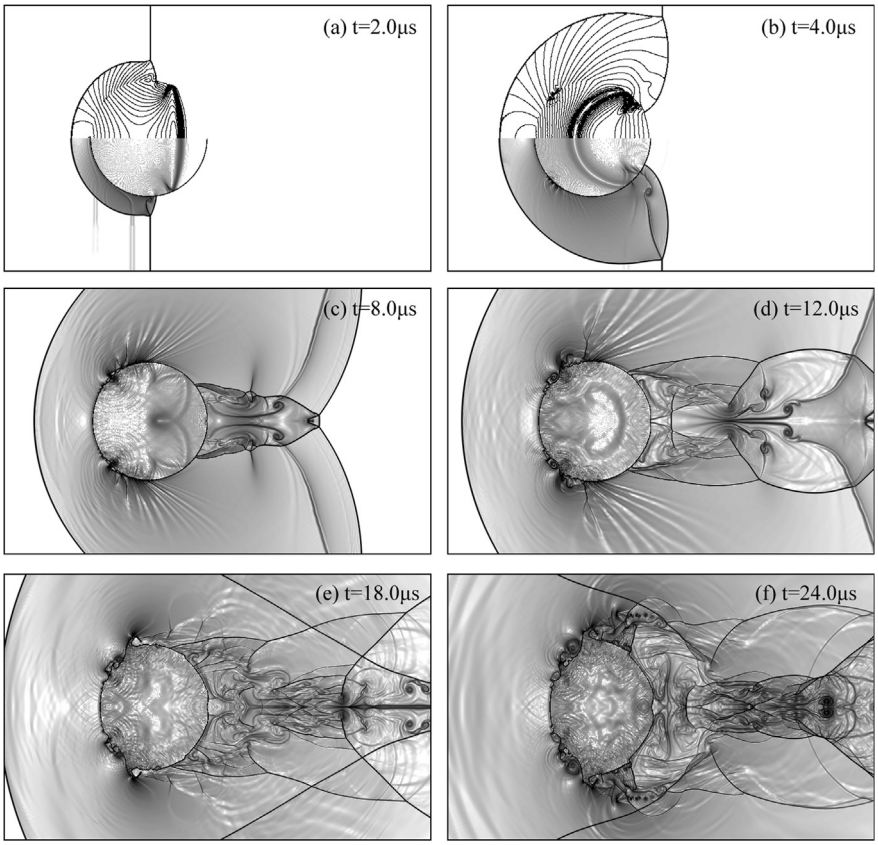

Fig. 18. Shock interacting with a single water column: Schlieren-type image for density at different times with an effective resolution of 2048 × 2048. The upper halves of the top two figures show pressure contours at 2.0 μs and 4.0 μs, respectively.

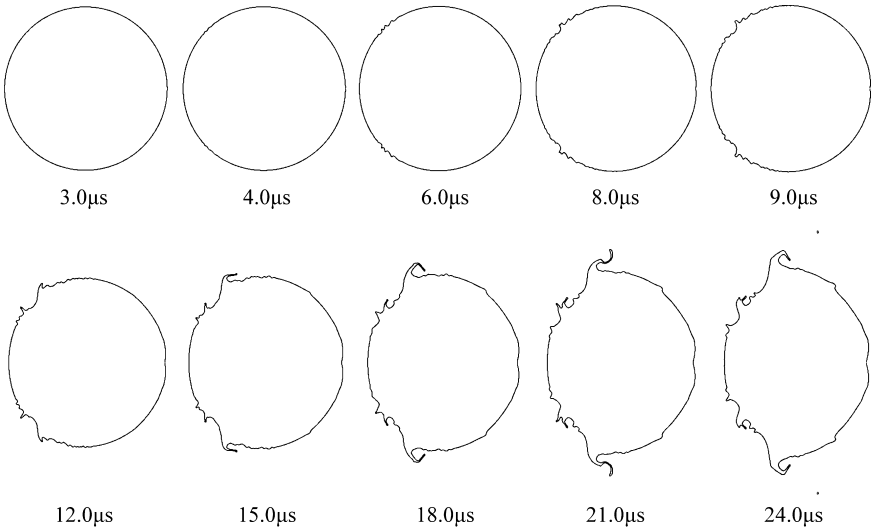

Fig. 19. Shock interacting with one water bubble: Interface outlines at different time instances.

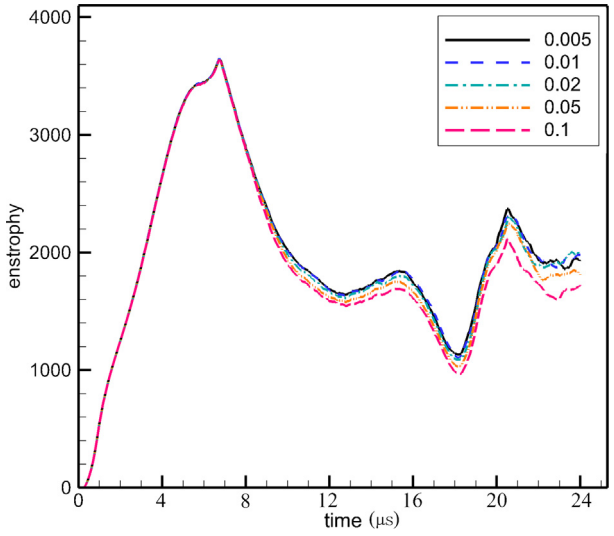

Fig. 20. Temporal evolution of enstrophy for different values of threshold ϵ.

Fig. 18 show Schlieren-type images of density at some specific times. All results here are in good agreement with those of Hu et al. [25] (their Fig. 10). Our results not only capture compressible interface instabilities (Fig. 19) better than previous computations, but also reproduce much finer details of the gas flow flied. From the last time instants shown in Figs. 18 and 19, we clearly see that the water filaments finally break up into isolated water droplets.

According to Fig. 17(a), there are 1356 "occupied" blocks in all tables at the beginning, distributed along the interface and shock front. Compared to a uniform grid with 2048×2048 cells, the data compression rate reaches up to 91.7%. Since a threshold value of 2.0% is adopted here for multi-resolution analysis, and the transmitted and reflected waves in the water column are relatively weak, almost all regions inside the bubble are not refined here, see Fig. 17(b). The turbulent air flows and interface location form a final distribution of 5064 blocks, which corresponds to a data compression rate of 69.1%. This case demonstrates the high efficiency memory usage of 669 MBytes.

Based on this case, a threshold study is performed for different values of ϵ between 0.005 and 0.1. As shown in Fig. 20, the temporal evolution of enstrophy ($\int_\Omega |\nabla \mathbf{v}|^2 \, d\Omega$) is plotted for different values of threshold. Since all near-interface regions are required to be refined to the finest grid level in our MR-SIM method, no matter how much the value of threshold is we capture almost the same vorticity in these regions. That is why we get almost the same enstrophy evolutions before $t = 8$ μs. After that, time differences in vorticity generation occur at the downstream side of interface in the air for different values of threshold. However, with decreasing threshold, the solution exhibits convergence as reported in [20].

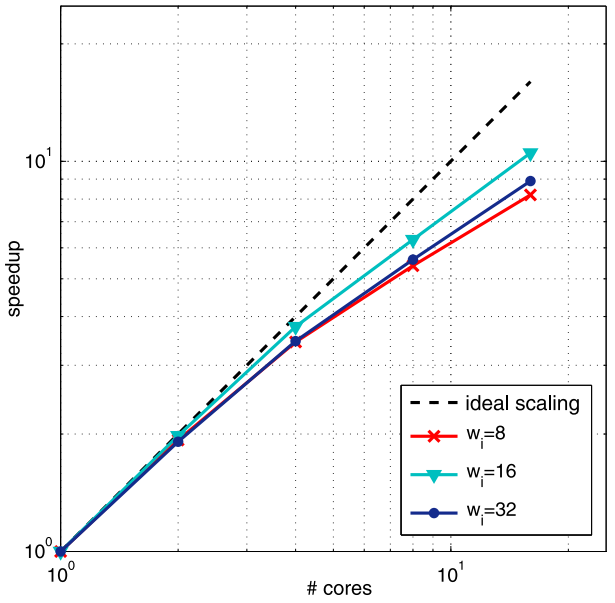

Fig. 21. Scaling versus number of cores for different number of interior cells (w_i) per block.

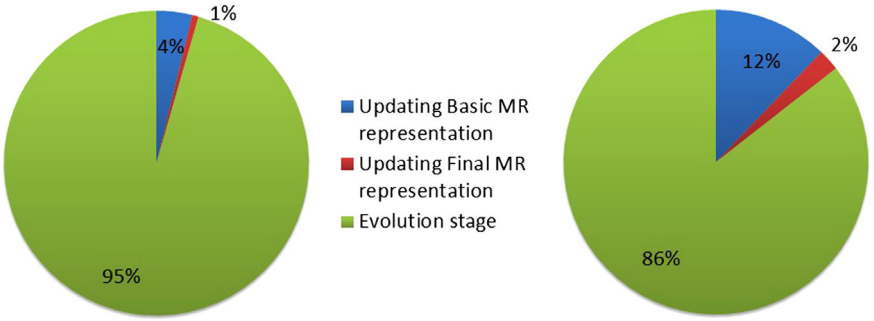

Fig. 22. Percentage of different stages in one simulation time step for execution on 1 core (left) and 16 cores (right). Both of the two "updating MR representation" stages consist of their own refinement and coarsening parts.

We also investigate the effect of the number of interior cells (w_i) in each block on the scaling of our MR-SIM method while running on from 1 core up to 16 cores for this set of simulations. It can be deduced from Fig. 21 that $w_i = 16$ gives the best scaling. Actually, this conclusion applies to most of our test cases, which is different from the observation in [20]. The reason is that the storage-and-operation-splitting technique avoids explicit inter-block communications. Furthermore, the construction of ghosts for other boundary cells at block interfaces with resolution jump is much easier for overlapping blocks than for non-overlapping blocks (Fig. 4). Accordingly, the decrease of w_i has a weak effect on the ghost construction, but obviously improves the data compression rate and efficiency. However, the decrease of w_i also increases logical operations. That is why $w_i = 8$ is inferior to $w_i = 16$.

Furthermore, we analyze the percentage of execution time spent in different stages of one simulation time step. The execution time is divided into three categories: updating the basic MR representation, updating the final MR representation and the evolution stage. The two MR-representation update stages consist of their respective refinement and coarsening operations. In the evolution stage we complete all other calculations in [24], such as solving flow evolution equations of individual phases, tracking interface, performing mixing procedure and calculating interface exchanges. As shown in Fig. 22, the scalability of the basic MR representation update stage is less than that of the others because almost all sequential logical operations are carried out in this stage. Note that since the special techniques employed in this paper such as the pyramid data structure, the storage-and-operation-splitting method and the memory pool all occur in the two MR representation update stages, and these two stages take only a small percentage of the whole simulation time, the present method shows only a moderate improvement in speedup compared to our previous work [21]. The most important feature of our method is the improved memory usage.

Next we consider a more complicated problem involving a stronger wave of Mach 6 and two different-size water columns. The initial conditions are

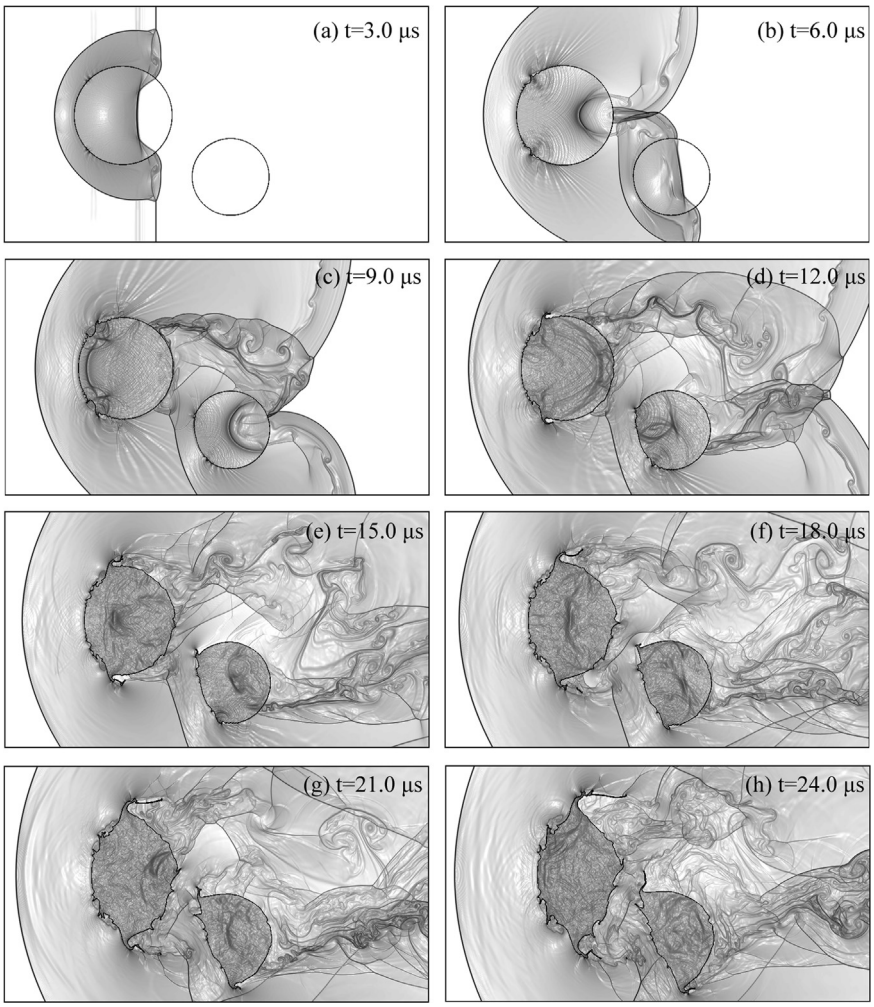

Fig. 23. Shock interacting with two water columns: Schlieren-type image for density at different times with an effective resolution of 2048×2048.

$$\begin{cases} (\rho = 1.2,\ u = 0,\ v = 0,\ p = 1,\ \gamma = 1.4) & \text{pre-shocked air,} \\ (\rho = 6.322,\ u = 5.25,\ v = 0,\ p = 41.83,\ \gamma = 1.4) & \text{post-shocked air,} \\ (\rho = 1000.0,\ u = 0,\ v = 0,\ p = 1,\ \gamma = 7.15) & \text{water columns.} \end{cases} \tag{33}$$

Different from the previous case, we use a larger domain size $[0, 2.8] \times [0, 2.8]$, where two 2D water columns of radius 0.32 and 0.25 are located at $(0.8, 1.4)$ and $(1.5, 1.0)$, respectively, and the shock wave is initialized at $x = 0.4$. The same reference quantities, boundary conditions and refinement parameters as in the first case are employed. The parameters lead to an effective resolution of 2048×2048 points at the finest level ($L = 7$). In order to capture more details about the reflected waves inside the two columns, we set all water regions at the finest resolution level.

The numerical Schlieren plots of the simulation results are shown in Fig. 23. They are in good agreement with those in [51] and [25]. Due to the high effective resolution, we capture a lot of finer details of both the interface outline and the flow evolution, especially in the water field. Since we refine all water region, the transmitted and reflected waves in the water columns are much finer than those of the first case. Although all-refined water regions increase the memory usage, our MR-SIM approach also shows a good performance with the maximum memory usage of 1.07 GB corresponding to a data compression rate of 52.17%.

5.5. Case V: Underwater explosion

This is a two-dimensional test case simulated before by Grove and Manikoff [53], Shyue [54] and Hu et al. [25]: an underwater cylindrical bubble of gaseous explosive products expands and drives the air–water interface, under the following non-dimensional initial conditions:

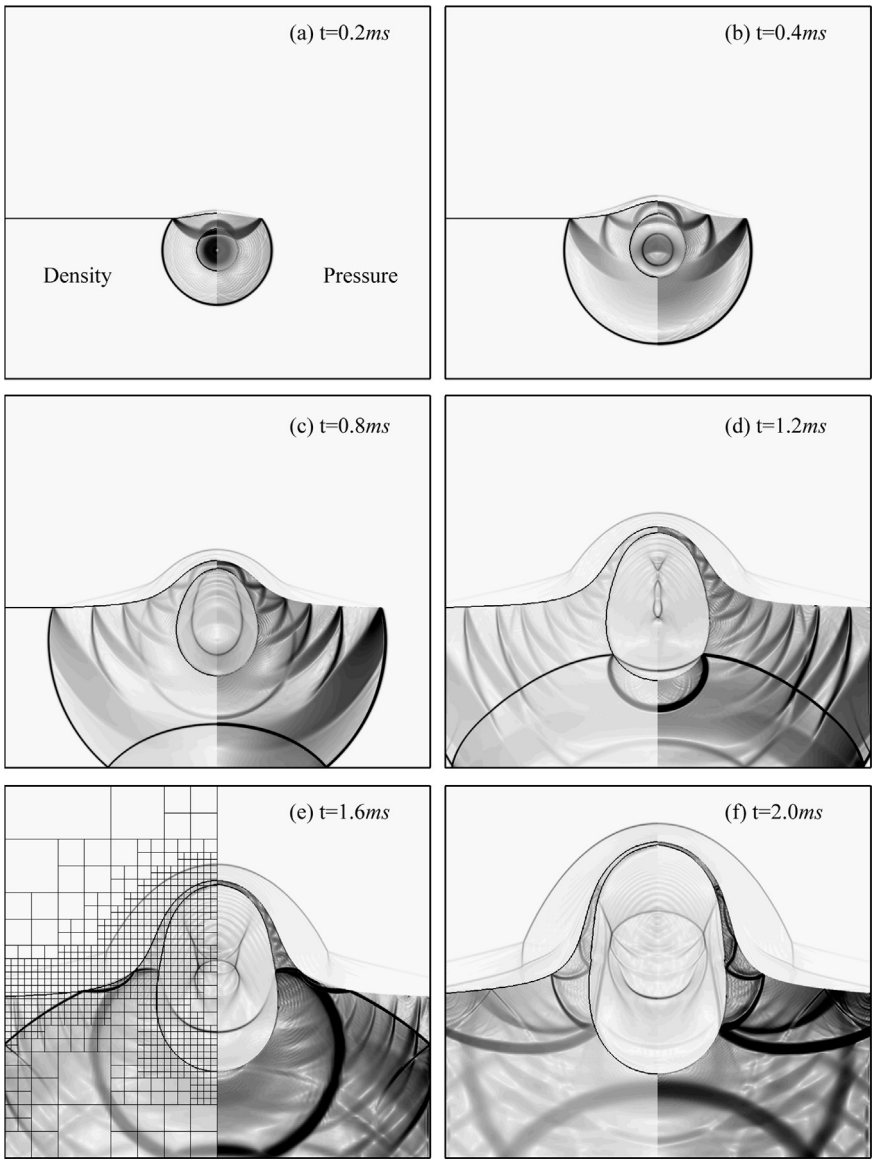

Fig. 24. Underwater explosion case: Schlieren-type image for density (the left half) and pressure (the right half) at different selected time instances with an effective resolution 1024×1024. A specific MR representation at $t = 1.6$ ms is shown in subplot (e).

$$\begin{cases} (\rho = 1.2 \times 10^{-3},\ u = 0,\ v = 0,\ p = 1,\ \gamma = 1.4) & \text{air,} \\ (\rho = 1,\ u = 0,\ v = 0,\ p = 1,\ \gamma = 7.15) & \text{water,} \\ (\rho = 1.25,\ u = 0,\ v = 0,\ p = 10\,000,\ \gamma = 1.4) & \text{gaseous bubble.} \end{cases} \tag{34}$$

Unlike the previous investigations that used a stiff gas equation of state for water, we employ the Tait's EOS.

In this test, we define a rectangular domain $(x, y) \in [0, 4] \times [0, 4]$ with only 1 block at the coarser level. The initial condition is composed of a horizontal air–water interface at $y = 1.5$ and a circular gaseous bubble of the radius 0.12 in water with center $(2, 1.2)$. The reference parameters are given by the state of water at 1 atmosphere and 1 m. The boundary conditions are a solid-wall boundary at the bottom and all outflow boundaries with zero gradient at the remaining sides. With a maximum refinement level of $L_{max} = 6$, the effective grid resolution at the finest level is 1024×1024.

Fig. 24 shows Schlieren-type images of density and pressure at several different time instances. The results are in good agreement with the works of Hu et al. [25] and Shyue [54]. We clearly observe the reflected waves from the water–air surface mentioned by Hu et al. [25]. Through extending the computation time to 2.0 ms, we capture more details in the latter process (1.2 ms–2.0 ms). A series of reflected waves travel along the water filament and accelerate the free surface motion, Fig. 24(f). In Fig. 24(e), we can see that, since almost all strong waves occur only near the interface, the MR-SIM

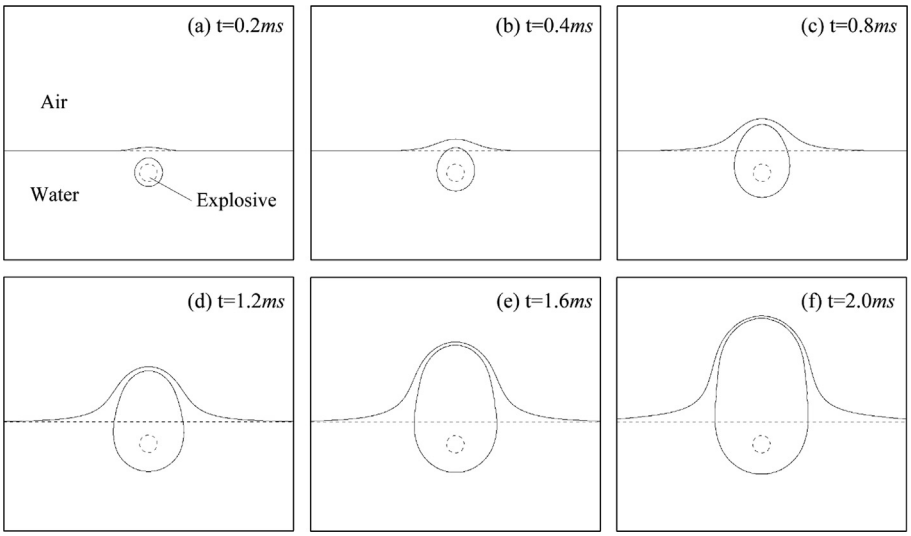

Fig. 25. Interface locations for the run shown in Fig. 24, where the dashed line in each subplot is the initial location of the horizontal and circular bubble interfaces at time $t = 0$.

approach shows a very good capability of capturing small scales. Fig. 25 shows the interface positions corresponding to the time instances in Fig. 24. The entire simulation requires less than 250 MBytes memory space.

6. Conclusions

In this paper, we have proposed a table-based storage-and-operation-splitting pyramid data structure for adaptive MR methods, which allows direct search among blocks. In this data structure, non-overlapping computational data are grouped into fine-grained data packages and stored within a memory pool. Partially overlapping blocks containing several packages are distributed into threads for parallel computing, where the overlapping part shares the same computational data in memory. We combine this efficient pyramid data structure with a sharp-interface model and an adaptive multi-resolution approach (called MR-SIM) for the simulation of multi-phase flows. In this new approach a basic multi-resolution representation is generated first according to the interface position. Second, a standard multi-resolution analysis is performed successively on all single-phase leaves from the coarsest to the finest level to complete the final refinement. This two-step approach requires that all ghost cell data and the level set only need to be stored in multi-phase blocks at the finest level, which makes the application of sharp interface methods straightforward. The actual memory efficiency of MR-SIM is approximately the same as that of smooth-interface based MR schemes. We introduce a tailored narrow-band technique to restrict all sharp-interface relevant operations to a neighborhood of the interface. A number of numerical examples in two dimensions are simulated with very high effective resolution and compared to results of previous works. The obtained results suggest that the present modeling approach exhibits superior memory efficiency and robustness on terms of reproducing physically meaningful solutions for very complex interface structures. Different from other adaptive results in the reference literature, MR-SIM captures more accurately small-scale details not only of the individual phases but also of the interfaces. Note that although numerical examples discussed in this paper are 2D, giving the purpose of facilitating presentation, assessment and discussion, the method has been extended in a straightforward way to 3D.

Acknowledgement

The first author gratefully acknowledges the support by the TUM IGSSE (Technische Universität München International Graduate School of Science and Engineering) for this work within the Project 6.07.

References

[1] G.S. Jiang, C.W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1996) 202–228.
[2] X.Y. Hu, Q. Wang, N.A. Adams, An adaptive central-upwind weighted essentially non-oscillatory scheme, J. Comput. Phys. 229 (2010) 8952–8965.
[3] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (3) (1984) 484–512.
[4] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.
[5] F. Miniati, P. Colella, Block structured adaptive mesh and time refinement for hybrid, hyperbolic + N-body systems, J. Comput. Phys. 227 (2007) 400–430.
[6] P. MacNeice, K.M. Olson, C. Mobarry, R. de Fainchtein, C. Packer, PARAMESH: A parallel adaptive mesh refinement community toolkit, Comput. Phys. Commun. 126 (2000) 330–354.
[7] A. Harten, Adaptive multiresolution schemes for shock computations, J. Comput. Phys. 115 (1994) 319–338.

[8] B.L. Bihari, A. Harten, Multiresolution schemes for the numerical solution of 2-D conservation laws I, SIAM J. Sci. Comput. 18 (1997) 315–354.
[9] A. Cohen, S.M. Kaber, S. Müller, M. Postel, Fully adaptive multiresolution finite volume schemes for conservation laws, Math. Comput. 72 (2003) 183–225.
[10] O. Roussel, K. Schneider, A. Tsigulin, H. Bockhorn, A conservative fully adaptive multiresolution algorithm for parabolic PDEs, J. Comput. Phys. 188 (2003) 493–523.
[11] M.O. Domingues, S.M. Gomes, O. Roussel, K. Schneider, Adaptive multiresolution methods, in: ESAIM Proc., vol. 34, EDP Sciences, 2011, pp. 1–96.
[12] R. Deiterding, M.O. Domingues, S.M. Gomes, O. Roussel, K. Schneider, Adaptive multiresolution or adaptive mesh refinement? A case study for 2D Euler equations, in: ESAIM Proc., vol. 29, 2009, pp. 28–42.
[13] K. Schneider, O.V. Vasilyev, Wavelet methods in computational fluid dynamics, Annu. Rev. Fluid Mech. 42 (2010) 473–503.
[14] A. Cohen, Wavelet methods in numerical analysis, Handb. Numer. Anal. 7 (2000) 417–711.
[15] S. Müller, Adaptive Multiscale Schemes for Conservation Laws, vol. 27, Springer, 2003.
[16] S. Osher, R. Sanders, Numerical approximations to nonlinear conservation laws with locally varying time and space grids, Math. Comput. 43 (1983) 321–336.
[17] C. Dawson, R. Kirby, High resolution schemes for conservation laws with locally varying time steps, SIAM J. Sci. Comput. 22 (2001) 2256–2281.
[18] M.O. Domingues, S.M. Gomes, O. Roussel, K. Schneider, An adaptive multiresolution scheme with local time stepping for evolutionary PDEs, J. Comput. Phys. 227 (2008) 3758–3780.
[19] M.O. Domingues, S.M. Gomes, O. Roussel, K. Schneider, Space–time adaptive multiresolution methods for hyperbolic conservation laws: Applications to compressible Euler equations, Appl. Numer. Math. 59 (2009) 2303–2321.
[20] B. Hejazialhosseini, D. Rossinelli, M. Bergdorf, P. Koumoutsakos, High order finite volume methods on wavelet-adapted grids with local time-stepping on multicore architectures for the simulation of shock-bubble interactions, J. Comput. Phys. 229 (2010) 8364–8383.
[21] L.H. Han, T. Indinger, X.Y. Hu, N.A. Adams, Wavelet-based adaptive multi-resolution solver on heterogeneous parallel architecture for computational fluid dynamics, Comput. Sci. Res. Dev. 26 (2011) 197–203.
[22] D. Rossinelli, B. Hejazialhosseini, M. Bergdorf, P. Koumoutsakos, Wavelet-based adaptive solvers on multi-core architectures for the simulation of complex systems, Concurr. Comput., Pract. Exp. 23 (2011) 172–186.
[23] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), J. Comput. Phys. 152 (1999) 457–492.
[24] X.Y. Hu, B.C. Khoo, N.A. Adams, F.L. Huang, A conservative interface method for compressible flows, J. Comput. Phys. 219 (2006) 553–578.
[25] X.Y. Hu, N.A. Adams, G. Iaccarino, On the HLLC Riemann solver for interface interaction in compressible multi-fluid flow, J. Comput. Phys. 228 (2009) 6572–6589.
[26] J. Glimm, E. Isaacson, D. Marchesin, O. McBryan, Front tracking for hyperbolic systems, Adv. Appl. Math. 2 (1981) 91–119.
[27] S. Zelinka, E. Praun, C. Ohazama, et al., Large-scale image processing using mass parallelization techniques, US Patent 7,965,902, 2011.
[28] G. Contreras, M. Martonosi, Characterizing and improving the performance of Intel Threading Building Blocks, in: 2008 IEEE International Symposium on Workload Characterization (IISWC 2008), 2008, pp. 57–66.
[29] X.Y. Hu, N.A. Adams, Scale separation for implicit large eddy simulation, J. Comput. Phys. 230 (19) (2011) 7240–7249.
[30] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[31] M. Sussman, E. Fatemi, P. Smereka, S. Osher, An improved level set method for incompressible two-phase flows, Comput. Fluids 27 (1998) 663–680.
[32] X.Y. Hu, B.C. Khoo, An interface interaction method for compressible multifluids, J. Comput. Phys. 198 (2004) 35–64.
[33] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1994) 146–159.
[34] E. Lauer, X.Y. Hu, S. Hickel, N.A. Adams, Numerical modelling and investigation of symmetric and asymmetric cavitation bubble dynamics, Comput. Fluids 69 (2012) 1–19.
[35] F. Coquel, Y. Maday, S. Müller, M. Postel, Q.H. Tran, New trends in multiresolution and adaptive methods for convection-dominated problems, in: ESAIM Proc., vol. 29, EDP Sciences, 2009, pp. 1–7.
[36] M. Sussman, A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, An adaptive level set approach for incompressible two-phase flows, J. Comput. Phys. 148 (1999) 81–124.
[37] R.R. Nourgaliev, T.N. Dinh, T.G. Theofanous, Adaptive characteristics-based matching for compressible multifluid dynamics, J. Comput. Phys. 213 (2006) 500–529.
[38] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, A PDE-based fast local level set method, J. Comput. Phys. 155 (2) (1999) 410–438.
[39] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, J. Comput. Phys. 43 (1981) 357–372.
[40] P.D. Lax, Weak solutions of nonlinear hyperbolic equations and their numerical computation, Commun. Pure Appl. Math. 7 (1954) 159–193.
[41] A. Harten, P.D. Lax, B.V. Leer, On upstream differencing and Godunov-type schemes for hyperbolic conservation laws, SIAM Rev. 25 (1983) 35–61.
[42] B. Einfeldt, On Godunov-type methods for gas dynamics, SIAM J. Numer. Anal. 25 (1988) 294–318.
[43] E.F. Toro, M. Spruce, W. Speares, Restoration of the contact surface in the HLL-Riemann solver, Shock Waves 4 (1994) 25–34.
[44] M.H. Chung, A level set approach for computing solutions to inviscid compressible flow with moving solid boundary using fixed Cartesian grids, Int. J. Numer. Methods Fluids 33 (2001) 1121–1151.
[45] T.G. Liu, B.C. Khoo, K.S. Yeo, Ghost fluid method for strong shock impacting on material interface, J. Comput. Phys. 190 (2003) 651–681.
[46] G.H. Miller, P. Colella, A conservative three-dimensional Eulerian method for coupled solid-fluid shock capturing, J. Comput. Phys. 183 (1) (2002) 26–82.
[47] K.K. So, X.Y. Hu, N.A. Adams, Anti-diffusion interface sharpening technique for two-phase compressible flow simulations, J. Comput. Phys. 231 (2012) 4304–4323.
[48] J.F. Haas, B. Sturtevant, Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities, J. Fluid Mech. 181 (1987) 41–76.
[49] T.-J. Chen, C.H. Cooke, On the Riemann problem for liquid or gas–liquid media, Int. J. Numer. Methods Fluids 18 (1994) 529–541.
[50] J.E. van Aanhold, G.J. Meijer, P.P.M. Lemmen, Underwater shock response analysis of a floating vessel, Shock Vib. 5 (1998) 53–59.
[51] C.-H. Chang, M.-S. Liou, A robust and accurate approach to computing compressible multiphase flow: Stratified flow model and AUSM$^+$-up scheme, J. Comput. Phys. 225 (2007) 840–873.
[52] C.-H. Chang, X. Deng, T.G. Theofanous, Direct numerical simulation of interfacial instabilities: A consistent, conservative, all-speed, sharp-interface method, J. Comput. Phys. 242 (0) (2013) 946–990.
[53] J. Grove, R. Manikoff, Anomalous reflection of shock wave at a fluid interface, J. Fluid Mech. 219 (1990) 313–336.
[54] K.M. Shyue, A wave-propagation based volume tracking method for compressible multicomponent flow in two space dimensions, J. Comput. Phys. 215 (2006) 219–244.