Journal of Computational Physics 341 (2017) 447–473

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A novel partitioning method for block-structured adaptive meshes

Lin Fu, Sergej Litvinov [1], Xiangyu Y. Hu [*], Nikolaus A. Adams

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

ARTICLE INFO

Article history:
Received 15 December 2015
Received in revised form 26 September 2016
Accepted 3 November 2016
Available online 18 November 2016

Keywords:
Lagrangian particle method
Smoothed-particle hydrodynamics
Multi-resolution cell-linked list
Dynamic ghost particle method
Adaptive mesh refinement
Grid partitioning

ABSTRACT

We propose a novel partitioning method for block-structured adaptive meshes utilizing the meshless Lagrangian particle concept. With the observation that an optimum partitioning has high analogy to the relaxation of a multi-phase fluid to steady state, physically motivated model equations are developed to characterize the background mesh topology and are solved by multi-phase smoothed-particle hydrodynamics. In contrast to well established partitioning approaches, all optimization objectives are implicitly incorporated and achieved during the particle relaxation to stationary state. Distinct partitioning sub-domains are represented by colored particles and separated by a sharp interface with a surface tension model. In order to obtain the particle relaxation, special viscous and skin friction models, coupled with a tailored time integration algorithm are proposed. Numerical experiments show that the present method has several important properties: generation of approximately equal-sized partitions without dependence on the mesh-element type, optimized interface communication between distinct partitioning sub-domains, continuous domain decomposition which is physically localized and implicitly incremental. Therefore it is particularly suitable for load-balancing of high-performance CFD simulations.

1. Introduction

Massively parallel computing is essential for Computational Fluid Dynamics (CFD) to cope with flow simulations involving complex geometries or a wide spectrum of length scales [1]. To reduce computational costs, unstructured or adaptive structured mesh topologies are frequently employed as state-of-the-art mesh discretization methods [2]. Refined mesh elements are distributed at flow regions of interest or regions containing discontinuities. Such nonuniform mesh topologies are the consequence of the compromise between computational accuracy and efficiency [3]. However, partitioning problems arise with the utilization of such locally refined mesh topologies when the number of processor cores increases in massively parallel computing environments. The optimization strategy for load-balancing and inter-processor communication becomes the critical bottleneck and limits the computational performance. A partitioning method should accomplish approximately equal-sized domain decomposition with minimum neighbor communication patterns as it reduces processor operation idle time and inter-processor communication time. Moreover, each individual sub-domain is expected to be physically localized and continuous since it facilitates data management and reduces communications [4]. Partitioning operations for parallel computing can be categorized into static partitioning and dynamic partitioning. For the latter, additional requirements are

* Corresponding author.
E-mail addresses: lin.fu@tum.de (L. Fu), sergej.litvinov@aer.mw.tum.de (S. Litvinov), xiangyu.hu@tum.de (X.Y. Hu), nikolaus.adams@tum.de (N.A. Adams).
[1] Current address: Computational Science and Engineering Laboratory, ETH Zürich, Clausiusstrasse 33, CH-8092, Switzerland.

http://dx.doi.org/10.1016/j.jcp.2016.11.016
0021-9991/

that small topology changes should only result in slight modification of the partitioning (implicitly incremental) and the repartitioning time cost is minimized [5].

Classical partitioning methods can be roughly classified as geometry-based and graph-based approaches [4,6]. Several open-source codes, such as Metis [7], have been developed to meet these principles. The former is based on the physical coordinates of the original background mesh and generally leads to very fast algorithms. The Recursive Coordinate Bisection (RCB) method recursively divides the computational domain into equal-sized sub-domains by cutting planes that are orthogonal to the coordinate axes and has been proposed by Berger and Bokhari [8] to partition meshes generated by Adaptive Mesh Refinement [9]. RCB is an efficient method to provide a basic partitioning for structured or unstructured meshes when the connectivity information between mesh elements is not at hand. However, the partitioning quality suffers from possible discontinuous sub-domains and inefficient communication requirements [4]. The key idea of Space-Filling Curve (SFC) [10,11] partitioning is to construct a linear sequence of mesh elements through mapping the higher-dimensional mesh onto one dimension. Partitioning is obtained by cutting the one-dimensional sequence into desired pieces. The advantages of this approach are that it is fast, robust and geometrically localized. It also provides an efficient data structure as all elements can be assigned a globally unique index, enabling direct access. Furthermore, physical adjacency more or less is preserved. However, similarly to RCB, communication needs cannot be controlled explicitly during the partitioning, and disconnected sub-domains may exist [4]. Nivarti et al. [12] develop an inexpensive algorithm to improve the spatial locality of SFC partitioning by sacrificing a certain degree of load balance. Despite the moderate partitioning quality, geometry-based partitioning approaches play important roles in applications without inherent connectivity information, such as for particle based simulations.

In terms of graph-based approaches, a graph, where vertices represent the mesh elements to be partitioned and edges stand for communications between elements, is generated to characterize the background mesh. Load-balancing optimization and edge-cut minimization are achieved via Recursive Spectral Bisection (RSB) [13] or a multilevel graph partitioning strategy [14,15] Thus, the detailed connectivity information of the partitioning mesh must be available. RSB generates subgraphs by exploiting the eigenvectors of the Laplacian matrix associated with the target graph. It is quite expensive due to the eigenvector calculation. The multilevel partitioning paradigm is much more efficient and mainly consists of three substages: (i) graph coarsening which generates a sequence of coarser graphs preserving the essential properties of the original graph; (ii) partitioning of the representative coarsest graph; and (iii) graph refinement (Kernighan–Lin [16] heuristics and its variant Fiduccia–Mattheyses algorithm [17] are widely employed) which improves the partition by interpolating the coarser partitioned graph back to the original one in combination with a fast local search algorithm [18]. These approaches have significant advantages over geometry-based approaches by generating high-quality domain decomposition with controllable interface communication, although they are more time consuming. However, classical graph-based approaches may suffer from the problem that the optimization objective function is not always adequate. The partition objective of minimizing the total communication message size is not directly equivalent to minimizing edge-cuts because edge-cuts are in general not proportional to the communication volume [19]. For many applications, the number of graph vertices located at the boundaries of partitioning sub-domains (boundary vertices) indeed characterizes the real communication cost much more accurately than the number of edge-cuts [20]. Boundary properties of partitioning sub-domain, e.g. aspect ratio [21], connectedness and smoothness [22], are also crucial in communication reduction [23]. This problem can be improved largely by introducing the Hypergraph partitioning concept, which encodes the communication volume better, with higher running time [24]. With respect to dynamic load-balancing which introduces additional optimization objectives, the problem arises in parallelizing the widely-used Kernighan–Lin [16] method on distributed-memory architecture and designing the multiple optimization metrics [25]. In such a scenario, graph-based diffusive partitioners [26,20] are generally applied as they produce partitions, that have more implicit incrementality and higher-quality than those from classical multilevel partitioning strategies. A drawback of these graph-based partitioners is that strictly-connected partitions cannot be guaranteed in principle [20]. Nevertheless, graph-based partitioning methods are widely accepted in well-established software [7].

We will explain below, that the partitioning result has high similarity to a stationary multi-phase fluid. Based on this observation, we propose a new partitioning method based on multi-phase particle simulation. Using the topology information from a background mesh, a set of physically motivated model equations is formulated and solved by a meshless multiphase Smoothed Particle Hydrodynamics method [27]. While most graph-based partitioning approaches explicitly optimize the communication by minimizing the edge-cuts, we seek to minimize the number of boundary vertices and optimize the partitioning-sub-domain shapes. All optimization objectives are implicitly incorporated and optimized during the particle evolution process. Our proposed partitioning method guarantees connected partitioning, optimized interface communication, good load-balancing and possesses the implicitly incremental property. Moreover, it does not depend on the specific mesh element type and can be straightforwardly extended to unstructured mesh partitioning.

2. Physically motivated models

2.1. Mathematical concepts

For parallel simulations, the load-balancing requirement can be formulated as a problem in graph theory. Considering an undirected graph $G = (V, E)$, where V denotes a set of vertices corresponding to the computation units and E denotes a

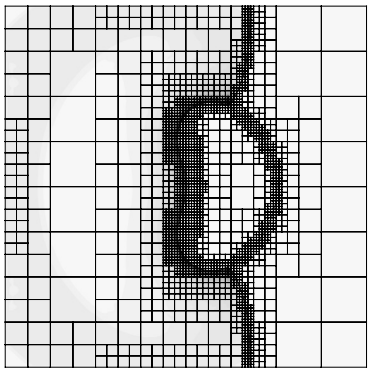


Fig. 1. Sketch of a block-structured mesh generated by multi-resolution analysis [2] with maximum tree level 6. Each mesh element represents a grid-block consisting typically of hundreds of cells in two-dimensions.

set of edges corresponding to the communications between computation units, the objective is to find subsets of vertices $V_1, V_2, \cdots, V_n$ which partitions graph G into n disjoint subgraphs and has the following properties [28,19]

(i) $V_1 \cup V_2 \cdots \cup V_n = V$ and $V_i \cap V_j = \emptyset$ with $i \neq j$;
(ii) $|V_i| \approx \lceil \frac{|V|}{n} \rceil,\ i = 1, \cdots, n$;
(iii) $\sum_{i<j} E_{ij}$ is minimized, where $E_{ij} = \{\{u, v\} \in E | u \in V_i \bigwedge v \in V_j\}$.

More generally, the graph vertices and edges can be assigned with attribute value w, resulting in a weighted graph, which we do not further consider below.

2.2. Basic idea

In the following context, we refer to the block-structured nonuniform mesh as in Fig. 1, in which a mesh element represents a computation unit within a partitioning sub-domain, or a graph vertex. An adjacency relationship between two mesh elements located separately in a sub-domain pair implies a communication element within a graph edge. Communication occurs at the subdomain boundaries, and the amount is proportional to the mesh element number in the neighborhood of subdomain boundaries.

With the observation that the load-balancing partitioning outcome has high analogy to the equilibrium of multi-phase fluid [2], a Lagrangian particle based multi-phase problem can be defined accordingly which relaxes to an equilibrium state with the following properties:

(i) The particle distribution exactly represents the mesh topology. Each mesh element contains the same number of particles which are uniformly distributed inside.
(ii) Each particle features a color function and particles of same color tend to concentrate. Furthermore, each phase, characterized by color, possesses the same number of particles.
(iii) The phase interface is sharp, convex, and the mesh element number close to the interface is optimized.
(iv) The particle evolution is localized and incremental.

If the equilibrium solution of the model equations satisfies these three properties, the remaining issue is to develop an algorithm to compute this equilibrium from a given initial configuration.

2.3. Governing equations

For isothermal compressible flows, the Lagrangian form of governing equations is

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{v}, \tag{1}$$

$$\frac{d\mathbf{v}}{dt} = -\mathbf{F_p} + \mathbf{F_v} + \mathbf{F_s} + \mathbf{F_f}, \tag{2}$$

where ρ is the fluid density, $\mathbf{v}$ is the velocity vector, t is the time and $\mathbf{F_p}$, $\mathbf{F_v}$, $\mathbf{F_s}$, $\mathbf{F_f}$ stand for the accelerations due to repulsive pressure force, viscous shear force, surface tension force and friction force respectively. This set of equations can be solved by the Smoothed Particle Hydrodynamics (SPH) method [27].

The same mass is assigned to all particles

$$m = 1. \tag{3}$$

Density can be calculated from summation over all neighboring particles rather than solving the continuity equation Eq. (1). Following the definition of $d_i = \sum_j W(r_{ij}, h)$ which is approximately the inverse of the specific particle volume [29], the particle average density is $m_i d_i$ and can be evaluated as

$$\rho_i = m_i \sum_j W(r_{ij}, h), \tag{4}$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, $r_{ij} = |\mathbf{r}_{ij}|$, h denotes the particle smoothing-length [29] and $W(r_{ij}, h)$ denotes the kernel function [27].

The discretization of the conservative repulsive pressure force is given as

$$\mathbf{F_p} = \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \frac{\partial W(r_{ij}, h)}{\partial r_{ij}} \mathbf{e}_{ij}, \tag{5}$$

where $\mathbf{e}_{ij} = \frac{\mathbf{r}_{ij}}{r_{ij}}$ is the normalized vector from particle i to j and p denotes the particle pressure. The arithmetic average of smoothing-lengths for particle pairs is used to ensure force anti-symmetry as

$$h = \frac{1}{2}(h_i + h_j). \tag{6}$$

In order to close the system of governing equations, the pressure must be defined by an Equation of State (EOS). For SPH methods, $p = \rho c^2$ or $p = p_0 \left[\left(\frac{\rho}{\rho_0} \right)^{\gamma} - 1 \right] + b$ are used [27]. Here, we define the pressure as

$$p = p_0 \frac{\rho^2}{\rho_t^2}, \tag{7}$$

which incorporates the target density ρ_t (the definition of ρ_t will be given later) and a constant reference pressure p_0. Substituting Eq. (7) into Eq. (5), we obtain

$$\mathbf{F_p} = \sum_j m_j \left(\frac{p_0}{\rho_{t,i}^2} + \frac{p_0}{\rho_{t,j}^2} \right) \frac{\partial W(r_{ij}, h)}{\partial r_{ij}} \mathbf{e}_{ij}. \tag{8}$$

In standard SPH, the accurate density summation depends on sufficient number of particle neighbors with respect to the kernel function support [27]. With the introduction of Eq. (7), direct density calculation becomes unnecessary.

2.4. Target information

The particle target information can be obtained straightforwardly from the background mesh. The target smoothing-length, representing the local mesh resolution, is approximated as

$$h_t = \vartheta \sqrt[3]{volume} \approx \vartheta \max(\Delta x, \Delta y, \Delta z), \tag{9}$$

where ϑ is defined such that the particle interaction cut-off radius for the chosen kernel function equals the local mesh resolution.

The target density can be calculated by

$$\rho_t = \frac{\sum_{i=1}^{n} m_i}{volume}, \tag{10}$$

provided that n particles are assumed to be distributed uniformly in each mesh element.

Each particle is mapped onto one certain background mesh element according to physical coordinate information. For a specific particle i,

$$\begin{cases} h_i \approx h_{t,i} = h_t(\mathbf{r_i}), \\ \rho_{t,i} = \rho_t(\mathbf{r_i}). \end{cases} \tag{11}$$

The repulsive force $\mathbf{F_p} = \frac{\nabla p}{\rho}$ depends on density and pressure gradient. If the particle distribution does not represent the mesh topology, i.e. $\rho \neq \rho_t$ and thus $p = p_0 \frac{\rho^2}{\rho_t^2}$ is not constant throughout the computational domain, particles will be

driven by the repulsive force produced by the pressure gradient. For $\rho = \rho_t$ everywhere, the pressure gradient vanishes and a relaxed equilibrium configuration is reached. In this way, the first partitioning design target is achieved.

The data required from the background mesh topology are volume and element location, both of which are basic mathematical properties of the specific type of elements and can be calculated in a fairly simple way. Consequently, our model equations are independent of mesh element types, such as structured mesh, adaptive unstructured mesh or hybrid mesh, and remain generic to a large extent.

2.5. Kernel function without paring instability

Considering smoothing kernels which denote the integral function based on discrete points, well known B-spline functions of different orders, e.g. cubic spline and quartic spline, are widely accepted [27]. However, gradient evaluation based on these kernel functions can produce numerical artifacts. The first-order gradients of these kernels exhibit a bell shape, generating a maximum negative value at $\frac{r}{h} \approx \frac{2}{3}$. When the ratio of smoothing-length to average particle spacing is large, particles tend to accumulate within the hump and develop the so-called paring instability [27].

Yang et al. [30] propose to use a hyperbolic-shaped kernel function with non-negative second-order derivative within the entire support as

$$W(r_{ij}, h) = \alpha \begin{cases} s^3 - 6s + 6, & 0 \le s < 1, \\ (2-s)^3, & 1 \le s < 2, \\ 0, & 2 \le s, \end{cases} \tag{12}$$

where $s = \frac{r_{ij}}{h}$ and α is a normalization

$$\alpha = \begin{cases} \frac{1}{3\pi h^2}, & 2D, \\ \frac{15}{62\pi h^3}, & 3D. \end{cases} \tag{13}$$

This kernel successfully avoids paring instability while satisfying the conventional requirements of a kernel function. In the following, we adopt this kernel for the force calculations.

2.6. Surface tension model

The classical metric, i.e. the number of edge-cuts between different partitioning sub-domains, is not a good measure for the real communication costs [19]. Partitioning sub-domains with convex boundary shapes, small aspect ratios and strict connectedness are preferred with respect to communication reduction [23,31]. Based on these observations, the objective is to optimize the area of partitioning-sub-domain boundaries which leads to convex shapes. This objective, i.e. the third design target, can be achieved physically by the introduction of a suitable surface tension model.

Surface tension originates from the cohesion of like molecules and thus promotes the coherence of distinct flow phases. With the effects of the surface tension force, the interface surface area tends to become minimum. This property naturally fulfills the optimization objective.

Although the interface curvature can be derived from the particle color function in SPH [29], we propose a simplified model which avoids a direct calculation of curvature by modifying the computation of the repulsive force.

The acceleration due to the inter-particle mutual repulsive pressure force can be redefined as

$$\mathbf{f_p} = \beta m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \frac{\partial W(r_{ij}, h)}{\partial r_{ij}} \mathbf{e}_{ij}, \tag{14}$$

where β is a constant representing the surface tension coefficient and is defined as

$$\beta = \begin{cases} 1, & C_i = C_j, \\ \sigma, & \text{otherwise}, \end{cases} \tag{15}$$

where C_k denotes the color function of particle k. σ determines the surface tension strength, and $\sigma = 4$ is adopted throughout all applications of our method. The total acceleration due to repulsive pressure force and surface tension effects for each particle is

$$\mathbf{F_p} = \sum_j \mathbf{f_p}. \tag{16}$$

Particles near an interface segment with large curvature are expected to have more neighboring particles of different color and thus subject to a larger repulsive force from distinct phases. In this way, the interface-curvature effect is implicitly incorporated.

2.7. Viscous and friction model

The particle acceleration induced by shear forces is approximated as

$$\mathbf{F_v} = \sum_j \frac{2\eta_i\eta_j}{\eta_i+\eta_j} m_j \left(\frac{1}{\rho_i^2}+\frac{1}{\rho_j^2}\right) \frac{\partial W(r_{ij},h)}{\partial r_{ij}} \frac{(\mathbf{v}_{ij}+\mathbf{e}_{ij}.\mathbf{v}_{ij}\mathbf{e}_{ij})}{r_{ij}}, \tag{17}$$

where the dynamic viscosity is $\eta = \rho\nu$, and $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ is the relative velocity vector from particle i to particle j [29]. The particle density is approximated as

$$\rho \approx \rho_t \tag{18}$$

for simplicity. The kinematic viscosity coefficient is defined as

$$\nu \sim 0.1 r_c |\mathbf{v}|, \tag{19}$$

where r_c is the kernel cut-off radius, leading to a local Reynolds number $\mathcal{O}(10)$.

The evolution of the particle system is determined by the background potential force and the viscous force. The potential force relaxes the particles towards the target configuration whereas the viscous force damps the velocity fluctuations. Mostly, the viscous effect is insufficient to damp all particle kinetic energies, making it difficult for the whole particle system to relax fully to steady state. Therefore, we propose a further damping at each time step by resetting the particle velocity

$$\mathbf{v}_n = \mathbf{0}, \tag{20}$$

after updating the particle positions.

2.8. Time integration method

For SPH, the Velocity-Verlet algorithm is widely utilized as time integration method [32]. This scheme is second-order accurate and reversible in time without viscous calculation. However it is quite time consuming because the total force for all particles is accumulated twice. As we seek for a steady solution, second-order accuracy is not necessary. We exploit a simple time integration method as

$$\begin{aligned} \tilde{\mathbf{v}}_{n+\frac{1}{2}} &= \mathbf{v}_n + \tfrac{1}{2}\mathbf{a}_n \Delta t, \\ \mathbf{v}_{n+\frac{1}{2}} &= \tilde{\mathbf{v}}_{n+\frac{1}{2}} + \tfrac{1}{2}\tilde{\mathbf{a}}_{n+\frac{1}{2}} \Delta t, \\ \mathbf{r}_{n+1} &= \mathbf{r}_n + \mathbf{v}_{n+\frac{1}{2}} \Delta t, \end{aligned} \tag{21}$$

where the acceleration $\mathbf{a}_n$ is calculated from the repulsive force $\mathbf{F_p}$ and $\mathbf{F_s}$, and the predicted velocity $\tilde{\mathbf{v}}_{n+\frac{1}{2}}$ at the mid-point is updated in the first step. The viscous force $\mathbf{F_v}$ is computed utilizing the predicted velocity field and $\tilde{\mathbf{a}}_{n+\frac{1}{2}}$ is obtained from this viscous force only. After that, a modified velocity field is defined in the second step, and the new particle coordinates are updated according to the modified velocity field. If the viscosity coefficient is zero, the second step is skipped and $\mathbf{v}_{n+\frac{1}{2}} = \tilde{\mathbf{v}}_{n+\frac{1}{2}}$.

For numerical stability, the time step Δt is constrained by a body force criterion

$$\Delta t \leq 0.25\sqrt{\frac{r_c}{|\mathbf{a}|}}, \tag{22}$$

a Courant–Friedrichs–Lewy (CFL) criterion [33] assuming that the sound speed $c \approx 40|\mathbf{v}|_{\max}$, and a viscous criterion

$$\Delta t \leq \min\left(\frac{1}{40}\frac{r_c}{|\mathbf{v}|}, 0.125\frac{r_c^2}{\nu}\right). \tag{23}$$

With the CFL constraint, in each iteration, the particle position is updated by a small step, which is proportional to the local smoothing length. Therefore, the fourth objective, i.e. the locality and incremental property, is satisfied.

3. Numerical algorithms

3.1. Initial condition

There are several basic ways to distribute the particles for the initial condition, e.g. uniformly or randomly. For our case, particles with the same color must be continuously distributed and this requirement can be achieved by two steps.

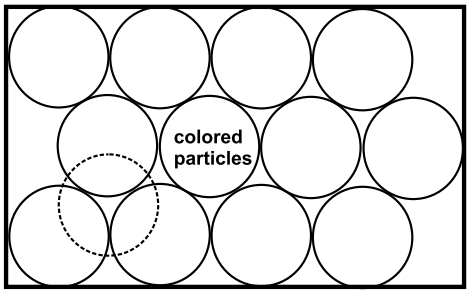


Fig. 2. Sketch for close packing methods. The circle with dotted line illustrates a projection of sphere in the third direction. Each circle is assigned with the same number of colored particles in the second step.

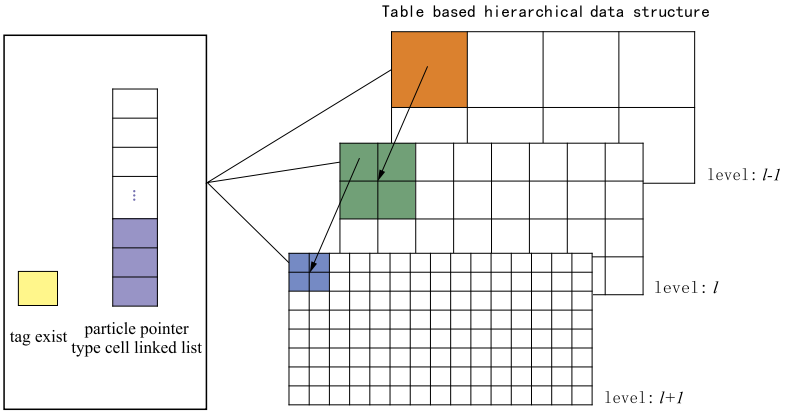


Fig. 3. Illustration of the multi-resolution cell-linked list for two dimensions. At each level, a table based data structure, which consists of an identifier *exist* and a cell-linked list, is initialized and dynamically updated throughout the simulation. The orange table element is the parent of the green ones while four blue table elements are the children of the green one. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The first step is to roughly divide the physical domain into a desired number of partition sub-domains. We propose to exploit the geometry based close packing technique [34], which densely arranges an initial set of congruent circles (spheres in 3D) in a regular domain. There are two main types of lattice arrangements, i.e. face-centered cubic (fcc) for which every third layer is the same and hexagonal close-packed (hcp) for which every other layer is identical, to achieve the highest density packing. Here, the hexagonal close-packing method is adopted and the physical coordinates of the circle centers or the sphere centers can be generated by

$$\mathbf{r}_{center} = \begin{bmatrix} 2i + 1 + \mathrm{mod}(j, 2) + \mathrm{mod}(k, 2) \\ \sqrt{3}j + 1 + \frac{\sqrt{3}}{3}\mathrm{mod}(k, 2) \\ \frac{2\sqrt{6}}{3}k + 1 \end{bmatrix} R, \tag{24}$$

where the index (i, j, k) starts with $(0, 0, 0)$, and R represents the radius of the circle or the sphere. For the second step, an equal number of particles with like color are randomly distributed within the corresponding circle or sphere as shown in Fig. 2. By this initial condition and the surface tension model, the second design target can be satisfied.

3.2. Fast neighbor search method

Cell-linked list [35] and Verlet list [32], which provide access to neighbor partners with $\mathcal{O}(1)$ operations, are two typical data structures to reduce the complexity for simulations with constant kernel cut-off radius. With the cell-linked list approach, the computational domain is divided into uniformly-distributed cells with edge length equal to the kernel cut-off radius. Each particle is mapped onto the corresponding cell-linked list by a pointer to the particle. By traversing the cell-linked list, where a certain particle is recorded, and the adjacent ones, all neighbor particles can be found rapidly. A Verlet list explicitly stores all the neighbor particles but strongly depends on an efficient construction method. A buffer is generally introduced to prevent frequent reconstruction of the Verlet list [32]. However, both of these methods are quite inefficient for resolution-adaptive particle simulations with variable kernel cut-off radii. Tree-based data structures have been proposed to address this problem [36].

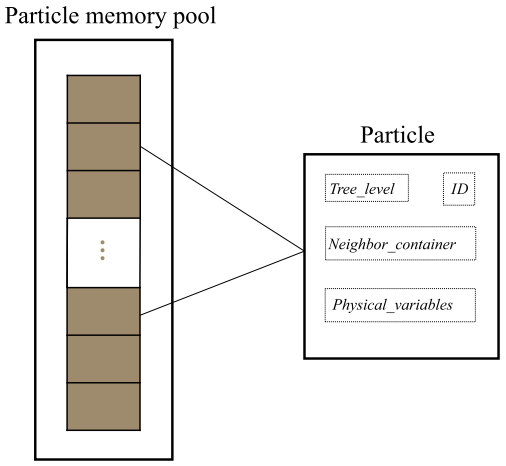


Fig. 4. Illustration of the particle data structure. *Physical_variables* denote the important particle properties, such as mass, density, pressure, smoothing-length. The particle data memories and the related operations, i.e. allocation and release, are managed by a "memory pool" [37].

We develop a multi-resolution neighbor search algorithm in combination with the Verlet list without margin to enable efficient neighbor access. Motivated by scatter-type sampling, the particle pairs are mutual neighbors when

$$r_{ij} \leq \max(r_{c,i}, r_{c,j}) \tag{25}$$

is satisfied. r_c is a function of physical space $r_c = r_c(\mathbf{r})$. The maximum multi-resolution level can be determined by

$$L_{max} = \log_2 \left(\frac{r_{c,\max}}{r_{c,\min}} \right) + 1. \tag{26}$$

The edge length of a cell on level l is

$$\Delta_l = \frac{r_{c,\max}}{2^l}, \quad l = 0, \cdots, L_{max} - 1\ . \tag{27}$$

At each level, a cell-linked list table, where the particle address pointers are recorded, is generated as shown in Fig. 3. The corresponding identifier *exist* is designed to record the status of a certain cell-linked list.

Fig. 4 shows the basic data structure for a particle, in which *Neighbor_container* is a pointer-type dynamic array to construct the Verlet list and ID is the globally unique particle identifier. The particle property *Tree_level* is defined as the level l at which

$$\frac{r_{c,\max}}{2^{l+1}} < r_c \leq \frac{r_{c,\max}}{2^l}. \tag{28}$$

Tree_level represents which level a certain particle belongs to and is unchanged within one time step. A specific particle is mapped onto the cell-linked list at level $l = Tree\_level$ with index (i, j)

$$(i, j) = \left(\text{int}(\frac{r_x}{\Delta_l}), \text{int}(\frac{r_y}{\Delta_l}) \right). \tag{29}$$

From the finest to the coarsest level, all members in the cell-linked list at level $l + 1$ are additionally appended to the parent cell-linked list at level l. The parent cell-linked list index (i, j) can be derived from the children cell-linked list index (m, n) by

$$(i, j)_l = \left(\text{int}(\frac{m}{2}), \text{int}(\frac{n}{2}) \right)_{l+1}; \tag{30}$$

The children cell-linked list index can also be derived from its parent by

$$(m, n)_{l+1} = (2i + \delta_1, 2j + \delta_2)_l\ ,\ \delta_1, \delta_2 = (0, 1). \tag{31}$$

Subsequently, the identifier *exist* is marked as 1 if the corresponding cell-linked list is occupied and 0 if it is empty. Each particle pointer is not only registered in the cell-linked list at *Tree_level* but also recorded at all coarser levels. Consequently, if a particle at *Tree_level* finds neighbors by searching the cell list where it is recorded and the adjacent ones, it can find particles at the same level and particles projected from the finer level. Particles can directly access others with equal or smaller kernel cut-off radius, but not those with larger kernel cut-off.

Algorithm 1 Fast neighbor search and Verlet list construction.

```
determine parameters and initialize multi-resolution based hierarchical data structure;
for all particles do
    define Tree_level through Eq. (28);
    map current particle to cell-linked list (i, j) on Tree_level through Eq. (29);
end for
for level from (L_max - 2) → 0 do
    for all cell-linked lists at current level do
        current cell-linked list ← current cell-linked list ∪ children cell lists at (level + 1);
    end for
end for
update the tag system exist;
for level from 0 → (L_max - 1) do
    for all cell-linked lists at current level ⋀ exist = 1 do
        for all particles in current cell-linked list do
            if Tree_level = current level then
                traverse all particles in current and adjacent lists, define as target particle;
                if Tree_level of target particle = current level then
                    Verlet list of current particle ⇐ target particle if satisfying Eq. (25);
                else
                    Verlet list of current particle ⇐ target particle if satisfying Eq. (25);
                    Verlet list of target particle ⇐ current particle if satisfying Eq. (25);
                end if
            end if
        end for
    end for
end for
```

As the final step, the multi-level cell-linked lists are traversed recursively from the coarsest to the finest level. For each particle registered in the cell list, if the tag *Tree_level* equals current loop level, then the particle neighbors can be detected in a similar way as that of cell-linked list technique with constant kernel cut-off radius, and the corresponding Verlet list is constructed at the same time. Note that particle pairs with the same *Tree_level* will access each other twice while only once with different values of *Tree_level*. Details are given in Algorithm 1.

3.3. Boundary condition

For particles in the vicinity of a computational domain border, the support region of kernel function is incomplete and adequate boundary conditions, which are physically meaningful and numerically stable, must be imposed. The boundary force particle method [38], the ghost particle method [39], the semi-analytical method [40] are classical techniques to provide the full kernel support. The ghost particle method with mirror-particle technique is believed to be problem independent and suitable for simple geometries. Image particles are reconstructed by reflection of inner particles at the domain boundaries. Symmetry and wall boundary condition can be enforced by defining the velocity of the ghost particles appropriately. However, for variable kernel cut-off radii which may differ by orders of magnitude, a general implementation of the ghost particle method is quite storage consuming, as the size of particle reflection region must correspond to the largest kernel cut-off radius. As remedy, we develop a dynamic ghost particle method, which temporally reconstructs the ghost particles for inner boundary particles.

As sketched in Fig. 5, all particles first are checked and marked as boundary particle if the minimum distance d from the particle to the nearest boundary is less than the kernel cut-off r_c. For each boundary particle, all neighbors in the inner-domain have already been found and recorded in the Verlet list. If a particle is not recorded as neighbor in the Verlet list, its mirrored image particle will also not contribute to density or force evaluation. Based on this idea, a temporary list of ghost particles for each boundary particle can be generated by mirroring the neighboring particles in the Verlet list, and the force contribution from these ghost particles are accumulated. Since this reconstruction is local and temporary, the amount of ghost particles depends on the individual kernel cut-off radius of each boundary particle.

3.4. Refinement and coarsening of particle number

In order to represent the background mesh topology well and to provide an appropriate number of neighboring particles, the total particle number involved in the simulation should be maintained on a certain level. The scale ratio is defined as

$$S = \frac{N_p}{N_e}, \tag{32}$$

where N_p, N_e represent the total number of particles and mesh elements, respectively. We set

$$S_t = \begin{cases} 4, & 2D, \\ 8, & 3D. \end{cases} \tag{33}$$

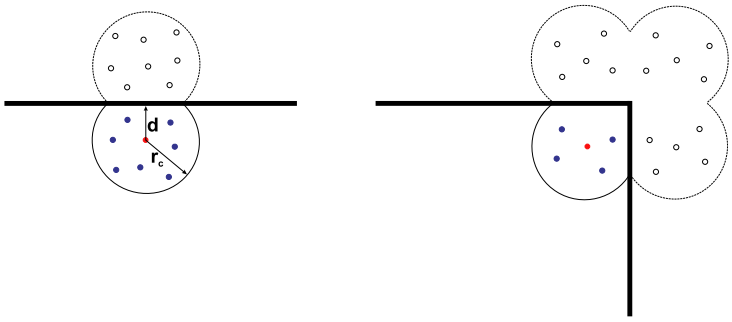


Fig. 5. Sketch for dynamic ghost particle method in two dimensions. The red particle denotes a specific boundary particle, and the blue ones are its neighbors inside the computational domain. The particles without colors are ghost particles constructed by current algorithm. The left panel indicates the boundary particle near a domain edge and the right panel near a domain corner. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

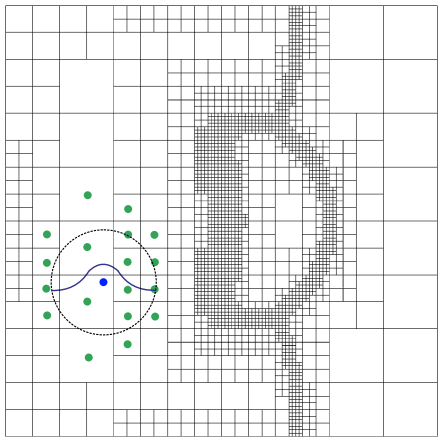


Fig. 6. Illustration for sampling procedure. The blue dot represents a simulation particle while green dots denote sampling particles. The blue curve indicates the kernel function shape on its support. The particle interaction cut-off radius is increased by a factor of 1.5 to take into account that some sampling particles fail to interact with any simulation particles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

When the particle simulation starts, the user may not know exactly the number of mesh elements from background mesh topology. The input particle number then does not satisfy the target criterion in Eq. (33). This scenario frequently appears when the mesh topology undergoes big changes in dynamic load-balancing simulations. For adaptive adjustment of the total particle number satisfying the target criterion, we develop a refinement and coarsening strategy.

When the difference between S and S_t is larger than a threshold, a corresponding refinement or coarsening procedure is invoked. The number of particles to be added or deleted can be calculated by

$$N = \alpha \frac{\left| N_p - N_e S_t \right|}{N_c}, \tag{34}$$

where N_c represents the total number of particle colors and α is a relaxation factor. Concerning refinement, new particles should be generated away from interface and boundaries such that the interface is not disturbed. Otherwise, the new inserted particles close to an interface may lead to penetration into the other phase. Moreover, a certain number of new particles is distributed in the respective phase in a random way. Too many new generated particles gathering together tend to induce numerical instability.

3.5. Sampling procedure for post-processing

In order to assign all mesh elements to a certain partitioning sub-domain, an averaged sampling algorithm is designed as post-processing method. We construct a set of sampling particles at the coordinate centers of leaf mesh elements as shown in Fig. 6. Each simulation particle interacts with all sampling particles located in its kernel support.

For sampling particle i, the sampling function $C_{i,k}$ defined by

$$C_{i,k} = \sum_j \gamma_{jk} W(r_{ij}, h), \quad \begin{cases} i = 0, \cdots, N_e - 1, \\ j = 0, \cdots, N_p - 1, \\ k = 0, \cdots, N_c - 1, \end{cases} \tag{35}$$

where $\gamma_{jk} = \begin{cases} 1, & C_j = k, \\ 0, & \text{otherwise}, \end{cases}$ is calculated. The Wendland kernel [41]

$$W(r_{ij}, h) = \alpha(1-s)^8(1 + 8s + 25s^2 + 32s^3), \tag{36}$$

with

$$\alpha = \begin{cases} \frac{78}{7\pi h^2}, & 2D, \\ \frac{1365}{64\pi h^3}, & 3D, \end{cases} \tag{37}$$

is chosen as smoothing function.

During this procedure another traverse of the multi-resolution based hierarchical data structure is involved. A specific simulation particle is mapped onto every level of the multi-resolution cell-linked list, sampling particles within the support domain of its kernel are searched and the sampling function $C_{i,k}$ is calculated. The computational cost of this procedure is negligible, as the number of sampling particles is relatively small and it is conducted only when a detailed load-balance measurement is needed during the simulation. Finally, one mesh element represented by the sampling particle i is assigned to a partitioning sub-domain of color k_0 satisfying

$$C_{i,k_0} = \max(C_{i,0}, \cdots, C_{i,N_c-1}),\ k_0 = 0, \cdots, N_c - 1. \tag{38}$$

3.6. Convergence criteria

The imbalance error of a certain mesh topology partition can converge to zero when an appropriate number of particles is used and the relaxation is performed for sufficient time. However, the global zero-error target implies a prohibitive amount of computational cost in certain cases. A maximum imbalance error Er_{max} is defined as

$$Er_k = \frac{\left| N_{e,k} - \frac{N_e}{N_c} \right|}{\frac{N_e}{N_c}},\ k = 0, \cdots, N_c - 1, \tag{39}$$

$$Er_{max} = \max(Er_0, \cdots, Er_{N_c-1}), \tag{40}$$

where $N_{e,k}$ stands for the number of mesh elements belonging to partitioning sub-domain k. A predefined threshold of Er_{max}, e.g. 4%, is adopted as load-balance convergence criterion in our current implementation. Such a small level of load-imbalance is also accepted through user-tunable parameter in other partitioners [20]. Some particles may never come fully to rest as the solution of our model equations may not be unique. A second criterion is developed as follows: within a certain number of iterations, e.g. 50, the partitioning sub-domains do not change. The simulation terminates under the condition that both of these criteria are satisfied.

For dynamic load-balancing simulation, the above algorithms are repeated consecutively after the mesh topology changes. The major difference is that the SPH simulation result of the previous background mesh topology acts as the initial condition for a new mesh partition.

In practice, if we want to partition one mesh topology with few elements into too many parts, it may be quite difficult to decrease Er_{max} to less than the predefined threshold. Total iterations are generally limited to a certain number, e.g. $8000 \sim 10000$ for static partitioning and 40 for dynamic partitioning, if not mentioned otherwise.

4. Validation

In this section, we consider validation examples in two spatial dimensions. The restriction to two dimensions is to facilitate the presentation. We emphasize that the algorithm also applies to three dimensions with analogous results. Due to space limitations, we refer the presentation of three-dimensional results to the forthcoming 2nd part of this paper.

An adaptive multi-resolution code (MR-SIM) [2] is employed to perform the fluid simulations and to provide the background mesh topology for partitioning. We compute two types of partitionings, a static partitioning for the initial mesh topology generated by multi-resolution analysis before the actual flow simulation starts, and a dynamic partitioning for adaptive mesh topology following flow evolution. We point out that at this point, the dynamic partitioning algorithms have not yet been parallelized on distributed-memory architectures, i.e. all algorithms are implemented in a serial version. In order to measure and compare the computational efficiency, all simulations are carried out on the same desktop workstation equipped with 12 Intel(R) Xeon(R) CPU E5-2620 V2 cores (2.1 GHz and 32 GB memory) and Scientific Linux 6.7 system.

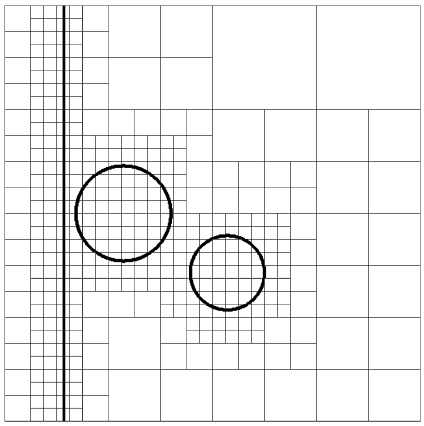


Fig. 7. Shock double water-columns interaction: target block-structured initial background mesh topology and schlieren-type images of density gradient $|\nabla\rho|$.

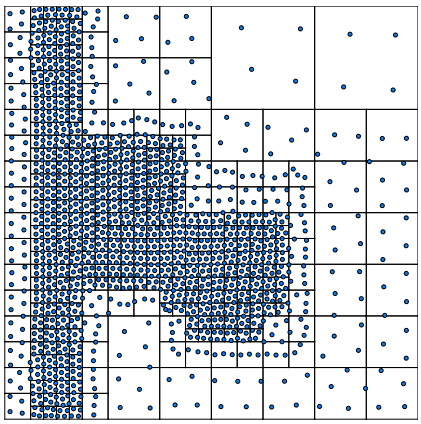


Fig. 8. Shock double water-columns interaction: particle representation of the target mesh topology. The blue circles represent simulated particles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1. Shock double water-columns interaction

The computational domain for this case is $[0, 2.8]\times[0, 2.8]$. The initial physical condition contains a shock wave and two water-columns close to each other. After multi-resolution analysis, the initial mesh topology is given in Fig. 7. The mesh resolution differs at most 8 times.

In order to check whether the target background mesh topology can be well represented, the surface tension model is turned off and all the simulated particles feature the same color. As shown in Fig. 8, the mesh resolution distribution is well characterized by the particle distribution. Each mesh element contains approximately 4 uniformly-distributed particles, which fully meets the design requirements.

We investigate the effect of the initial condition. In Fig. 9, the left panel shows uniformly distributed particles with colored function as straightforward choice of initial condition; on the right, particles are randomly assembled in circles arranged by a close packing method as newly proposed initial condition. The radius R is scaled slightly such that a gap between different particle phases is preset. This initial gap guarantees that there is enough time left to establish the sharp interface of multi-phase flow. It can be observed that particles near interface corners are always surrounded by four distinct phases for the former initial condition. Fig. 10 shows two time snapshots of SPH simulation results, the interface shape of the central phase gradually develops into quadrilateral with uniform initial condition. It becomes a hexagon or pentagon with our newly proposed initial condition. The latter ensures that all interface corners are shared by three phases and the interface angle is approximately 120°, which is a stable interface configuration.

Fig. 11 shows the particle distribution after 8000 iterations and the corresponding domain decomposition. The interface shape of particle phase with dark-orange color originating from a uniform initial condition forms a longer and thinner filaments than that from new initial condition. Particles in the filament may break away from their host phase and penetrate into other phases, as the inherent function of surface tension is unphysical. Furthermore, the sub-domain with purple color

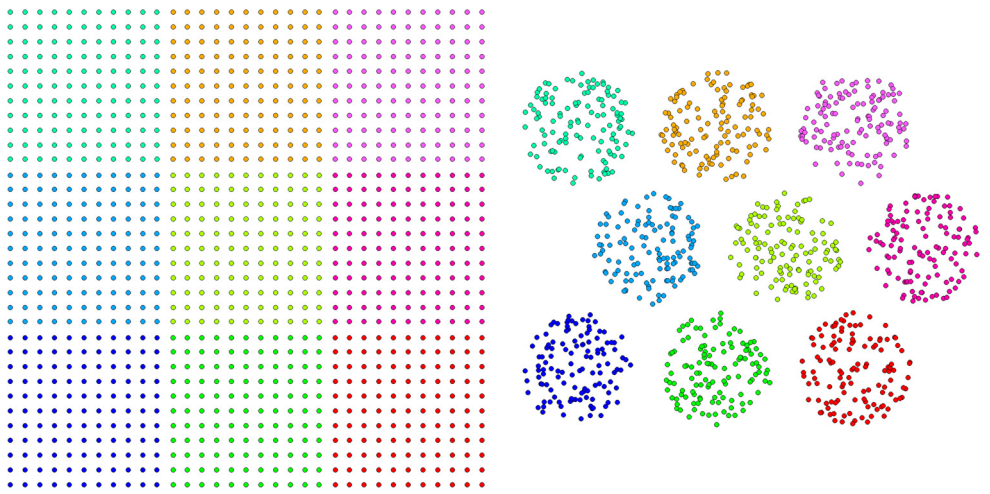


Fig. 9. Shock double water-columns interaction: initial particle distribution. Uniform distribution (left) and random distribution in combination with close packing method (right). The domain is partitioned into 9 sub-domains.

Table 1
Maximum imbalance error measurement and wall-clock time.

Number of total mesh elements	Number of partitioning sub-domains	Iteration number	Er_{max}	Wall-clock time (s)
385	4	3200	3%	38.4
385	6	3200	3%	38.7
385	9	8000	5% $\sim$ 10%	82.5

Shock double water-columns interaction. The wall-clock time is measured without multi-threaded parallelization. For block-structured mesh, 385 mesh elements mean typical 98560 cells with each block of resolution 16×16 [2].

generated by the uniform initial condition contains one mesh element with only a point connected to its host phase, while all mesh elements of the partition produced by the new initial condition connect to the other elements of the same phase or color with an edge. The new initial condition is beneficial for numerical robustness, interface communication reduction and the production of well connected sub-domains.

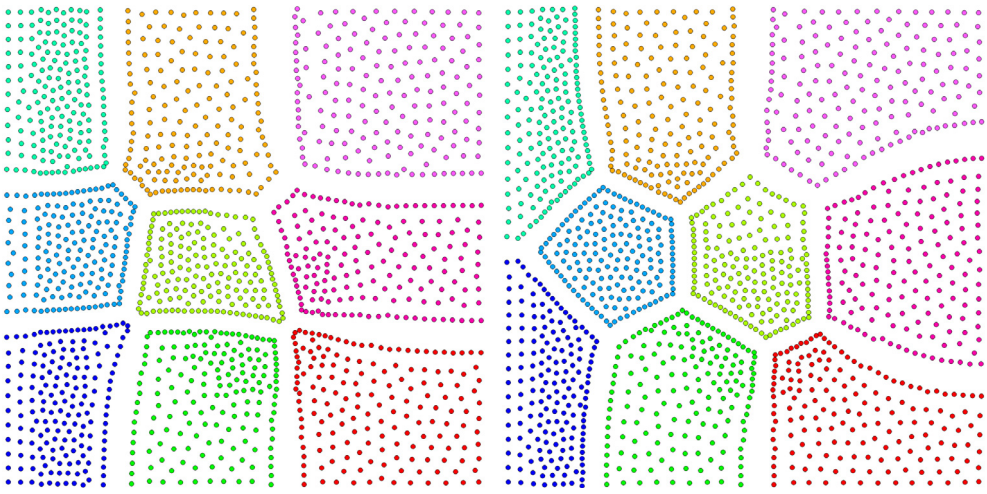
In Fig. 12, particles are mapped onto the background mesh topology. The left panel shows that the interfacial gap is less than the local mesh element size, and each mesh element contains approximate 4 particles except for those within an interface neighborhood. The right panel shows the coordinate relationship between the final partitioning sub-domains and the colored particle distributions. The proposed sampling procedure accurately classifies mesh elements into respective partitioning sub-domains.

Fig. 13 provides the distribution of the particle-neighbor number and the distribution of target density. Except for boundary particles, particles generally possess 10 neighbors, which is insufficient for standard SPH. The target density differs at most 64 times throughout the computational domain and jumps according to the mesh resolution change. With our proposed physically motivated models and specially designed EOS, the simulation is numerically stable and relaxes to the target result.

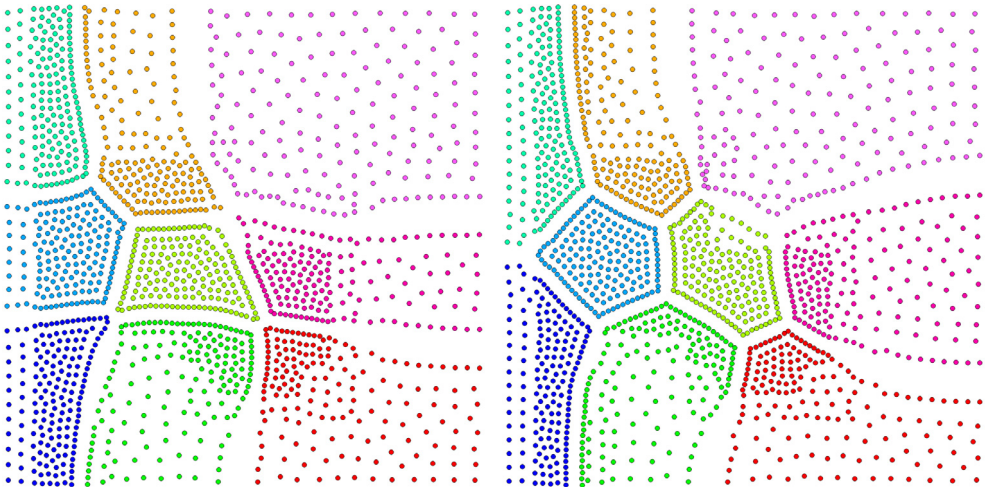
As shown in Table 1, for partition number less than 9, the simulations reach the imbalance error tolerance and converge within 3200 iterations. The error exhibits oscillations between 5% $\sim$ 10% and the simulation fails to achieve convergence until the iteration limit upon increasing the partition number to 9. Considering that the number of total mesh elements is too small at about 385 (representing 98560 mesh cells), when the computational domain is expected to be divided into 9 parts, the interfacial region dominates over partitioning sub-domains. Although the partitioning load-balance is sacrificed somewhat, a parallel simulation will still gain benefits as the interface communication is well optimized. For static partitioning, the proposed method is more time-consuming than the state-of-the-art methods. However, the present particle-based method is straightforward to be parallelized in a multi-threaded environment so that the wall-clock time can be reduced significantly. On the other hand, the dynamic partitioning can be quite efficient since only 40 iterations are necessary after the mesh topology changes. A preconditioning method may further improve the static partitioning efficiency. Another notable result is that the total computing-time only depends on the total mesh element number and the iteration number regardless of the partitioning number.

Table 2 gives the run-time analysis in detail. It is noted that the particle evolution algorithm accounts for the largest proportion of total wall-clock time.

Fig. 14 shows the dynamic partitioning results including particle distribution and partitioning sub-domains. It can be observed that the background mesh topology adjusts significantly during the simulation. Since our refinement and coarsening algorithms always adjust the total particle number correspondingly, the mesh topology is well represented with the



(a) Solution after 1000 iterations



(b) Solution after 2000 iterations

Fig. 10. Shock double water-columns interaction: evolution of SPH simulation results. Uniform initial distribution (left) and random initial distribution in combination with close packing method (right).

Table 2
Wall-clock time measurement for component algorithms.

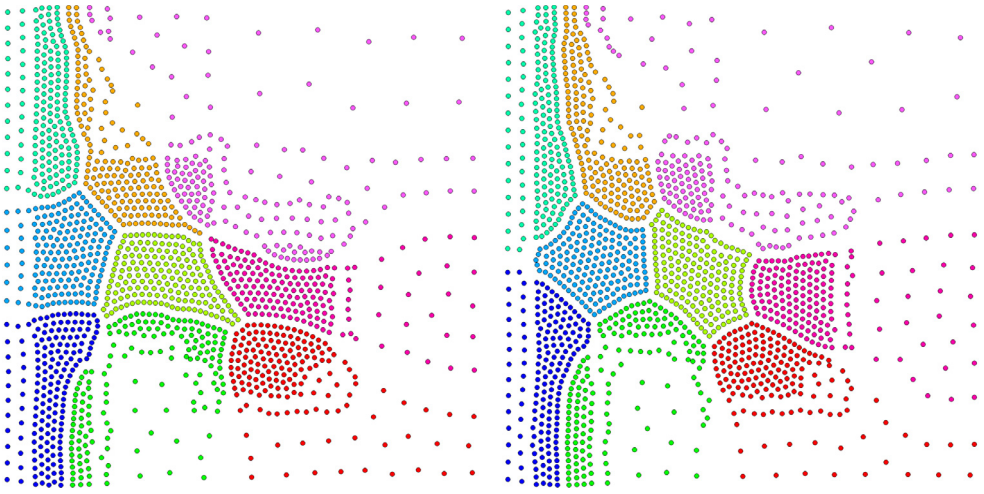
Total time (s)	Refinement and coarsening (s)	Boundary condition (s)	Fast neighbor search (s)	Particle evolution (s)
82.5	0.02 (0.02%)	6.3 (7.6%)	2.9 (3.5%)	73.28 (88.8%)

Shock double water-columns interaction with 385 mesh elements. The partitioning number is 9.

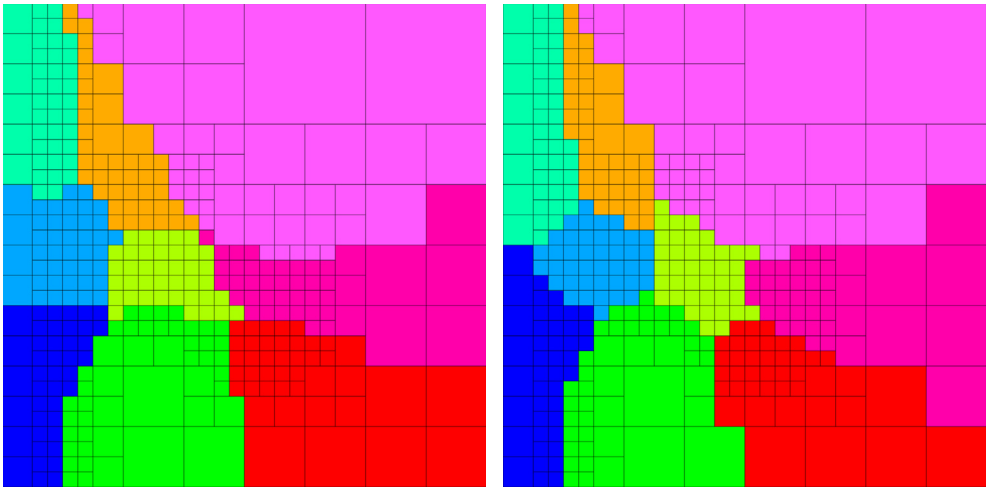
same accuracy throughout the simulation. The boundaries of different particle phases preserve convex and smooth shapes satisfying the design target. Although the mesh topology varies strongly, the partitioning topology however merely suffers from minor adaption, which implies that the produced partition is physically localized and implicitly incremental. For high performance computing, the data migration between processors is considerably reduced.

Fig. 15 gives the performance data of the load-balancing and the communication volume. During the simulation, the maximum imbalance error is approximately maintained to be 5% $\sim$ 10%. For the static partitioning, the communication volume is optimized by the surface tension model and converges roughly within 4000 iterations. For dynamic partitioning, the communication volume is optimized to decrease almost monotonically.

In order to demonstrate the potential of proposed method, static partitionings of 36 and 64 subdomains are computed for this case. The background mesh is generated at simulation time $t = 0.3$ of flow evolution. As shown in Fig. 16, the mesh element number is 10485, corresponding to a mesh cell size 2684160 in 2D. Fig. 17 gives the converged particle distribution.



(a) Solution after 8000 iterations



(b) Partitioning sub-domains after post-processing

Fig. 11. Shock double water-columns interaction: convergent solution of SPH simulation and domain decomposition. With initial uniform distribution (left) and random initial distribution in combination with close packing method (right). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

Table 3
Memory consumption.

Method	8^3 cells per mesh element			16^3 cells per mesh element		
	Number of elements	Memory consumption (GB)	Relative	Number of elements	Memory consumption (GB)	Relative
MR-SIM	7120	3.25	100%	5972	16.90	100%
partition	7120	0.56	17.23%	5972	0.55	3.25%

3D shock water-droplet interaction: the partitioning number is 8. The statistical data of MR-SIM represents the memory consumption of all fluid variables and the AMR data structure while that of partition denotes all the memory consumptions by our partitioning method.

The partitioning subdomains mostly are convex with sharp interfaces. As shown in Fig. 18, both the load-imbalance error and the communication volume are well optimized.

In order to test the memory consumption, we run a rather large realistic simulation of similar case. As shown in Table 3, when each mesh element represents 8^3 cells, the total memory usage increases by 17.23% with our partitioning method. If 16^3 cells are distributed in each mesh element following a typical setting [2], the memory consumption of our partitioning method only accounts for 3.25% of that taken by the fluid simulation. The increased memory consumption is small even without code optimization.

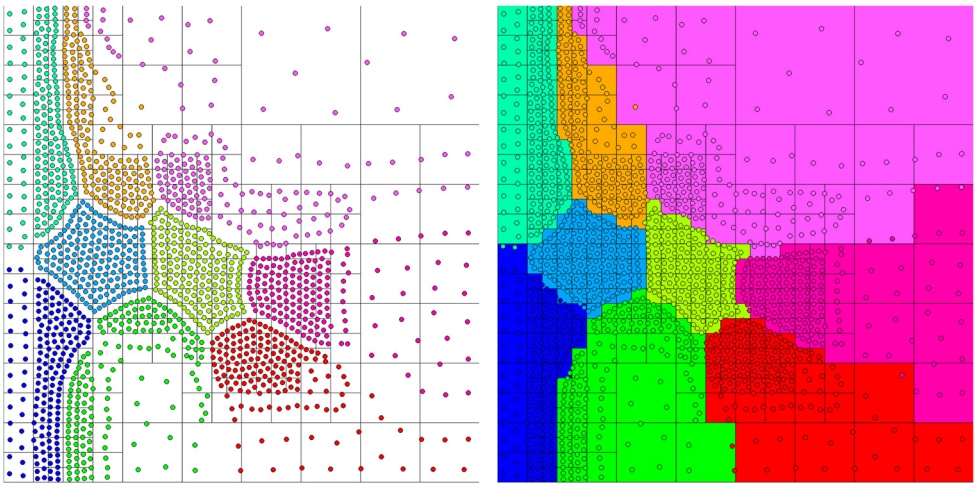


Fig. 12. Shock double water-columns interaction: post-processing procedure. Colored particles with background mesh (left) and partitioning sub-domains with particles after post processing (right).

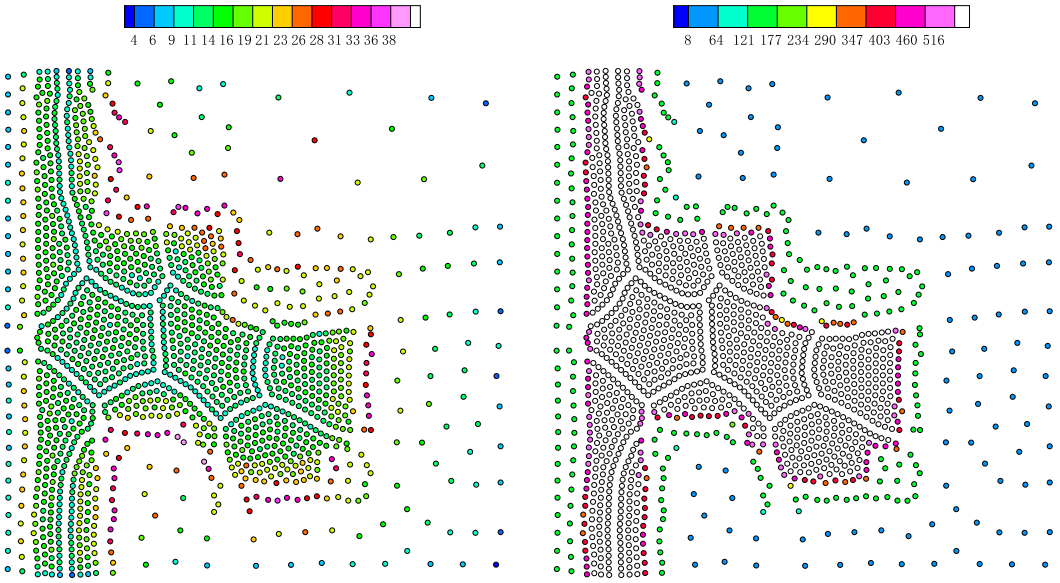


Fig. 13. Shock double water-columns interaction: distribution of the particle-neighbor number (left) and the target density (right) for a convergent solution. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)

4.2. Uniform mesh partitioning

We consider a static partition for uniform background mesh topology, which consists of 30×30 mesh elements (representing 230400 mesh cells). The SPH simulation involves 3600 particles and converges to target convergence criteria within 3200 iterations. Fig. 19 shows the SPH simulation result and the corresponding domain decomposition. Since all partitioning sub-domains preserve convex shapes as expected, the total interfacial area is well optimized. A considerably larger static partition, which involves 230400 particles and 57600 uniform mesh elements (representing 14745600 mesh cells), is shown in Fig. 20. The maximum imbalance error Er_{max} is approximate 4.17% with 50000 iterations. It can be observed that our partitioning result is similar to that from representative diffusion BUBBLE-FOS/C graph (Re)partitioning heuristic (see their Fig. 1 and Fig. 7 [20]) and is better than the solutions delivered by well-established softwares, e.g. Metis [7]. H. Meyerhenke et al. [20] point out that such convex sub-domain shapes, e.g. hexagon, are the regular shapes which cover a two-dimensional domain with smallest interface areas and thus the total interfacial energy is reduced to minimum.

As shown in Fig. 21, the load imbalance error and the communication volume are optimized to decrease monotonically.

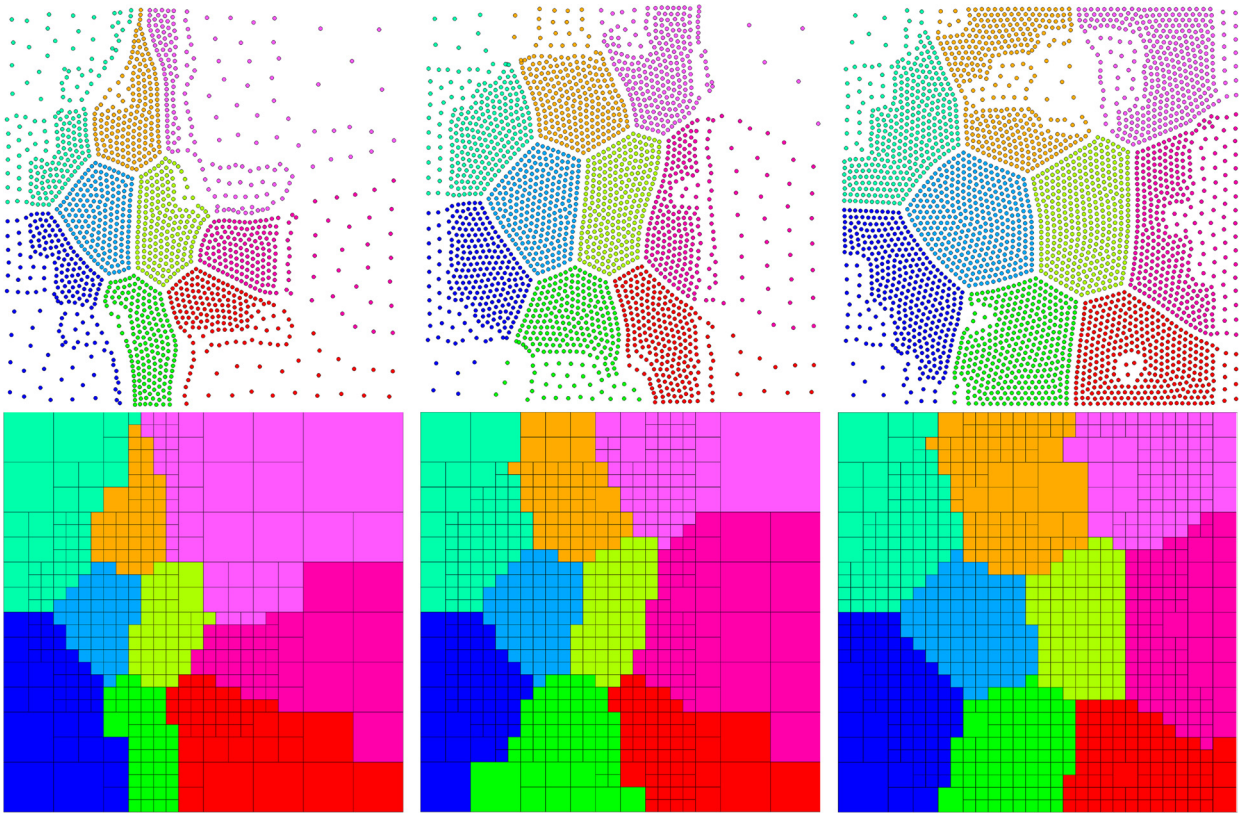


Fig. 14. Shock double water-columns interaction: dynamic load-balancing. SPH simulation solution (top) and domain decomposition with background mesh (bottom). From left to right: snapshots of simulation result at time 0.1, 0.2, 0.3.

4.3. Underwater explosion

This case is carried out in the computational domain $[0, 4]\times[0, 4]$. The ratio of the highest to the lowest resolution is 16 and most of mesh elements gather together in a narrow region as shown in Fig. 22. This inhomogeneity imposes a great deal of suffering on partitioning methods and is considerably challenging.

Fig. 23 gives the comparisons of SPH simulation result and domain decomposition with partition number of 4 and 9. The particle simulation converges within 3400 iterations when partitioning the domain into 4 parts while experiences 8000 iterations for partition number 9. The reason is similar to that discussed in case 1. All the particle phases maintain sharp and convex shapes well. The consumed wall-clock time is 61.0 s and 112.0 s, respectively without multi-threaded parallelization. Note that the 448 mesh elements typically represent 114688 mesh cells [2].

Fig. 24 shows the results of dynamic load-balancing simulation with partition number 4. It is obvious that the mesh topology experiences extremely large changes during the simulation. The mesh element distribution is continuously reconstructed via the coarsening and refinement procedures of multi-resolution analysis [2]. The total number of mesh elements involved grows rapidly from 448 (representing 114688 mesh cells) to 1822 (representing 466432 mesh cells). Nevertheless, the topology of partitioning sub-domains is highly analogous and the interface of the multi-phase flow only adjusts the position and shape slightly. In addition, all mesh elements connect to host partitioning sub-domain with edge, leading to strictly continuous partition. The surface tension builds up the sharp interface between distinct particle phases and keeps individual partitioning sub-domain closed. No particle penetration occurs throughout the entire simulation. When the partition number increases to 9, we once again obtain domain decompositions, which are physically localized and strictly connected with optimized interface area as sketched in Fig. 25.

The history of maximum imbalance error Er_{max} and scale ratio S involving both static and dynamic load-balancing simulations is shown in Fig. 26. For the partition number 4, SPH simulation always obtains a convergent result since the iteration number 3000. According to our numerical experiments, the error can be further reduced to nearly zero if more iterations are conducted when mesh topology changes. In terms of the partition number 9, Er_{max} fluctuates between 5% $\sim$ 10%, hence resulting in more total iterations accordingly. For both scenarios, the target scale ratio $S_t = 4$ is well maintained after 1000 iterations by the proposed refinement and coarsening algorithms.

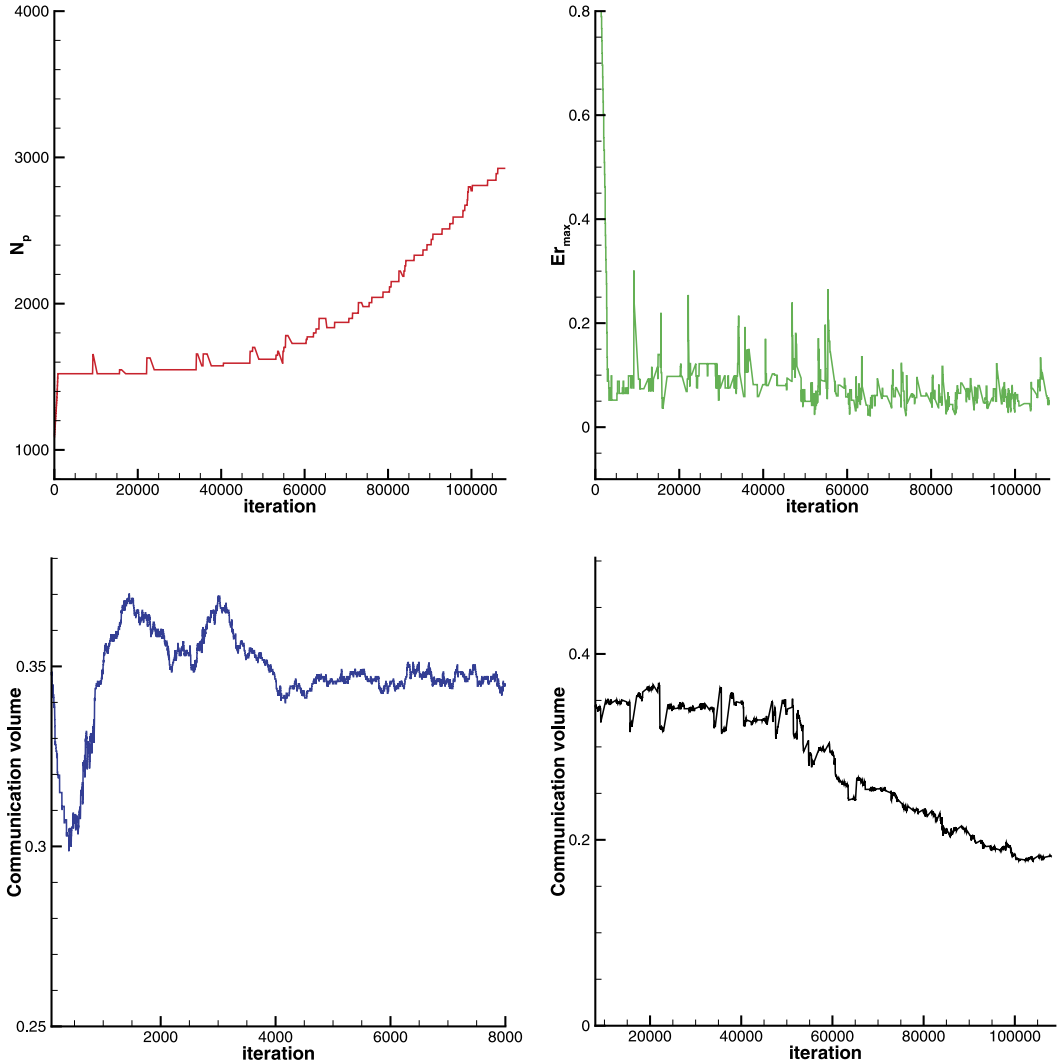


Fig. 15. Shock double water-columns interaction: history of total particle number N_p (top left panel), maximum imbalance error Er_{max} (top right panel), communication volume for static partitioning (bottom left panel) and dynamic partitioning (bottom right panel) versus iteration number. The partition number is 9. The communication volume is defined as the boundary particle number normalized by the total particle number while the boundary particle number is defined as the number of particles which feature neighboring particle of different color within its smoothing region.

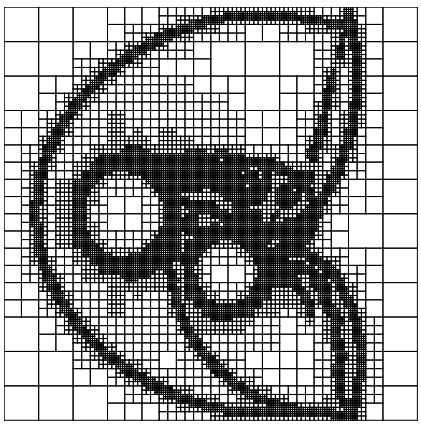


Fig. 16. Shock double water-columns interaction: target block-structured background mesh topology. The mesh element number is 10485, corresponding to a mesh cell size 2684160 in 2D.

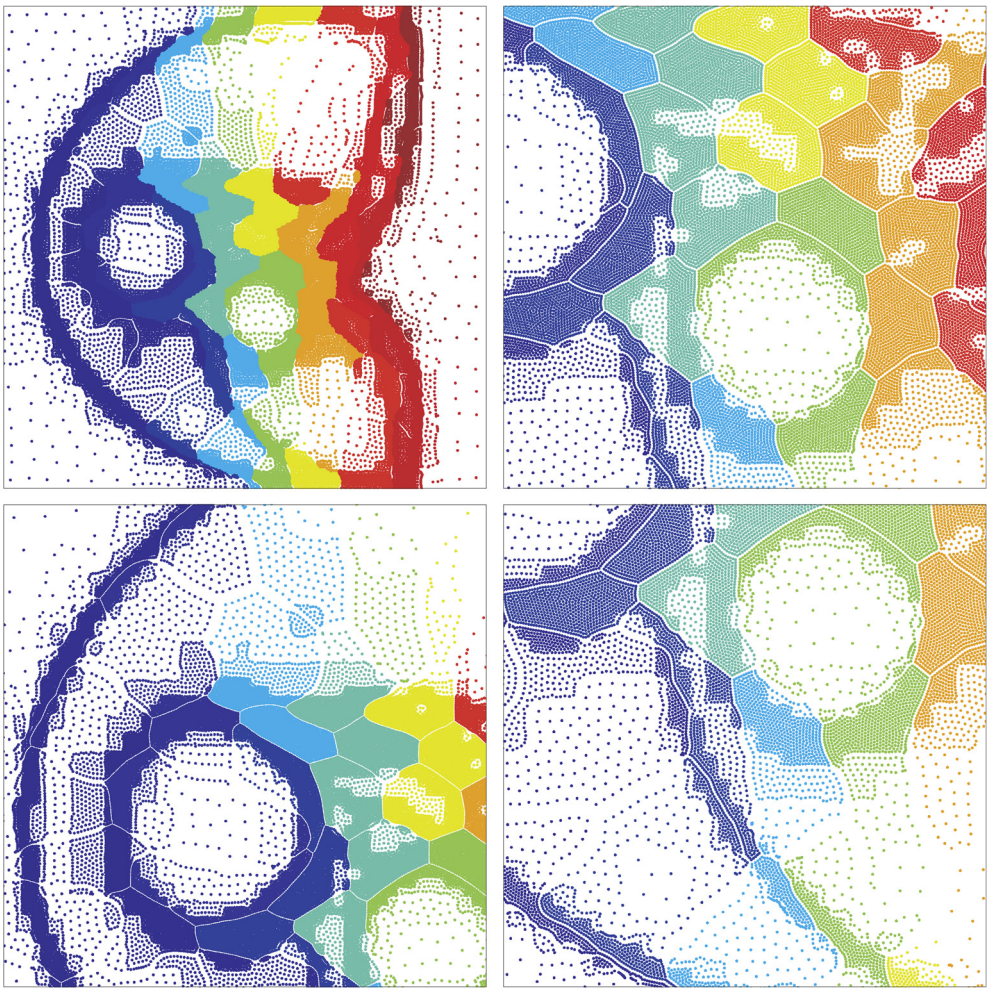


Fig. 17. Shock double water-columns interaction: the partitioning number is 64. The simulated particle number is 41728. The overall view (top left panel) and zoom-in views (others).

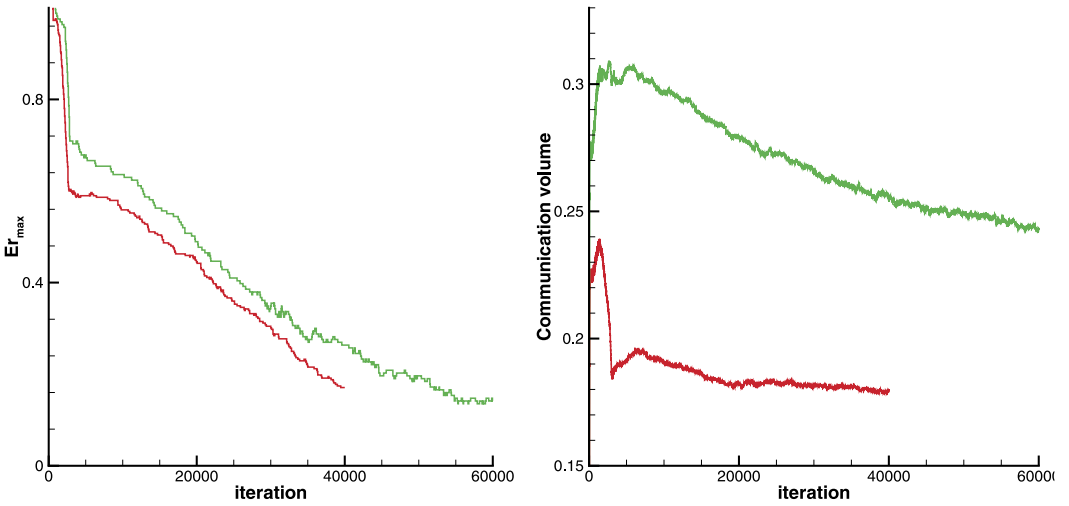


Fig. 18. Shock double water-columns interaction: history of load-imbalance error and communication volume versus iteration number (static partitioning). The red line represents the partition number 36 while the green line represents the partition number 64. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

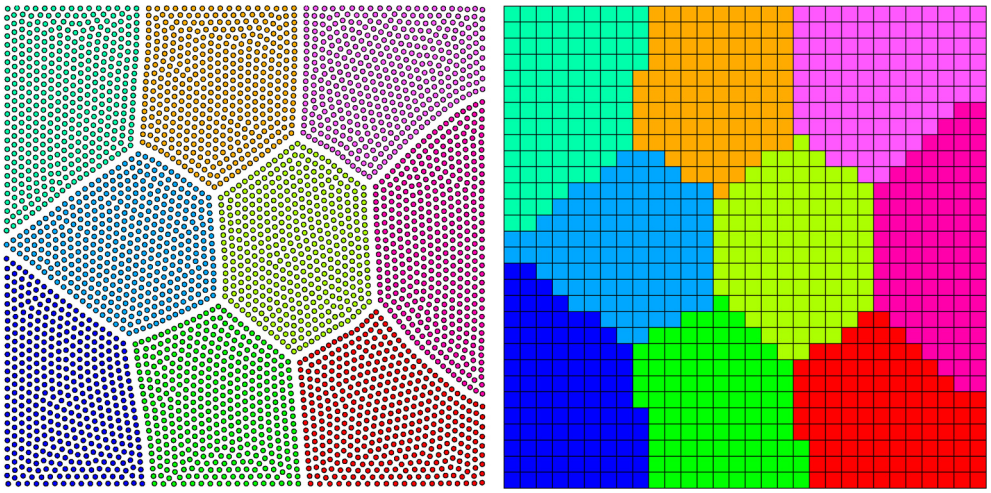


Fig. 19. Static partitioning for uniform mesh. SPH simulation solution (left panel) and domain decomposition with background mesh (right panel). The partition number is 9. The mesh element number is 900, corresponding to 230400 mesh cells.

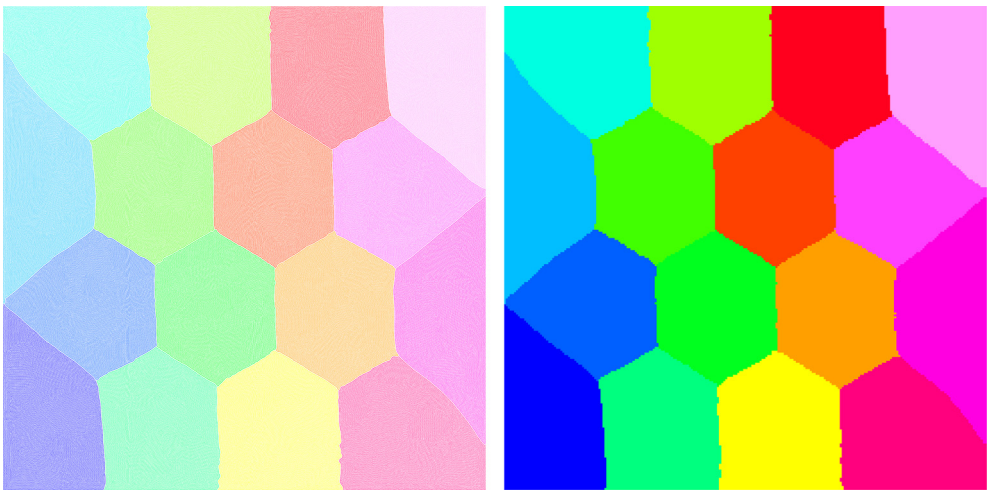


Fig. 20. Static partitioning for uniform mesh. SPH simulation solution (left panel) and domain decomposition (right panel). The partition number is 16. The mesh element number is 57600, corresponding to 14745600 mesh cells.

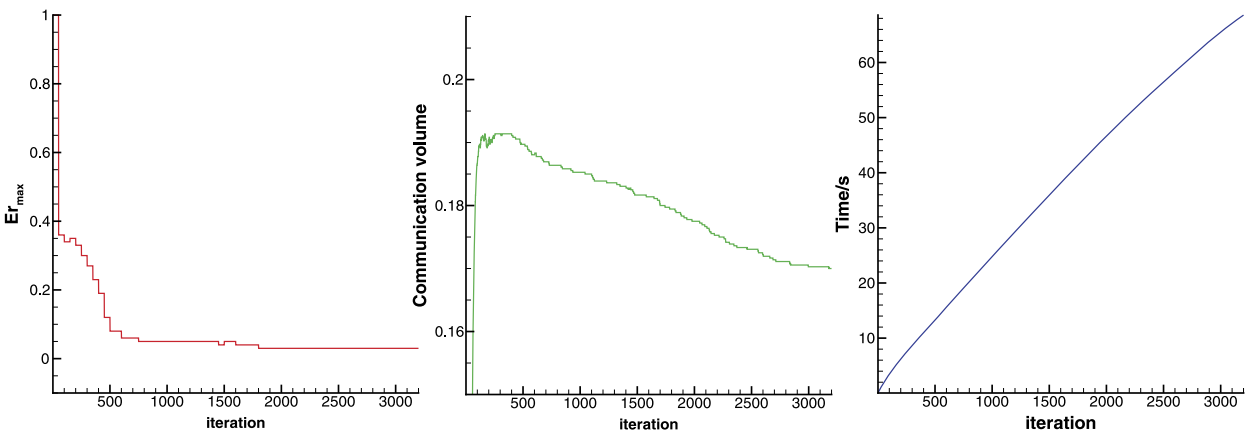


Fig. 21. Static partitioning for uniform mesh. History of maximum imbalance error Er_{max} (left), communication volume (middle) and wall-clock time (right) versus iteration number. The partition number is 9.

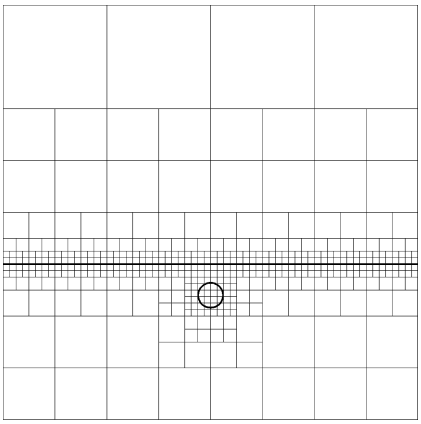
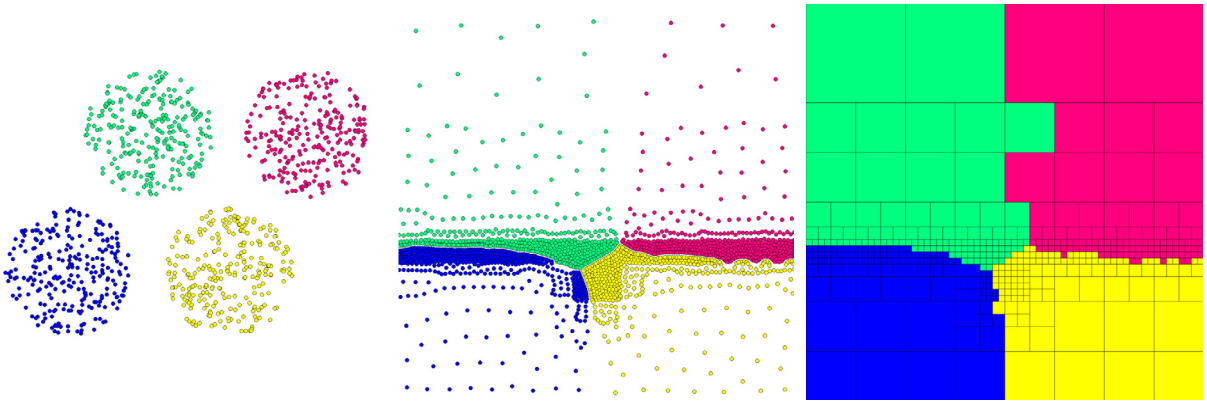
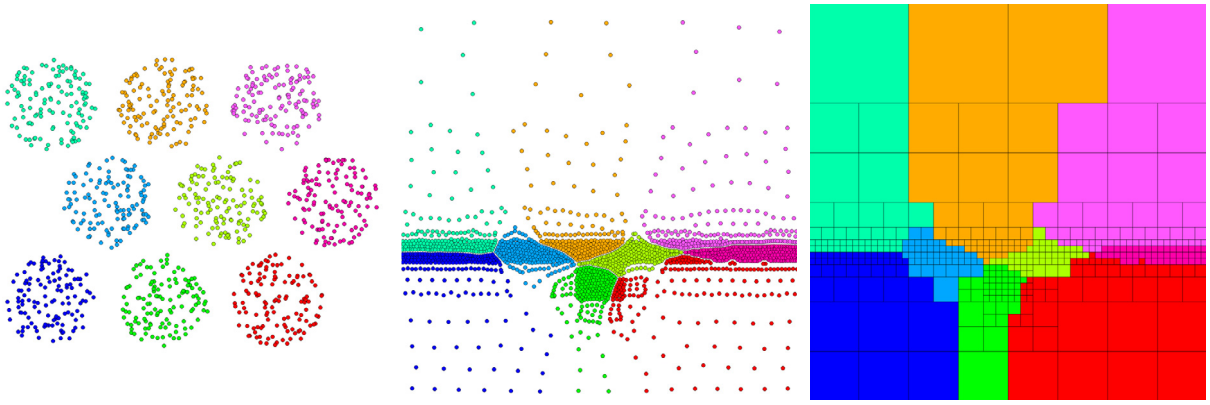


Fig. 22. Underwater explosion: target initial background mesh topology and schlieren-type images of density gradient $|\nabla\rho|$.



(a) From left to right: initial condition, solution at iteration number 3400, domain decomposition. The consumed wall-clock time is $61.0s$.



(b) From left to right: initial condition, solution at iteration number 8000, domain decomposition. The consumed wall-clock time is $112.0s$.

Fig. 23. Underwater explosion: solution of SPH simulation and corresponding domain decomposition. The domain partition number is 4 (top) and 9 (bottom) respectively. The total mesh element number is 448 representing 114688 mesh cells.

As shown in Fig. 27, for the static partitioning, the communication volume converges well. For the dynamic partitioning, the communication volume is well optimized and decreases monotonically. The large partitioning number typically induces more communication and larger iteration numbers.

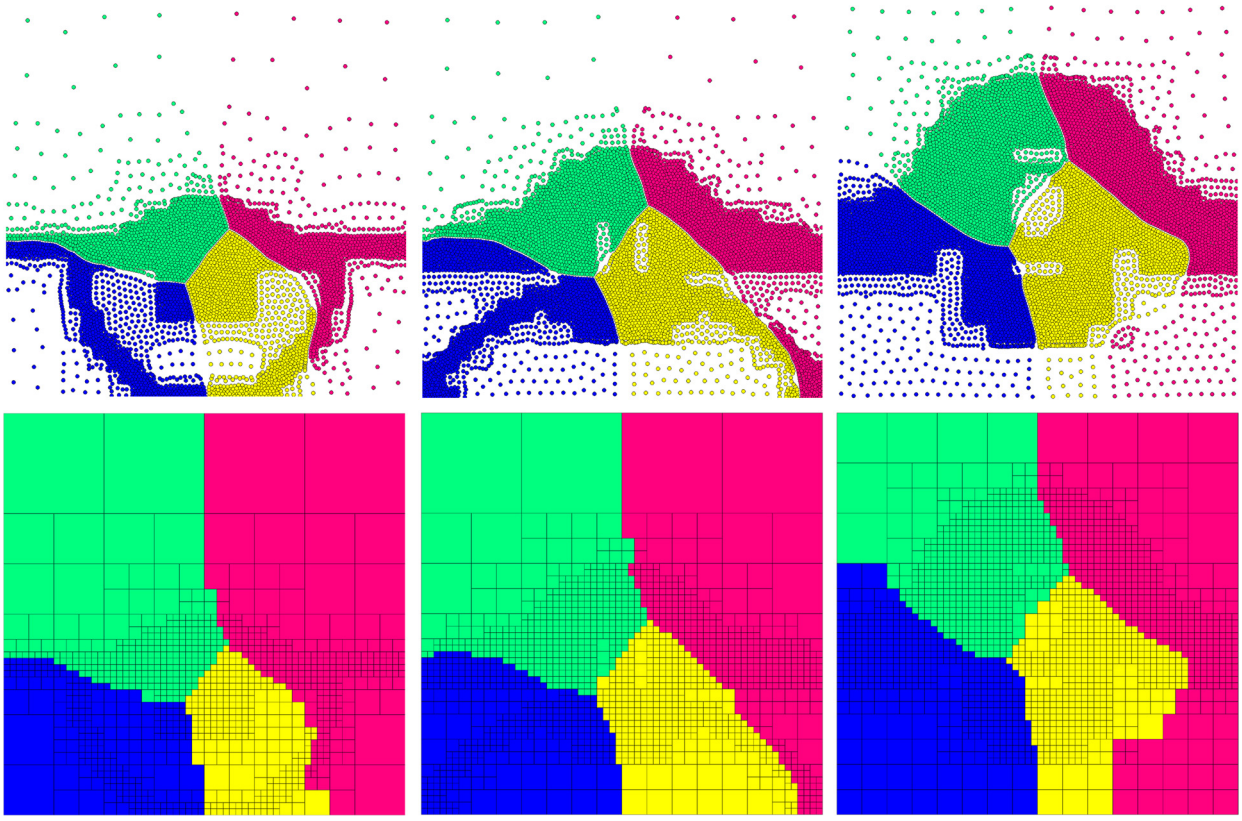


Fig. 24. Underwater explosion: solution of SPH simulation and corresponding domain decomposition. From left to right: SPH solution at fluid simulation time 0.06, 0.12, 0.20. Top: particle distribution, bottom: domain decomposition. The domain partition number is 4.

Table 4
Maximum imbalance error measurement and wall-clock time.

Number of total mesh elements	Number of partitioning subdomains	Iteration number	Er_{max}	Wall-clock time (s)
548	4	6850	4%	348.0
548	6	7200	3%	364.8
548	9	8000	10∼13%	381.4

Shock air-R22 bubble interaction. The wall-clock time is measured without multi-threaded parallelization. For block-structured mesh, 548 mesh elements mean typical 140288 cells with each block of resolution 16×16 [2].

4.4. Shock air-R22 bubble interaction

For this case, the mesh topology is canonical for the shock-bubble interaction simulation as Fig. 28. The computational domain is [0, 0.445]×[0, 0.089] with high aspect ratio. The mesh resolution differs by 16 times, leading to ratios of target density up to 256.

Fig. 29 and Fig. 30 provide the simulation results for static partition. Initially, particles are distributed in a small region and then expand to fill the full computational domain. After the simulations have converged, the characteristics of background mesh topology are well represented. As shown in Table 4, the SPH simulations converge within less than 8000 iterations for the partition number 4 and 6. In terms of larger partition number 9, it takes more iterations and produces larger imbalance error due to the relatively too small number of total mesh elements $N_e = 548$ (representing 140288 mesh cells).

Fig. 31 shows the dynamic partition results for the partition number 9. Although the mesh topology changes strongly, the SPH simulation robustly captures these target mutations and provides satisfactory domain decompositions, which are strictly continuous and localized. The relative-position relationship between distinct partitioning subdomains is preserved while minimizing the data movements among processors for parallel simulations.

Fig. 32 gives the convergence history of the communication volume. The communication volume converges well for the static partitioning and decreases monotonically during the dynamic partitioning.

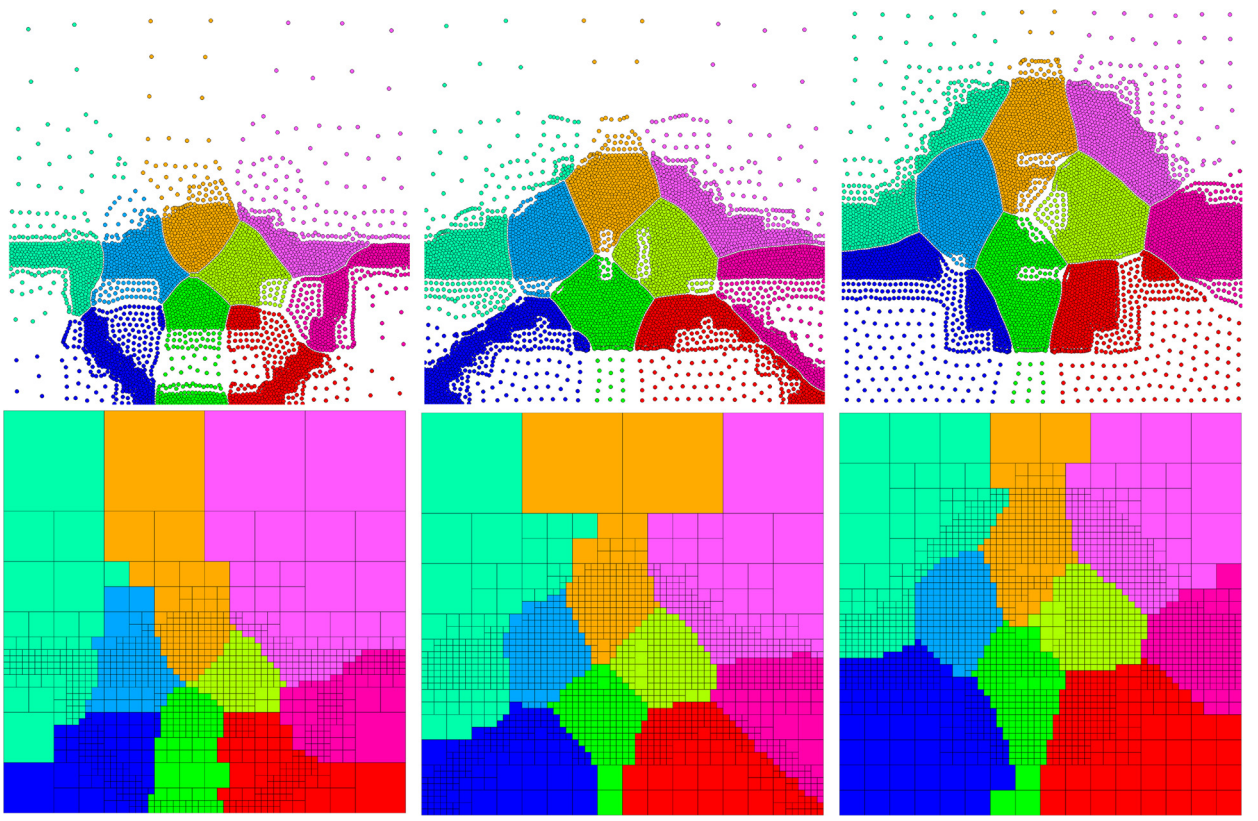


Fig. 25. Underwater explosion: solution of SPH simulation and corresponding domain decomposition. From left to right: SPH solution at fluid simulation time 0.06, 0.12, 0.20. Top: particle distribution, bottom: domain decomposition. The domain partition number is 9.

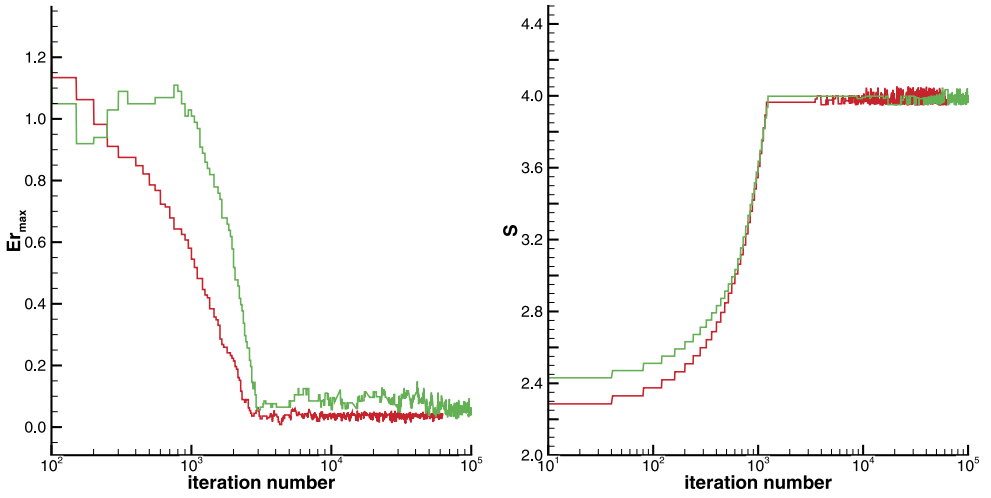


Fig. 26. Underwater explosion: history of maximum imbalance error Er_{max} (left panel) and scale ratio S (right panel) versus iteration number. The red line represents partition number 4 while the green line represents partition number 9. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Conclusion

In this paper, we propose a novel approach to partition complex mesh topologies following a physically motivated approach. A set of model equations is developed and solved by a multi-phase SPH method based on physically meaningful variables extracted from the background mesh. The main properties of our proposed method are summarized as follows:

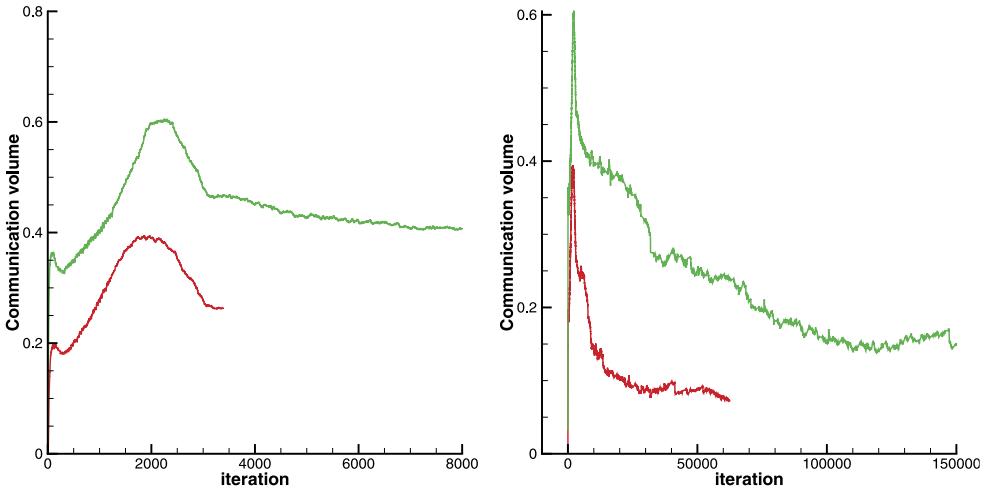


Fig. 27. Underwater explosion: history of communication volume versus iteration number. Left: for static partitioning; right: for dynamic partitioning. The red line represents partition number 4 while the green line represents partition number 9. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

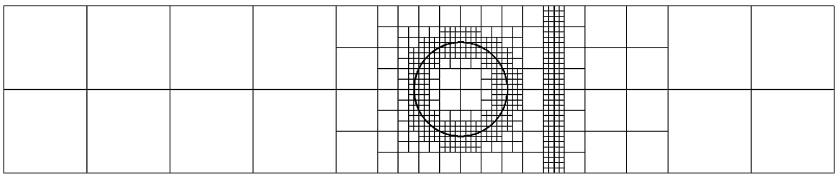


Fig. 28. Shock air-R22 bubble interaction: target background mesh topology and schlieren-type images of density gradient $|\nabla\rho|$.

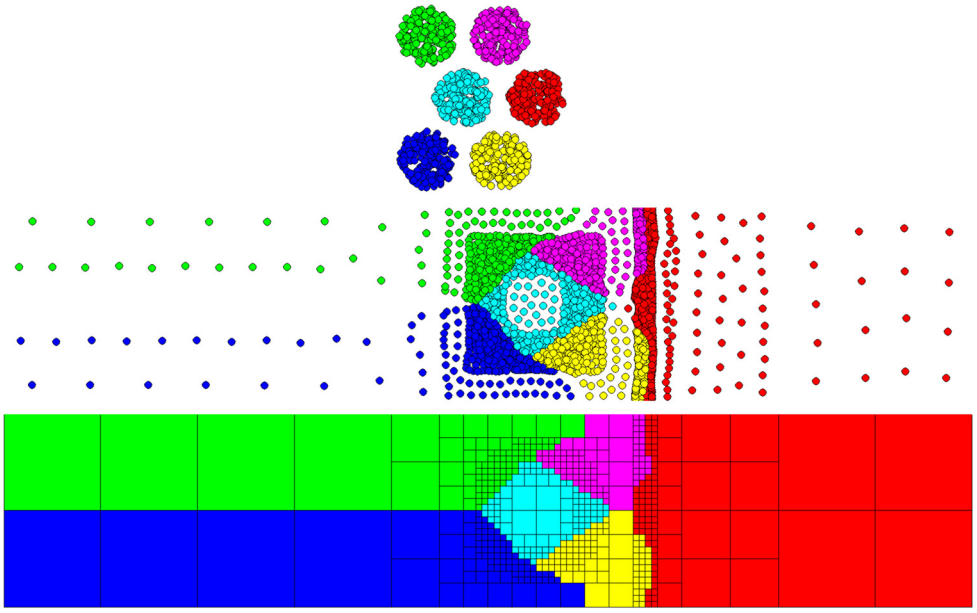


Fig. 29. Shock air-R22 bubble interaction: initial condition, solution at iteration 7200, convergent partition result. The domain partition number is 6. The total mesh element number is 548, corresponding to 140288 mesh cells.

(1) The imbalance error can be controlled with convergence criteria. In practice, it is sufficient to perform a certain number of iterations, as a compromise of computational efficiency and accuracy, instead of driving the imbalance error to zero.
(2) Different from graph-based partitioning approaches (except for the diffusion-based partitioning), which optimize the communications between distinct partitioning sub-domains by minimizing the number of edge-cuts explicitly, the proposed method optimizes mass near boundaries, i.e. element number, by a surface tension model implicitly.

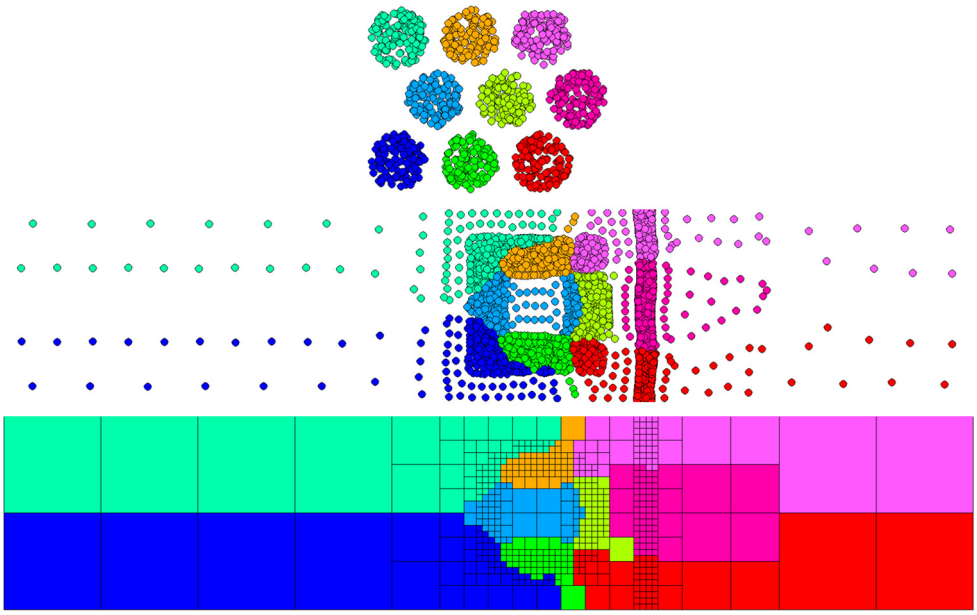


Fig. 30. Shock air-R22 bubble interaction: initial condition, solution at iteration 8000, convergent partition result. The domain partition number is 9. The total mesh element number is 548, corresponding to 140288 mesh cells.

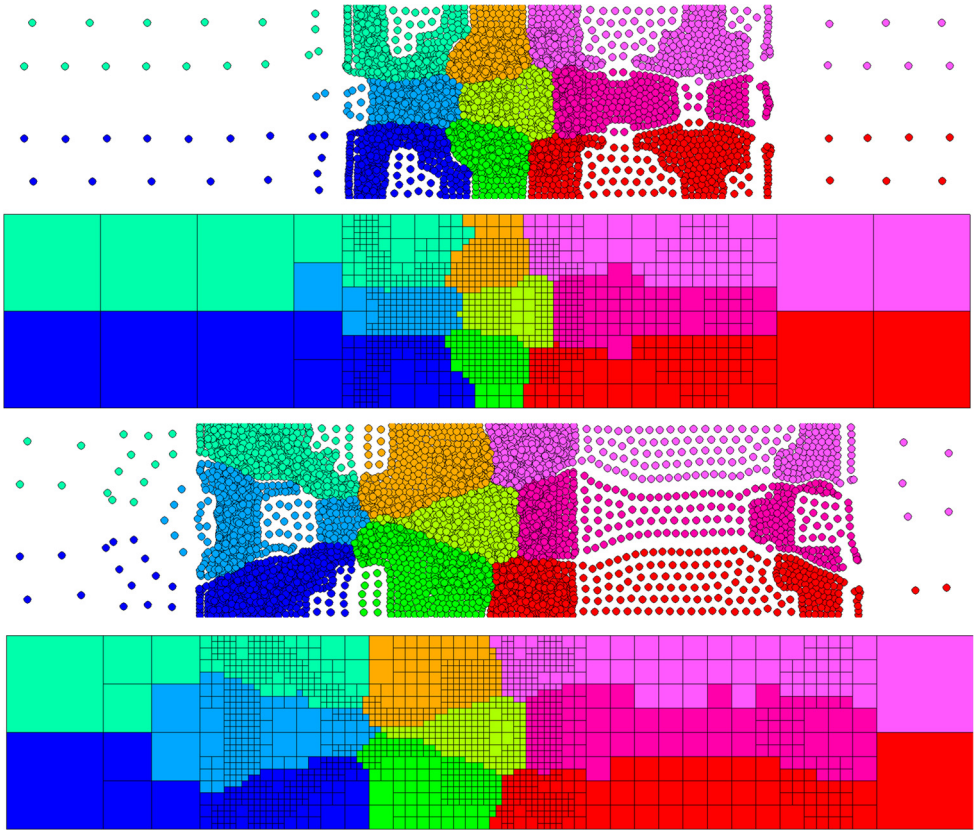


Fig. 31. Shock air-R22 bubble interaction: solution of SPH simulation and domain decomposition at fluid simulation time 0.0892 and 0.1386. Top: particle distribution; bottom: domain decomposition. The domain partition number is 9. The final mesh topology consists of 1244 mesh elements, corresponding to 318464 mesh cells.

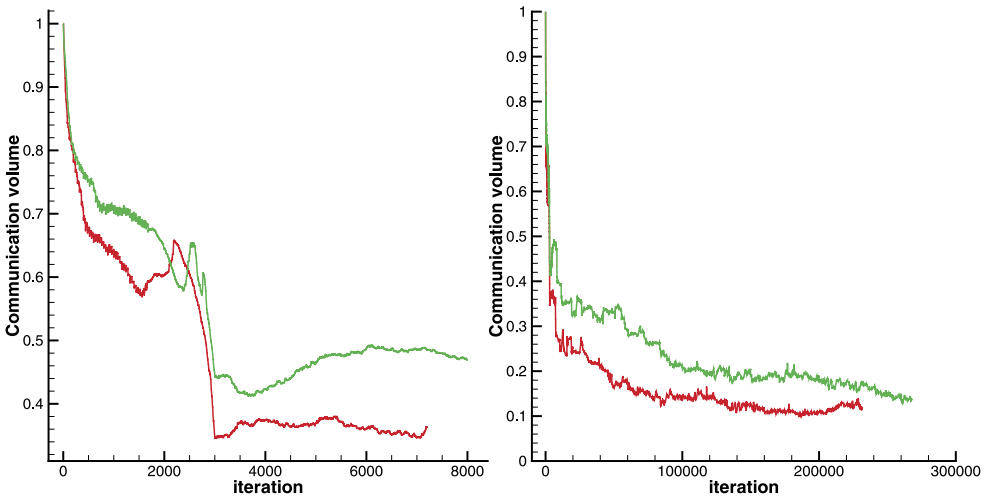


Fig. 32. Shock air-R22 bubble interaction: history of communication volume versus iteration number. Left: for static partitioning; right: for dynamic partitioning. The red line represents partition number 6 while the green line represents partition number 9. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(3) Since the solution of the model equations continuously depends on the initial and boundary condition, the partition topology experiences minor changes when the mesh topology evolves. Thus, our proposed method is implicitly incremental and physically localized.
(4) The sufficient surface tension model guarantees the coherence of each particle phase, such that connected sub-domains are produced.

For static partitioning, the method may be slower than state-of-the-art graph-based partitioning methods because the initial particle distribution may be far from the equilibrium state and the SPH evolution is explicit in time. However, for dynamic load-balancing, the method becomes efficient as typically the number of necessary iterations becomes quite small due to its implicitly incremental and physical localization properties.

Acknowledgements

The first author is partially supported by China Scholarship Council (No. 201206290022). The second author acknowledges the support from his current institute.

References

[1] D. Butz, Y. Gao, A.M. Kempf, N. Chakraborty, Large eddy simulations of a turbulent premixed swirl flame using an algebraic scalar dissipation rate closure, Combust. Flame 162 (9) (2015) 3180–3196.
[2] L.H. Han, X.Y. Hu, N.A. Adams, Adaptive multi-resolution method for compressible multi-phase flows with sharp interface model and pyramid data structure, J. Comput. Phys. 262 (2014) 131–152.
[3] H. Ji, F.S. Lien, E. Yee, A new adaptive mesh refinement data structure with an application to detonation, J. Comput. Phys. 229 (2010) 8981–8993.
[4] E. Boman, K. Devine, R. Heaphy, R. Hendrickson, V. Leung, L.A.C. Vaughan, Zoltan: Parallel Partitioning, Load Balancing and Data Management Services, User's Guide Version 3.0 2007, pp. 1–173.
[5] B. Hendrickson, K. Devine, Dynamic load balancing in computational mechanics, Comput. Methods Appl. Mech. Eng. 184 (2C4) (2000) 485–500.
[6] C. Kavouklis, Y. Kallinderis, Parallel adaptation of general three-dimensional hybrid meshes, J. Comput. Phys. 229 (9) (2010) 3454–3473.
[7] G. Karypis, V. Kumar, Metis: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, 1995.
[8] M. Berger, S. Bokhari, A partitioning strategy for nonuniform problems on multiprocessors, IEEE Trans. Comput. 36 (1987) 570–580.
[9] M.J. Berger, J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53 (3) (1984) 484–512.
[10] A.R. Butz, Space filling curves and mathematical programming, Inf. Control 12 (1968) 314–330.
[11] H. Sagan, Space-Filling Curves, Springer, 1994.
[12] G.V. Nivarti, M.M. Salehi, W.K. Bushe, A mesh partitioning algorithm for preserving spatial locality in arbitrary geometries, J. Comput. Phys. 281 (2015) 352–364.
[13] A. Pothen, H. Simon, K. Liou, Partitioning sparse matrices with eigenvectors of graphs, SIAM J. Matrix Anal. Appl. 11 (3) (1990) 430–452.
[14] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1999) 359–392.
[15] S.T. Barnard, H.D. Simon, Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems, Concurr., Pract. Exp. 6 (2) (1994) 101–117, http://dx.doi.org/10.1002/cpe.4330060203.
[16] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell Syst. Tech. J. 49 (2) (1970) 291–307.
[17] C.M. Fiduccia, R.M. Mattheyses, A linear-time heuristic for improving network partitions, in: 19th Conference on Design Automation, IEEE, 1982, pp. 175–181.
[18] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, C. Schulz, Recent advances in graph partitioning, CoRR, arXiv:1311.3144, http://arxiv.org/abs/1311.3144.
[19] B. Hendrickson, T.G. Kolda, Graph partitioning models for parallel computing, Parallel Comput. 26 (12) (2000) 1519–1534.

[20] H. Meyerhenke, B. Monien, S. Schamberger, Graph partitioning and disturbed diffusion, Parallel Comput. 35 (10–11) (2009) 544–569.
[21] R. Diekmann, R. Preis, F. Schlimbach, C. Walshaw, Shape-optimized mesh partitioning and load balancing for parallel adaptive fem, Parallel Comput. 26 (12) (2000) 1555–1581.
[22] F. Pellegrini, A parallelisable multi-level banded diffusion scheme for computing balanced partitions with smooth boundaries, in: Euro-Par 2007 Parallel Processing, in: Lect. Notes Comput. Sci., vol. 4641, 2007, pp. 195–204.
[23] A. Rama Mohan Rao, Parallel mesh-partitioning algorithms for generating shape optimised partitions using evolutionary computing, Adv. Eng. Softw. 40 (2) (2009) 141–157.
[24] U. Catalyurek, C. Aykanat, Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication, IEEE Trans. Parallel Distrib. Syst. 10 (7) (1999) 673–693.
[25] H. Meyerhenke, J. Gehweiler, On dynamic graph partitioning and graph clustering using diffusion, in: Dagstuhl Seminar Proceedings, 2010.
[26] H. Meyerhenke, B. Monien, T. Sauerwald, A new diffusion-based multilevel algorithm for computing graph partitions of very high quality, in: IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, 2008, pp. 1–13.
[27] J.J. Monaghan, Smoothed particle hydrodynamics, Rep. Prog. Phys. 68 (2005) 1703–1759.
[28] K. Andreev, H. Raecke, Balanced graph partitioning, in: Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, 2004, pp. 120–124.
[29] X.Y. Hu, N.A. Adams, A multi-phase SPH method for macroscopic and mesoscopic flows, J. Comput. Phys. 213 (2006) 844–861.
[30] X.F. Yang, M.B. Liu, S.L. Peng, Smoothed particle hydrodynamics modeling of viscous liquid drop without tensile instability, Comput. Fluids 92 (2014) 199–208.
[31] B. Osting, C.D. White, É. Oudet, Minimal Dirichlet energy partitions for graphs, SIAM J. Sci. Comput. 36 (4) (2014) A1635–A1651.
[32] L. Verlet, Computer experiments on classical fluids. I. thermodynamical properties of Lennard–Jones molecules, Phys. Rev. 159 (1967) 98–103.
[33] R. Courant, K. Friedrichs, H. Lewy, Über die partiellen Differenzengleichungen der mathematischen Physik, Math. Ann. 100 (1) (1928) 32–74.
[34] S. Diehl, G. Rockefelle, C.L. Fryer, D. Riethmiller, T.S. Statler, Generating optimal initial conditions for smooth particle hydrodynamics simulations, arXiv:1211.0525.
[35] R.W. Hockney, J.W. Eastwood, Computer Simulation Using Particles, Institute of Physics Publishing, 1998.
[36] L. Arge, M.D. Berg, H.J. Haverkort, K. Yi, The priority r-tree: a practically efficient and worst-case optimal r-tree, in: Proc. SIGMOD, Intl. Conf. Management of Data.
[37] G. Contreras, M. Martonosi, Characterizing and improving the performance of Intel threading building blocks, in: 2008 IEEE International Symposium on Workload Characterization, IISWC2008, 2008, pp. 57–66.
[38] J.J. Monaghan, J.B. Kajtar, SPH particle boundary forces for arbitrary boundaries, Comput. Phys. Commun. 180 (2009) 1811–1820.
[39] A. Ferrari, M. Dumbser, E.F. Toro, A. Armanini, A new 3d parallel SPH scheme for free surface flows, Comput. Fluids 38 (2009) 1203–1217.
[40] A. Leroy, D. Violeau, M. Ferrand, C. Kassiotis, Unified semi-analytical wall boundary conditions applied to 2-D incompressible SPH, J. Comput. Phys. 261 (2014) 106–129.
[41] W. Dehnen, H. Aly, Improving convergence in smoothed particle hydrodynamics simulations without pairing instability, Mon. Not. R. Astron. Soc. 425 (2012) 1068–1082.