Fakultät für Maschinenwesen

Lehrstuhl für Aerodynamik und Strömungsmechanik

Numerical methods and high-performance computing for Smoothed Particle Hydrodynamics

Zhe JI

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Julija Zavadlav
Prüfer der Dissertation:	1. Prof. Dr.-Ing. Nikolaus A. Adams
	2. Prof. Takayuki Aoki
	3. Priv.-Doz. Dr.-Ing. habil. Xiangyu Hu

Die Dissertation wurde am 31.08.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 17.02.2021 angenommen.

Declaration of Authorship

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Zhe Ji
April 16, 2021

zhe.ji@tum.de

Released April 16, 2021
Typesetting LaTeX

谨以此文献给我的所有老师们。

This work is dedicated to all the mentors in my life.

泉涸，鱼相与处于陆，相呴以湿，相濡以沫，不如相忘于江湖。

—— 庄子

Abstract

This cumulative thesis presents several novel numerical methods and parallel techniques relating to Smoothed Particle Hydrodynamics (SPH). More specifically, a new multi-resolution parallel framework for SPH, a Lagrangian Inertial Centroidal Voronoi Particle (LICVP) method for dynamic load balancing of particle-based method and a consistent multi-phase SPH formulation for parallel unstructured isotropic mesh generation are proposed.

A new multi-resolution parallel framework is proposed for the large-scale simulations with SPH. In contrast to the classical tree-based SPH solver, several novel algorithms are developed and integrated into the unified framework. By developing an adaptive rebalancing criterion and a monitoring system, the Centroidal Voronoi Particle (CVP) partitioning method is integrated, which guarantees the convergence to target partitioning with a good load balancing and an optimized communication volume. In order to construct ghost buffer particles in remote processors, a localized table-based hierarchical data structure is developed in cooperation with a tailored parallel fast neighbor search (PFNS) algorithm. Comparing to tree-based methods, the proposed data structure features lower querying cost and can be built with higher efficiency. Moreover, to overcome the bottleneck of graph construction time, the concept of "diffused graph" is proposed to improve the performance of the graph-based communication strategy developed in [1]. With the above algorithms integrated, the framework achieves a two-level parallelism, where an inter-node coarse-grained parallelization is handled by Message Passing Interface (MPI) [2] and Threading Building Blocks (TBB) [3] is employed for the fine-grained parallelization inside each node. A wide range of gas dynamics benchmarks is investigated to demonstrate the capability of the framework and its unique characteristics. Intensive performance tests show that a scalable performance is achieved.

To address the additional target encountered in dynamic load balancing and to extend the original CVP method [4] as a rebalancer, a LICVP method is developed. Two key concepts are proposed in LICVP. First, a background velocity is introduced to transport Voronoi particles according to the local fluid field, which facilitates data reuse and lower data redistribution cost. Second, in order to handle problems with skew-aligned computational load and large void space, an inertial-based partitioning strategy is proposed. An inertial matrix is utilized to characterize the load distribution and to confine the motion of Voronoi particles dynamically adapting to the physical simulation. Intensive numerical tests in fluid dynamics reveal that the underlying LICVP method improves the incremental property remarkably without sacrificing other objectives, i.e. the inter-processor communication is optimized simultaneously, and the repartitioning procedure is highly efficient.

A consistent parallel unstructured mesh generator based on a multi-phase SPH method is developed. This work extends the original SPH-based mesh generator [5] to a parallel context and to three dimensional mesh generation. To characterize the target function for both partitioning the geometry and distributing the mesh-vertexes, a unified density function is defined. By employing the same set of consistent governing equations, the proposed method achieves the targets of domain

decomposition, communication volume optimization and high-quality unstructured mesh generation simultaneously. The target of communication reduction is achieved by introducing a surface tension model between distinct partitioning sub-domains, which are characterized by colored SPH particles. While the communication volume being optimized on the sub-domain interface, the target of mesh generation is calculate in the meantime for regions inside the sub-domain following the same fluid relaxation analogy. Once the target of domain decomposition is achieved, the surface-tension force is removed gradually. Consequently, the mesh quality near the interface area is improved. Since a steady state has already been achieved, the local mesh-quality optimization brings minor effect to the compact shape of sub-domains. The governing equations are solved by a multi-phase SPH formulation implemented in the parallel framework developed in this work. A set of benchmarks consisting various scales and complexities is tested. Results show that all the optimization targets are achieved consistently within the proposed method.

Acknowledgements

First of all, I would like to thank Prof. Nikolaus A. Adams and PD Dr. Xiangyu Hu for providing me the opportunity to pursue my four-year Ph.D study in TUM, Germany. It is a great experience to get inspiring suggestions from Prof. Adams in the weekly group meeting and the very detailed revising for my papers, which help a lot in building a big picture for my research. I would like to thank PD Dr. Xiangyu Hu for the detailed discussions and his constructive suggestions, which teaches me to be critical to ideas and to think different.

I would like to thank China Scholarship Council for the support during the four-year program. It gives me a great opportunity to see, to learn and to become innovative.

Special thanks to Dr. Lin Fu. I enjoyed working with you, both as a colleague and a friend. I will never forget those "coding nights" and "academic weekends", when we worked tirelessly to overcome problems, encouraged each other to stay strong, fought for our own standpoints and celebrated the birth of new ideas.

Also many thanks to my colleagues and friends in Altair and formerly Fluidyna, where I work previously as a part-time student and now as a full-time engineer. Thank you, Milos, Nima, Michael, Jiayu, Gareth, Eric, Sam and Anxo, for all the support and love as a team. Also, my gratitude to the colleagues in the ultraFluidX team and from other offices. I feel proud of being one of you.

I also want to thank Prof. Fei Xu for her guidance and support since my years in the graduate school. Thank you for providing kind suggestions and warm encouragement during my PhD study.

My gratitude extends to Prof. Takayuki Aoki for agreeing to be the committee member and to Prof. Dr. Julija Zavadlav for agreeing to be the chair of my thesis defense.

I am also thankful to my colleagues Rongzong Huang, Chi Zhang, Jianhang Wang, Shucheng Pan, Luhui Han, Yue Li, Jingyu Wang, Xiuxiu Lyu and many others. They made my stay in Munich full of fun. I also want to thank staff members of our chair, Amely Schwörer, Angela Grygier, Dr. Christian Stemmer and many others, for their support and help in daily life and scientific work. Also I would like to acknowledge Prof. Xiang Yang for inviting me to a joint visiting program to the Penn State University during the summer in 2018.

Finally, and most importantly, my deepest gratitude goes to my parents and the love of my life. This dissertation would not be possible without your love and support. I know words are not enough, and I will always be with you.

Contents

Nomenclature

AM	Additive Manufacturing
AFT	Advancing Front Method
ASPH	Anisotropic SPH
BEM	Boundary Element Method
CFD	Computational Fluid Dynamics
CVP	Centroidal Voronoi Particle
CVT	Centroidal Voronoi Tessellation
CPU	Central Processing Unit
CLL	Cell Linked List
CFL	Courant-Friedrich-Lewy
CUDA	Compute Unified Device Architecture
DSM	Distributed Shared Memory
DM	Molecular Dynamics
DPD	Dissipative Particle Dynamics
EOS	Equation Of State
FSI	Fluid Structure Interaction
FDM	Fused Deposition Modeling
FEM	Finite Element Method
FVM	Finite Volume Method
GSPH	Godunov SPH
GPU	Graphics Processing Unit
HPC	High-Performance Computing
LBM	Laser Beam Melting
LAPD	Locality-Aware Parallel Delaunay
MHD	MagnetoHydroDynamics
MPI	Message Passing Interface
MUSCL	Monotonic Upwind Scheme for Conservation Laws
NNS	Nearest Neighbor Search
NASA	National Aeronautics and Space Administration
PFNS	Parallel Fast Neighbor Searching
PDE	Partial Differential Equation
PODM	Parallel Optimistic Delaunay Mesh
PAFT$_{SM}$	Parallel AFT for Shared Memory Computers
P^{2}DM	Parallel Projective Delaunay Meshing
PD3	Parallel Delaunay Domain Decoupling
POAFT	Parallel Octree AFT
PCDM	Parallel Constrained Delaunay Meshing
RCB	Recursive Coordinate Bisection
RHS	Right Hand Side
RIB	Recursive Inertial Bisection

RSB	Recursive Spectral Bisection
SPH	Smoothed Particle Hydrodynamics
SFC	Space Filling Curve
SOA	Structure Of Arrays
TBB	Threading Building Blocks
VP	Voronoi Particle
VL	Verlet List
WENO	Weighted Essentially Non-Oscillatory

Chapter 1

Introduction

In this thesis, my research during the four years (09.2015 − 09.2019) in the Chair of Aerodynamics and Fluid Mechanics (AER) at the Technical University of Munich (TUM) is presented. During this period, I have been working on novel numerical methods and the HPC for SPH.

The thesis is arranged as following: In Chapter 1, the motivation of the thesis is first introduced. Then a general overview of the scientific backgrounds is presented. Lastly, the objectives of the entire thesis are summarized. In Chapter 2, several numerical methods are introduced: 1) A Godunov SPH formulation for compressible gas dynamics is first reviewed. 2) Popular sequential and parallel unstructured mesh generation methods are discussed and a novel SPH-based isotropic/anisotropic mesh generation method is introduced. 3) Lastly, the level-set function is briefed. In Chapter 3, several essential technical aspects, i.e. domain decomposition method, PFNS and communication strategy, and recent progress in HPC for particle-based methods are discussed. Chapter 4 summarizes the main accomplishments achieved in the current thesis. Lastly, concluding remarks and future work are presented in Chapter 5. The e-prints of the publication accomplished in the current thesis are attached in Appendix A.

1.1 Motivation

The research in this thesis is mainly supported by the China Scholarship Council (CSC). Originally, the research topic is "Unified Multi-resolution Modeling for Both Solid and Fluid Dynamics with Smoothed Particle Hydrodynamics". The long-term goal is to develop an innovative and high-performance numerical modeling framework using advanced SPH method that tackles multi-physics problems involving both fluid and solid dynamics. Typical examples range from the modeling of asteroid impact processes [6], stellar collision and collapse problems in computational astrophysics [7] to the modeling of Laser Beam Melting (LBM) in additive manufacturing (AM) technology [8]. The numerical modeling of these problems is highly challenging due to the intrinsic complexity of the underlying physics. The embedded numerical solvers need to capture the dynamic response and mechanical behavior of the system/material subjecting to a wide range of extreme phenomena, e.g. shock waves, melting, vaporization, radiation, Marangoni convection and etc. These topics are beyond the realm of traditional computational fluid dynamics and computational solid mechanics. Interdisciplinary knowledges are required in order to develop a generalized modeling technique. Besides the numerical solver, advanced discretization technique, e.g. mesh generation and mesh adaptation technique, is required too to create an accurate and efficient discretized representation of the geometry/domain being simulated. This is particularly critical and also difficult for problems featuring complex interfaces and large variation of spatial scales. Lastly,

to overcome the bottleneck of computational time and memory consumption, all the aforementioned algorithms and solvers are in tremendous need for massive parallelization. Comparing to numerical improvements that can better characterize the underlying physics, it is equally challenging and imperative nowadays to develop parallel codes that can achieve a scalable performance in clusters featuring tens of thousands of cores.

The above description gives a grand picture of the general motivation of this thesis, which can be further classified into three finer-grained tasks, i.e. 1) generalized SPH formulation for multi-material and multi-physics modeling involving both fluids and solids, 2) novel parallel mesh generation and adaptation method, and 3) scalable multi-resolution parallel framework for SPH. The first task is partially addressed by Dr. Chi Zhang, who is one of the team members in AER. He has recently graduated with thesis titled as "Smoothed particle hydrodynamics for fluid and solid dynamics". For more details, I refer to his featured publication [9]. The main focus of this work then resides in the last two tasks. In the following two sections, an overview of the scientific background relating to the objectives of the current thesis is presented.

1.2 An overview on SPH and its applications

In 1977, SPH was first introduced independently by Lucy [10] and Gingold and Monaghan [11]. Due to the simplicity in coupling with a gravitational solver, SPH first gained its success in computational astrophysics [12][13][14]. Unlike the mesh-based methods, SPH, as a mesh-free method, employs a set of arbitrarily distributed particles to discretize the computational domain. By introducing a smoothing kernel function, any field value and its derivatives of particle i can be expressed by a summation over its neighboring particles enclosed in the cutoff range. Hence, the numerical solution to the governing equations can be calculated on the particles. To advance the system in time, particles are updated in a Lagrangian fashion following the local structures of the material, thus the Galilean invariance is ensured. The Lagrangian property is the most important advantage of SPH, which allows it to follow the nature development of the underlying physics. As a result, it is particularly suitable for simulating problems evolving large deformations and complex interface motions, e.g. dambreaks [15] and oil flow in a gearbox [16]. Moreover, since no mass transfer is calculated between particles, the constraint of sharp interface condition can be easily achieved by assigning particles of same phase with a unique color function. Consequently, when applied to multi-phase simulations, an explicit front-tracking method is no longer needed and the necessary of handling the triple-point issue is eliminated too. Another advantage of SPH is that the coupling with other physics or solvers is generally easy, since forces and source terms can be directly imposed on individual particles.

Due to the aforementioned advantages, SPH has been applied to many research topics and demonstrated successful in resolving a wide variety of complex phenomena. E.g. star formation [12][14], magnetohydrodynamics (MHD) [17], free-surface flows [18][19][20], multi-phase flows [21][22], solid mechanics [23][9], and fluid structure interaction (FSI) [24][25][26]. These are the "hospot" for SPH since 1977. Moreover, evidents show that an increasing amount of commercial softwares, e.g. Altair nanoFluidX, PreonLab, SPH-flow, dive solutions and etc., have been developed since 2010, which all feature SPH as their primary/sole numerical solver. Most use cases of these softwares fall into the above fields, excluding computational

astrophysics, which implies that a stronger industrial relevance has been established and it is competitive enough against mesh-based CAE softwares in those fields.

Other than in the "traditional" area of strength, recently there has been some worth-noticing novel applications that further extend the capability of SPH. In [27], SPH is used to study the Fused Deposition Modeling (FDM) 3D printing process of fiber-reinforced polymers. To characterize the fiber suspension, a tensor-based microstructure constitutive model is developed. Moreover, SPH is also used to model the SLM process in AM. In [8], a multi-phase SPH formulation is developed incorporating a variety of physical phenomena, e.g. thermal conduction, melting, resolidification, convection and thermocapillarity effect. In addition, the influence of the recoil pressure induced by evaporating atoms from the melt pool surface is included too. These two applications demonstrate that SPH has huge potentials in handling multi-physics phenomena involving both fluid and solid.

Another type of application that has been exploit recently is to solve a mathematical problem by finding the solution to a set of governing equations based on a physical analogy. First, a SPH-based partitioning method is proposed in [28]. The method is motivated by the observation that an optimum partitioning has high analogy to the relaxation of a multi-phase fluid to steady state. Then a set of physics-motivated model equations are developed to characterize the underlying mesh topology, and are solved by a multi-phase SPH method. The main advantage of this method is that all the optimization objectives are achieved implicitly during the particle relaxation procedure. Later, a SPH-based isotropic unstructured mesh generation method is developed [5] and further extended to the anisotropic mesh generation [29], following similar methodology. In contrast to the partitioning method, the target field for mesh generation is discretized on a background Cartesian mesh and defined considering geometry information and user-defined inputs to characterize the target feature-size distribution inside the mesh generation region. To describe the geometry for mesh generation, the level-set method is employed. The main advantage comparing to classical mesh generation method is that this method is well-suitable for massive parallelization and no explicit triangulation/tetrahedralization is required to obtain a quality-guaranteed mesh. More details regarding this method will be elaborated in Chapter 2.

1.3 High-performance computing for SPH

Despite the aforementioned advantages and developments in SPH, one of the main drawbacks of this method is the high computational cost. Due to the smoothing kernel, the number of operations used for solving the governing equations per particle per step is significantly higher than other standard grid based methods. Therefore, more computational time is generally expected for SPH simulations. On the other hand, the pair-wised interaction pattern in SPH is well-suitable for the newer-generation HPC units, e.g. the graphics processing unit (GPU). From various researches [30][31][32], it is observed that when parallelized in GPU, a speedup of one to two orders of magnitude can be achieved comparing to running on a single CPU card. For industrial applications, the concept of HPC is basically the trademark for the latest SPH-based commercial CFD softwares, i.e. all of them are either accelerated by GPU or intensively optimized for multi-core CPUs running on small DSM clusters or in the "cloud". Considering the advantage in handling large deformations, these commercial codes can reduce the "turnaround time" from days to hours

comparing to the grid-based counterparts in the area of water management [33] and oil-bath lubrication systems [34][16].

Although high-performance is achieved for the codes reviewed above, they are limited by a constant-resolution constraint. The efficiency in nearest neighbor search (NNS) is accomplished with a single-level cell linked list (CLL), which becomes inefficient if multiple resolutions are presented in the domain. To support variable resolution, normally a tree data structure is required. The tree-based SPH solver is well-established in computational astrophysics community and shares the same data structure with a N-body solver for gravity force calculation [12]. To perform NNS, a tree-walk operation is required to traverse neighboring cells and to find the neighboring particles within the cutoff range. The tree-walk is computationally more expensive comparing to the NNS in CLL. Moreover, the maintenance of the tree is cumbersome in a parallel environment. More details on the NNS and tree-based data structure is arranged in Chapter 3 and Chapter 4.

Another challenge in the HPC for SPH is to properly distribute the computational load to a set of computational nodes, where each node has separate physical memory space and calculation units and cables are used to connect all the node into a network, i.e. the cluster. The structure of the cluster may be homogeneous, where each node shares the same computational power. Then the total computational load needs to be divided into equal pieces. However, in clusters featuring heterogeneous configurations, e.g. the CPU-GPU hybrid cluster, scheduling of the load is more challenging since each cluster may be different and the resources received may be different too. Consequently the optimal scheduling of the load in each simulation should be adjusted accordingly. Once the load is divided and distributed, inter-node data communication can not be avoided, since particles inside each node may require information from neighboring nodes to complete the kernel support. Therefore, in each timestep, a subset of particle information, e.g. position, mass, velocity and etc., needs to be communicated to neighboring sub-domains. Based on different partitioning of the computational load, the resulting total amount of communication, referred as communication volume, may be different too, which implies different computational overhead. Hence, the optimization of communication volume is another critical issue in parallel computing. To address the issue of load scheduling and communication-volume optimization, a proper partitioning/domain decomposition method is required. Classical partitioning method can be concluded as the geometry-based and graph-based method, which is introduced in detail in Chapter 3.

1.4 Objectives

Based on the motivation section and the general overview of the scientific backgrounds, I summarize the objectives of the present work as following.

The first objective of the present work is to develop a new multi-resolution parallel framework for SPH combining several algorithms recently developed in AER. The framework should be able to achieve the two-level parallelism by utilizing both multi-threading and MPI techniques. The primary target is to achieve a scalable performance in sate-of-the-art clusters with adaptive resolution. The work is presented in paper I

- Ji, Z., Fu, L., Hu, X.Y. and Adams, N.A., 2019. A new multi-resolution parallel framework for SPH. *Computer Methods in Applied Mechanics and Engineering*, **346**, pp.1156-1178,

which is available in Appendix A.1.

The recently proposed physics-motivated domain decomposition method, i.e. the CVP method [4], previously is only validated in static partitioning problems. The performance of CVP as a load rebalancer needs to be assessed. Several issues should be investigated, e.g. whether the locality property is preserved and the communication volume is optimized after each rebalancing, and is the incremental property achieved for the load balancer. Moreover, the issue of skew-aligned load distribution appeared in particle-based simulations, e.g. the tsunami simulation in free-surface flows, may cause severe deterioration of the parallel performance. The second objective of this work is to extend the original CVP to the context of dynamic load balancing and to address the additional targets encountered in rebalancing the system. This work is presented in paper II

- Ji, Z., Fu, L., Hu, X.Y. and Adams, N.A., 2019. A Lagrangian Inertial Centroidal Voronoi Particle method for dynamic load balancing in particle-based simulations. *Computer Physics Communications*, **239**, pp.53-63,

which is attached in Appendix A.2.

The last objective of the present work is to develop a parallel unstructured mesh generation method based on a consistent SPH-based formulation. The target is to achieve 1) domain decomposition of the mesh generation region and 2) an optimized adaptive unstructured mesh simultaneously. This work also intends to extend the original SPH-based mesh generator proposed in [5] to three-dimensional meshes. The new parallel mesh generator is presented in

- Ji, Z., Fu, L., Hu, X. and Adams, N., 2020. A consistent parallel isotropic unstructured mesh generation method based on multi-phase SPH. *Computer Methods in Applied Mechanics and Engineering*, **363**, p.112881,

and detailed in Appendix A.3.

Chapter 2

Numerical methods

In this chapter, numerical methods used in this thesis are reviewed. The governing equations and numerical concepts regarding SPH with adaptive smoothing length are presented in Section 2.1. In Section 2.2, both the serial and parallel unstructured mesh generation methods are reviewed. The SPH-based isotropic and anisotropic mesh generator are introduced too. The level-set method is introduced in Section 2.3 for geometry definition.

2.1 SPH with adaptive smoothing-length

In SPH, since the domain is discretized with particles carrying specific mass and particles are advanced in a Lagrangian fashion, the solution naturally follows the local flow structures. In compressible gas dynamics, benefiting from this characteristics, particle smoothing-length [14] adapts to the flow density, i.e. in high-compression regions smaller smoothing-length is obtained. Therefore, the resolution is automatically adaptive, which is an attractive feature for simulating gas dynamics. Recently in [35], results demonstrate that with an appropriate initial particle setup, SPH achieves competitive performance comparing to some of the high-order shock capturing schemes, such as WENO5. In incompressible/weakly-compressible hydrodynamics, the possibilities to incorporate adaptive-resolution have been explored intensively in recent years [36][37][38][39], and significant advances have been made. By allowing the resolution to be locally adaptive, more complex industrial applications involving large scale differences and velocity/pressure gradients, e.g. simulating the oil bath lubrication systems [16], water management problems in automotive industry, and etc., can be handled in the future.

In the current section, a Godunov SPH scheme designed for compressible gas dynamics is reviewed briefly. For more technical details, I refer to [40][41][42].

2.1.1 Governing equations

The Euler equation of an idea inviscid gas can be expressed as the conservation of mass, momentum and energy [12]

$$\frac{d\rho}{dt} = -\rho \bigtriangledown \cdot \mathbf{v}, \tag{2.1}$$

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}(\bigtriangledown p), \tag{2.2}$$

$$\frac{de}{dt} = -\frac{p}{\rho} \bigtriangledown \cdot \mathbf{v}, \tag{2.3}$$

where ρ is the density, p the pressure, e the specific thermal energy, $\mathbf{v}$ the velocity vector.

In addition to the above equations, the positions of particles ($\mathbf{r}$) are updated in a Lagrangian way following

$$\frac{d\mathbf{r}}{dt} = \mathbf{v}. \tag{2.4}$$

To close the system, an equation of state (EOS) is required. For idea gas, the EOS is

$$p = (\gamma - 1)\rho e, \tag{2.5}$$

where γ is the adiabatic index.

2.1.2 The smoothing kernel

The governing equations are solved on discrete SPH particles. By cooperating with a smoothing function W, any field quantity ϕ at coordinate $\mathbf{r}$ can be calculated by

$$\bar{\phi}(\mathbf{r}) = \int \phi(\mathbf{r}')W(\mathbf{r} - \mathbf{r}', h)d\mathbf{r}'. \tag{2.6}$$

where $\mathbf{r}'$ is the coordinate of an integration point, and $d\mathbf{r}'$ is a differential volume element.

Similarly, the gradient of ϕ can be calculated through a convolution integral over the cutoff range of the kernel function W as

$$\bigtriangledown \bar{\phi}(\mathbf{r}) = \int \bigtriangledown\phi(\mathbf{r}')W(\mathbf{r} - \mathbf{r}', h)d\mathbf{r}'. \tag{2.7}$$

The radially symmetric smoothing function (kernel) W should reduce to a delta function, i.e.

$$\lim_{h\to 0} W(\mathbf{r} - \mathbf{r}', h) = \delta(\mathbf{r} - \mathbf{r}'), \tag{2.8}$$

in the limit of a vanishing smoothing-length h. Moreover, to reproduce a constant field, the partition of unity property, i.e. $\int W(\mathbf{r} - \mathbf{r}', h)d\mathbf{r}' = 1$, is required.

The choice of the smoothing kernel is not exclusive. On the contrary, various types have been developed in the literature. Among all the choices, the Wendland kernels [43] and B-splines [44] are mostly employed, and a comprehensive study on the performance of different kernel functions can be found here [45].

2.1.3 Numerical discretization

By calculating the summation of the weights over all the neighboring particles j, the discrete form of Eq. 2.6 with respect to particle i can be written as

$$\bar{\phi}(\mathbf{r}_i) = \sum_j \phi(\mathbf{r}_j)W(\mathbf{r} - \mathbf{r}_j, h)V_j, \tag{2.9}$$

where V is the particle volume [46].

Regarding the discretized gradient calculation, by integrating by parts and assuming a compact kernel support, i.e. the cutoff region of particle i is entirely inside the computational domain, Eq. 2.7 can be approximated as

$$\bigtriangledown \bar{\phi}(\mathbf{r}) = -\int \phi(\mathbf{r}') \bigtriangledown W(\mathbf{r} - \mathbf{r}', h)d\mathbf{r}'. \tag{2.10}$$

A direct discretization can be expressed as

$$\bigtriangledown \bar{\phi}(\mathbf{r}) = -\sum_j \phi(\mathbf{r}_j) \bigtriangledown W(\mathbf{r} - \mathbf{r}_j, h) V_j. \tag{2.11}$$

However, this operator is not conservative if applied to the pair-wised particle interaction. For more gradient operators developed, I refer to [47] for a comprehensive overview.

In this thesis, the Godunov SPH scheme is implemented following [40][41][42].

Instead of directly solving the continuity equation, the density is calculated via summation by

$$\rho_i = \sum_j m_j W(\mathbf{r}_i - \mathbf{r}_j, h_i). \tag{2.12}$$

As mentioned before, the change of particle density will affect the particle "size", i.e. smoothing length h. In order to ensure a smoothed transition of h, an iterative approach is employed following [17].

The discretized momentum and energy equation for GSPH are

$$\frac{d\mathbf{v}_i}{dt} = -\sum_j m_j p^*_{ij} \left(\frac{1}{\rho_i^2} \nabla W_{ij}(h_i) + \frac{1}{\rho_j^2} \nabla W_{ij}(h_j) \right), \tag{2.13}$$

$$\frac{de_i}{dt} = -\sum_j m_j p^*_{ij} \left(\mathbf{v}^*_{ij} - \mathbf{v}^*_i \right) \cdot \left(\frac{1}{\rho_i^2} \nabla W_{ij}(h_i) + \frac{1}{\rho_j^2} \nabla W_{ij}(h_j) \right), \tag{2.14}$$

where, $\mathbf{v}^*_i$ represents the time centered velocity for particle i. The starred quantity p^*_{ij} and $\mathbf{v}^*_{ij}$ are the intermediate states computed by solving an one-dimensional Riemann problem.

The main advantage of employing a Riemann solver is that numerical dissipation is introduced implicitly and no parameter tunning is required comparing to the traditional artificial viscosity based SPH schemes. The Riemann problem is solved at the imaginary interface (see S^*_{ij} in Fig. 2.2) on the connecting vector of the interacting particle pair. The interface position is calculated considering the smoothing length of particle i and j

$$S^*_{ij} = \frac{h_i}{h_i + h_j} \mathbf{r}_{ij}. \tag{2.15}$$

The input of the Riemann solver, i.e. left and right state, is defined as

$$\begin{cases} \mathbf{U}_r = (p_r, u_r, \rho_r), \\ \mathbf{U}_l = (p_l, u_l, \rho_l). \end{cases} \tag{2.16}$$

To solve the Riemann problem, various options are available. Practically, to avoid the expensive calculation raised by solving the exact Riemann solver, non-iterative approximate Riemann solvers can be employed. Fig. 2.1 illustrates a simplified Riemann fan with two intermediate states. In general, the choice of approximate Riemann solvers, e.g. the local Lax-Friedrichs solver [48], the ROE solver [49], the HLLC solver [50], the Ducowicz solver [51] and etc., may affect the performance of the result to different extent. I refer to [42] for a detailed study.

Different approaches can be applied to achieve different accuracies of the input states. With piece-wise constant assumption, i.e. particle i being the right state and

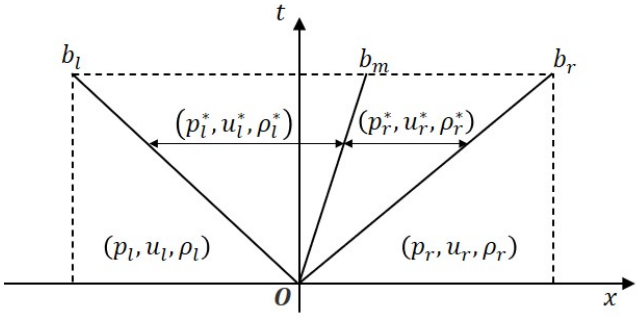


FIGURE 2.1: Sketch of the simplified Riemann fan with two intermediate states.

particle j the left state, the 1st-order reconstruction can be achieved

$$\begin{cases} \mathbf{U}_r = (p_i, \mathbf{v}_i \cdot \mathbf{e}_{ij}, \rho_i), \\ \\ \mathbf{U}_l = (p_j, \mathbf{v}_j \cdot \mathbf{e}_{ij}, \rho_j), \end{cases} \tag{2.17}$$

where $\mathbf{e}_{ij} = \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|}$.

To achieve higher-order accuracy, a number of methods, e.g. the Monotonic Upwind Scheme for Conservation Laws (MUSCL) scheme [52] and the Weighted Essentially Non-Oscillatory (WENO) scheme [53], can be employed. By assuming piecewise linear reconstruction, second-order accuracy can be achieved at the interface in smooth regions, i.e.

$$\begin{cases} \mathbf{U}_r = \mathbf{U}_i + (\frac{\partial \mathbf{U}}{\partial s})_i [s_{i,j}^* + C_{s,i}\frac{\delta t}{2} - s_i], \\ \\ \mathbf{U}_l = \mathbf{U}_j + (\frac{\partial \mathbf{U}}{\partial s})_j [s_{i,j}^* - C_{s,j}\frac{\delta t}{2} - s_j], \end{cases} \tag{2.18}$$

where $C_{s,i}$ is the sound speed of particle i, s_i the distance between particle i and interface $s_{i,j}^*$ and δt the timestep size. To enforce the monotonicity near discontinuities, the harmonic gradient averaging based limiter proposed by Van Leer [54] can be applied. For detailed explanation I refer to Ref. [40]. Fig. 2.2 gives a sketch of the piecewise-linear reconstruction of primitive variables $\mathbf{U}$ along the connecting vector of particle i and j.

2.1.4 Time integration

To update the system, the gradients of primitive variables $\mathbf{U}$, i.e. $\frac{\partial \mathbf{U}}{\partial s}$, are first calculated for reconstructing the inputs of the Riemann problem. Then, the RHS of Eq. 2.13 and Eq. 2.14 are calculated.

The time centered velocity $\mathbf{v}_i^*$ is obtained with acceleration $\frac{d\mathbf{v}_i}{dt}$

$$\mathbf{v}_i^*(t + \frac{\delta t}{2}) = \mathbf{v}_i(t) + \frac{\delta t}{2}\frac{d\mathbf{v}_i}{dt}. \tag{2.19}$$

The position, velocity and energy are updated respectively following

$$\mathbf{x}_i(t + \delta t) = \mathbf{x}_i(t) + \delta t \mathbf{v}_i^*, \tag{2.20}$$

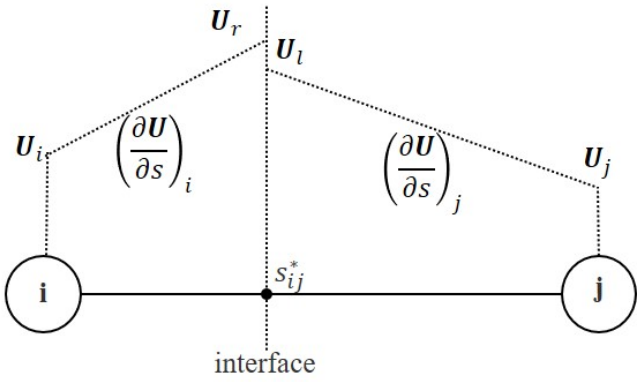


FIGURE 2.2: Sketch of the piecewise-linear reconstruction of primitive variable **U** along the connecting vector of particle i and j.

$$\mathbf{v}_i(t+\delta t) = \mathbf{v}_i(t) + \delta t \frac{d\mathbf{v}_i}{dt}, \tag{2.21}$$

$$e_i(t+\delta t) = e_i(t) + \delta t \frac{de_i}{dt}. \tag{2.22}$$

Once the new particle position is calculated, the new density and smoothing length are updated iteratively using the Newton-Raphson method to ensure an approximately similar number of neighbors for all particles

$$\rho_i(t+\delta t), h_i(t+\delta t) = \mathbb{R}(\rho_i(t), h_i(t)), \tag{2.23}$$

where $\mathbb{R}(\rho, h)$ denotes the function to calculate density and smoothing-length. For detailed algorithm, I refer to [17][14].

The timestep-size δt is calculated considering the Courant condition to ensure the numerical stability of simulation

$$\delta t = C_{CFL} \min_i \{h_i / C_i\}, \tag{2.24}$$

where C_{CFL} is the CFL constant.

2.2 Unstructured mesh generation

The problem of mesh generation is defined as decomposing the interior of a physical domain Ω into a mesh M of simple and "well-shaped" elements such as boxes or simplices with a bounded aspect ratio [55]. The elements comprising a mesh should cover the entirety of the domain without overlapping. Mesh generation is a key step for numerical methods, e.g. the Finite Element method (FEM), Boundary Element method (BEM), Finite Volume method (FVM) and etc., to find a proper approximating solutions of the underlying partial differential equations (PDE). The topic of mesh generation was brought into attention in the late 1970s, due to the increasing demands from the areas of computational mechanics and computational fluid dynamics. At that time the process of constructing a mesh can take orders of magnitude more hours than perform and analyze the solution on the mesh [56]. Four decades later, according to the NASA CFD vision 2030 study [57], quality-guaranteed mesh generation and adaptation is still a recurring bottleneck in CFD and more efforts are needed.

Quality-guaranteed and high-performance mesh generation is a highly interdisciplinary topic, which involves computational geometry, data structures, applies mathematics and parallel computing. New challenges has raised recently due to the rapidly growing capabilities and capacities of modern supercomputers. Developing parallel mesh generation codes is essential to achieve an acceptable turnaround time in complex industrial applications involving complex geometry boundaries and adaptive mesh topologies. To fully exploit the computational horsepower and to reduce the communication overhead in the distributed memory systems, the parallel mesh generator needs to resolve various difficulties, e.g. the data dependency, the load balancing and the irregular behavior of the mesh refinement [58].

Regarding to serial unstructured (triangular and tetrahedral) mesh generation, tremendous advancements have been made recently [59]. The most well-established ones can be classified into five categories: (1) advancing front methods (AFT) [60][61]; (2) Octree refinement-based methods [62][63]; (3) Delaunay refinement-based methods [64][65]; (4) Delaunay variational-based methods [66][67]; and (5) particle-based methods [5][68][69]. More details can be found in the following Section 2.2.1.

Additional constraints arise when the mesh is generated in parallel in a distributed memory system. As described from a review in 2006 [70]:

> The challenges in parallel mesh generation methods are: to maintain stability of the parallel mesher (i.e., retain the quality of finite elements generated by state-of-the-art sequential codes) and at the same time achieve 100% code re-use (i.e., leverage the continuously evolving and fully functional off-the-shelf sequential meshers) without substantial deterioration of the scalability of the parallel mesher.

Therefore, a consistent formulation of the mesh generator is required to achieve the additional goals in a parallel environment. A key factor in improving the performance of the parallelized mesh generation method is to seek a balance between generating a mesh in each subdomain and the inter-domain boundaries. Depending on the data synchronization strategy, the parallel mesh generation can be categorized into tightly-coupled, partially-coupled and decoupled approaches [70].

Recently, a SPH-based unstructured mesh generator [5] has been developed, and then further extended to the adaptive anisotropic mesh generation in [29]. The key idea is to analogize the optimization of mesh-vertices distribution to the Lagrangian particle relaxation procedure guided by a target density field. The relative discrepancy of particle density and target density is characterized as pseudo pressure, which is calculated by a tailored EOS. The pressure gradient results in pair-wise particle interaction force and drives particles towards target density distribution while maintaining a regularized and isotropic distribution [71]. Similar procedure is applied to the anisotropic mesh generation by employing an anisotropic SPH (ASPH) formulation.

One of the main appealing features is that the SPH-based mesh generation method is inherently suitable for the large-scale parallel computing. Since the pair-wise particle interaction is completely localized, it fulfills the fine-grained parallelism naturally. With a proper domain decomposition method, the coarse-grained parallelism can be achieved efficiently. As shown in multiple researches [72][73][74], scalable performances are exhibited for SPH solvers on large number of nodes. Moreover, SPH is well-known for being easy to program and maintain using modern parallel techniques, e.g. MPI [2], OpenMP [75] and CUDA [76], which facilitates the maintainability of the underlying mesh generator.

Another advantage over other mesh generation methods is that the mesh-generation and mesh-optimization targets are achieved implicitly without relying on an explicit mesh. By modeling the motions of particles with a set of physics-motivated governing equations and interpolating particle values using an isotropic smoothing kernel function, the system converges to an optimized solution once all pair-wised particle forces vanish. The triangulation/tetrahedralization is only needed once after the convergence error is below a predefined threshold [77].

In the reset if this section, a set of well-established serial mesh generation methods is reviewed in Sub-section 2.2.1. The development of parallel mesh generators is introduced in Sub-section 2.2.2. The SPH-based isotropic and anisotropic unstructured mesh generation method are presented in Sub-section 2.2.3.

2.2.1 Serial methods

- *The advancing front method* generates a mesh starting from the domain boundary and advances towards the "inside" of the geometry. In each iteration, elements are created based on the positions of existing meshes until a new layer is constructed. The free surfaces/edges of the newly created meshes are referred as the "front", which marches forward until the closure of the mesh-generation region [60]. A sequence of increasing dimension has to be followed in order to mesh all the features, i.e. the boundaries of the geometries need to be meshed first before meshing the interior. This method generates extremely high-quality meshes close to the boundaries, and both high-quality isotropic and anisotropic elements can be placed easily. This feature makes it particularly suitable for computational aerodynamics, where the flow structure near the geometry boundary is often of high interest. However, in regions where the front collides, the quality of elements obtained by AFT deteriorate significantly. The problems of choosing the optimal faces to advance, finding the best preexisting vertices as the candidate and detecting illegal intersections in the front are long-lasting issues of AFT [56].

- *Octree refinement-based method* starts to generate the elements with cubes overlaying the mesh-generation region as a background mesh. With a sizing function, the cubes are subdivided recursively until an octree data structure is constructed [78]. The resolution of the background mesh should be small enough comparing to the size of the features in order to describe the geometry with adequate accuracy. Moreover, the resolution jump between two adjacent octants should below a limit to satisfy the balance constraint [56]. For leaf octants that intersect with the geometry boundary, the vertices of the cubes are adjusted to recover the geometry and the intersections are then triangulated. Different techniques can be applied to generate the final volumetric mesh subsequently. The main advantage of this approach is the high efficiency and the simplicity of applying parallelization. It is also easy to adapt to a target feature-size function. The method is widely used for fast geometric searching [79] and image discretization in bioinformatics [80]. On the other hand, the Octree refinement-based method also suffers from several drawbacks, e.g. the worst elements are more likely to appear near the boundaries of the geometry, and the mesh quality is influenced by the directions of the background mesh.

- *Delaunay refinement-based method* is by far the most popular mesh generation method. It starts generating a mesh from a Delaunay triangulation and refines the existing mesh by adding new vertices recursively into the domain while maintaining the Delaunay property of the new elements, i.e. the circumcircle of the element encloses no vertices other than the vertices of the element. The refinement terminates until a prescribed criterion is achieved [64]. A valid mesh can always be constructed by this approach with mathematical guarantees. Since one of the key steps of this method is to insert new vertices into the domain for refinement, the choice of where to put the new vertices is critical to the resulting mesh quality, and various algorithms have been developed to achieve both guaranteed quality and target sizing function [81][82][83][84][85][86]. Unlike AFT, this method produces high-quality meshes in the interior of the domain, whereas, for most variants, the worst elements are concentrated near the boundary.

- *Delaunay variational-based method* can be treated as a standalone mesh generator or combined with other mesh generation methods, e.g. the Delaunay refinement-based method, as a mesh-quality optimizer. This method relies on a predefined target sizing function [66][87] or a target density function [59] and a customized cost function based upon the target function. The mesh generation procedure is performed by seeking a global minimum on the cost function using different numerical approaches, e.g. constructing the Centroidal Voronoi Tessellation (CVT) [59]. By relating the cost function definition with the distortion of the mesh shape and the discrepancy to the desired mesh size, the target of high-quality mesh generation is achieved once a global optimal solution is obtained. It is essentially different from the classical optimization methods, i.e. geometric optimization, topology optimization and vertex insertion or deletion, due to the global optimization process. The main advantage of this method is that significant less slivers are produced comparing to previous approaches, and in some researches, even sliver-free meshes are generated [66][88]. Moreover, it is less sensitive to the initial mesh configuration. Besides the aforementioned advantages, one of the major issue is the performance, owing to the iterative nature of the method, which is significantly slower than the state-of-the-art refinement-based methods, and the parallelization is non-trivial [89][90].

- *Particle-based method* shares high similarities with the Delaunay variational-based method, where the same cost function is required and the mesh generation is also characterized by the cost-minimization process. As demonstrated in various studies [29][5][68][91], high quality meshes can be obtained with particle-based method. Despite the similarities, one fundamental difference comparing to Delaunay variational-based method is whether the connectivity information is required during the optimization procedure. For particle-based methods, pair-wise forces are defined between interacting particles to relax the system towards the target distribution. Therefore, the mesh quality is improved implicitly without constructing a Delaunay tessellation for each optimization iteration. Moreover, since the interaction is constrained locally within a short cutoff radius, only local information is required for each particle. Benefiting from aforementioned unique features, the particle-based method can be easily extended to parallel systems and achieve scalable performance [71]. Moreover, in my recent preprint [77], significant improvement in terms of performance is achieved by the proposed feature-aware SPH formulation, which makes

the underlying methodology more competitive to other mesh-generation approaches.

2.2.2 Parallel methods

Comparing to serial mesh generators, additional constraints need to be overcome in order to achieve high performance in parallel environments. First, on the node level, the parallel algorithm is able to achieve a well-balanced load distribution and an optimized communication volume; Second, the locality of the data is maintained inside each node to ensure a high concurrency; Last, the resulting mesh quality should be consistent when the underlying serial mesher is extended to the parallel environment, and the existing code is reused as much as possible [70][61]. In order to achieve these additional targets, tremendous efforts have been done in the past two decades. A detailed survey can be found here [70].

In the early stage of developing parallel meshers, a direct coarse-grained strategy is employed. The domain is first partitioned, either continuously or discretely, into a set of sub-domains. Then each sub-domain is meshed separately into sub-meshes by applying different sequential mesh-generation kernels. However, merely computing all the sub-meshes does not necessarily accomplish the goal of meshing the entire domain. The interfaces of sub-domains remain an issue if the mesh vertices are not properly synchronized. In order to ensure a consistent mesh in the sub-domain boundaries, additional communications are required during the mesh generation. This overhead due to data communication is critical if the number of nodes become significant. The choice of data synchronization strategy characterizes the parallel mesh generation into tightly-coupled, partially-coupled and decoupled approaches [70].

- *Tightly-coupled method* is the most straightforward implementation, where MPI or multi-threading technique is applied directly to parallelize existing sequential mesh generators in a brute-force manner. The mesh is generated in the interior and on the interface of each individual sub-problem simultaneously. This method can be employed both in shared-memory and distributed-memory systems. The advantage of this method is that the mesh quality is consistent with its original sequential mesher and the result does not depend on a properly defined domain decomposition. However, in order to achieve the consistency, a significant amount of communication overhead and irregular communication patterns is introduced. Meanwhile, the parallelization needs to basically rewrite the code, thus this method features zero code reuse. Successful examples of this method is the Parallel Optimistic Delaunay Mesh (PODM) [92] and Parallel Advancing Front Technique for shared memory computers (PAFT_{SM}) [93].

- *Decoupled method*, on the contrary to the tightly-coupled method, first partition the domain into continuous sub-domains and the interfaces of the sub-domains are meshed prior to generating the volumetric mesh. During the rest of the mesh generation, the meshes of the interfaces remained intact, therefore the data communication is completely eliminated and the meshing of sub-domains is decoupled from the system. Inside each sub-domain, different sequential meshers can be employed with only trivial modifications. This method features low communication overhead and high code reuse, whereas the resulting mesh quality is sensitive to the initial decomposition of the domain. Popular

methods are the Parallel Projective Delaunay Meshing (P^2DM) method [94] and the Parallel Delaunay Domain Decoupling (PD3) method [95].

- *Partially-coupled method*, e.g. Parallel Octree AFT (POAFT) method [93] and Parallel Constrained Delaunay Meshing method (PCDM) [96], finds a compromise between communication reduction/code reuse and stability/consistency comparing to the previous two methods. Instead of completely decoupling each sub-domains, limited communications are allowed in this method to facilitate the mesh quality near the interface, while in the interior regions of the sub-domain, the meshes are generated separately using existing sequential mesh generators. Therefore, the amount of communication is reduced comparing to tightly coupled ones and the codes are more stable in terms of achieving high-quality meshes. However, this method relies on single-pattern message that communicates between neighboring sub-domains, which is difficult to control the resulting mesh quality and a unified solution in three dimension is yet to come [70].

The single-level coarse-grained parallelism generally struggles in finding a balance between controlling the amount of communication on the sub-mesh interfaces and maintaining a consistent output of mesh quality. More recently, since the rapid development of modern manycore architectures, e.g. GPUs, the coarse-grained model is no longer suitable to these hardwares [97]. To exploit the computational horsepower of the newly-emerged high-performance architectures, developing fine-grained parallel mesh generation method is of high demand. Comparing to previous-generation architectures, the manycore machines normally feature a significant increasing number of cores per node and a decreased memory/clock rate per core. Therefore, the fine-grained model needs to achieve a high degree of concurrency and maintain high data locality for lower memory access penalties [98]. This is currently still a new topic, I refer to [99][98] for some of the recent developments.

Despite the efforts in developing fine-grained parallel mesh generators, some researches extend the original single-level parallel algorithm with a hierarchical parallel model, where different parallel meshers are overlaid to achieve a better balance and performance in distributed memory systems. Recently a hybrid two-level Locality-Aware Parallel Delaunay imaging-to-mesh conversion algorithm (LAPD) is developed in [100]. A partially coupled scheme is employed operating at the coarse level, and a tightly coupled method PODM is utilized to optimize mesh quality within each sub-domain. The inter-node communication only happens at the coarse level and high-concurrency is maintained by the tightly coupled approach. More recently, a nested master-worker communication model is proposed in [101] to overlap the communication and computation and to further exploit the two-level parallelism on DSM.

2.2.3 SPH-based mesh generation method

The SPH-based isotropic/anisotropic unstructured mesh generation methods are briefly reviewed together in the section. For detailed methodology and results, I refer to [5][29]. In both researches, only two-dimensional tests are validated. Later the isotropic mesh generator is extended and validated in three-dimensional volumetric and surface meshes [71][77].

Target feature-size function

As introduced in previous sections, the variation-based mesh generation method relies on a target density/feature-size function to optimize the mesh-vertices distribution and the mesh quality. In practice, the target function needs to be defined based on specific problems, which can be geometrical information, local flow details or arbitrary user-defined inputs. In [5], the effects of distance to the geometry surface ϕ, the minimum distance from the geometry singularities ψ and the diffused curvature field in the domain κ are considered,

$$\begin{cases} h_t = f(\phi, \psi, \kappa), \\ \rho_t = g(\phi, \psi, \kappa), \end{cases} \tag{2.25}$$

where ρ_t is referred as target density function and h_t is the target feature-size function. ρ_t can be obtained from h_t following $\rho_t = \frac{1}{h^{N_d}}$, where N_d is the spatial dimension. The three characteristic fields contributing to the feature-size function are calculated by solving three modeling equations, utilizing a multi-resolution Cartesian background mesh [5].

For anisotropic meshes, the target function is further characterized by a Riemannian metric tensor $\mathcal{M}(\mathbf{r})$, which describes the distortion of the space. $\mathcal{M}$ is a symmetric, positive-defined $N_d \times N_d$ matrix. In [29], the anisotropy is defined by introducing a two-dimensional local coordinate system which is based on the normal and tangential direction of the level-set iso-contours. Consequently, h_t along each axis i and ρ_t are modified as

$$\begin{cases} h_{i,t} = f(\phi, \psi, \kappa, \boldsymbol{n}), \\ \rho_t = g(\phi, \psi, \kappa, \boldsymbol{n}), \end{cases} \tag{2.26}$$

where $i = 1, 2$ denotes tangential and normal direction respectively, $\boldsymbol{n}$ is the normal direction of the iso-contour. Moreover, the relationship between ρ_t and $h_{i,t}$ is reformulated as

$$\rho_t = \begin{cases} \dfrac{1}{h_{1,t}} & 1D, \\ \dfrac{1}{h_{1,t} h_{2,t}} & 2D. \end{cases} \tag{2.27}$$

The total number of mesh vertices can be calculated based on ρ_t. First, the total mass inside the geometry and on the geometry surfaces is obtained by a volume integral over Ω

$$M_v = \int_\Omega \rho_t \triangle \, dv, \tag{2.28}$$

where

$$\triangle = \begin{cases} 1, & \text{if inside the geometry,} \\ 0, & \text{otherwise,} \end{cases} \tag{2.29}$$

and a surface integral over $\partial\Omega$

$$M_s = \int_{\partial\Omega} \rho_t ds. \tag{2.30}$$

Then by assuming each particle carries unit mass, the total number of volumetric-mesh vertices and surface-mesh vertices can be obtained.

Governing equations

The mesh quality and mesh-vertex distribution are optimized when the approximation error between the particle discretization and the given target function ρ_t is minimized. According to [66], the L^p norm between the gradient of $\rho_t(\mathbf{r})$ ($\bigtriangledown\rho_t(\mathbf{r})$) and its interpolation ($\bigtriangledown\tilde{\rho}_t(\mathbf{r})$) can be employed to characterize the error since the mesh quality can strongly affect the gradient error. An optimal mesh is obtained when the following criterion is satisfied

- $L(\mathbf{r}) = \int_\Omega \| \bigtriangledown \tilde{\rho}_t(\mathbf{r}) - \bigtriangledown\rho_t(\mathbf{r})\|_{L^p} d\mathbf{r}$ is minimum,

where $L(\mathbf{r})$ is the loss function. The optimization of $L(\mathbf{r})$ can be solved based on a fluid relaxation analogy.

First, by introducing an EOS

$$p = f(\rho), \tag{2.31}$$

where p is the pressure, the target density is linked to a physical pressure. Then the gradient of pressure results in inter-particle forces. The system achieves an equilibrium states when all the particle forces vanish. One can assume

$$p = P_0\Big(\frac{\rho}{\rho_t}\Big)^K, \tag{2.32}$$

where P_0 is a constant pressure and K is a user-defined parameter. When the equilibrium is obtained, $\frac{\rho}{\rho_t}$ becomes a constant, i.e. the target of minimizing $L(\mathbf{r})$ is achieved.

The particles are updated following a Lagrangian form of governing equations for isothermal compressible flows

$$\frac{d\rho}{dt} = -\rho \bigtriangledown \cdot \mathbf{v}, \tag{2.33}$$

$$\frac{d\mathbf{v}}{dt} = -\mathbf{F}_p + \mathbf{F}_v, \tag{2.34}$$

$$\frac{d\mathbf{r}}{dt} = \mathbf{v}, \tag{2.35}$$

where $\mathbf{F}_p$ and $\mathbf{F}_v$ denote the pressure force and the viscous force respectively. The viscous force is introduced to ensure an adequate amount of dissipation of the system.

Numerical discretization

The target information is calculated and stored in a multi-resolution Cartesian background mesh. For each SPH particle i, $\rho_{t,i}$ can be interpolated from the background mesh with its position $\mathbf{r}_i$. The $h_{t,i}$ is calculated following

$$h_t = \tau\Big(\frac{1}{\rho_t}\Big)^{1/N_d}, \tag{2.36}$$

where τ is a scaling factor depending on the kernel function.

The pressure force is discretized in a symmetric form as

$$\mathbf{F}_{p,i} = \sum_j m_j\Big(\frac{p_0}{\rho_{t,i}^2} + \frac{p_0}{\rho_{t,j}^2}\Big) \bigtriangledown W_{ij}. \tag{2.37}$$

The viscous force is calculated following

$$\mathbf{F}_{v,i} = \sum_j m_j \frac{2\eta_i \eta_j}{\eta_i + \eta_j} \Big(\frac{1}{\rho_{t,i}^2} + \frac{1}{\rho_{t,j}^2} \Big) \mathbf{v}_{ij} \bigtriangledown W_{ij} \cdot \mathbf{e}_{ij}, \tag{2.38}$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, $\mathbf{e}_{ij} = \dfrac{\mathbf{r}_{ij}}{|r_{ij}|}$, and $\eta = \rho\nu$ is the dynamic viscosity. ν is given as

$$\nu \sim 0.1 r_c |\mathbf{v}|, \tag{2.39}$$

where r_c is the cut-off radius of particle interaction range, assuming that the local Reynolds number is on the order of $O(10)$. Meanwhile, a simplified friction model is utilized to set an effectively infinite friction coefficient and to damp particle kinetic energy to zero after each time-step.

The system is marched forward in time by a simplified kick-drift-kick scheme [28]

$$\widetilde{\mathbf{v}}_{n+\frac{1}{2}} = \mathbf{v}_n + \frac{1}{2}\mathbf{a}_n \Delta t, \tag{2.40}$$

$$\mathbf{v}_{n+\frac{1}{2}} = \widetilde{\mathbf{v}}_{n+\frac{1}{2}} + \frac{1}{2}\widetilde{\mathbf{a}}_{n+\frac{1}{2}} \Delta t, \tag{2.41}$$

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_{n+\frac{1}{2}} \Delta t. \tag{2.42}$$

The time-step size is calculated by the CFL criterion, the viscous criterion, and the force criterion respectively [5],

$$\Delta t = min\Big(0.25\sqrt{\frac{r_c}{|\mathbf{a}|}}, \frac{1}{40}\frac{r_c}{|\mathbf{v}|}, 0.125\frac{r_c^2}{\nu} \Big), \tag{2.43}$$

where the artificial speed of sound is assumed as $c_s \sim 40|\mathbf{v}|_{max}$.

In anisotropic SPH (ASPH), an additional linear coordinate transformation is required to map the physical space to the normalized position space for kernel calculation. The coordinate transformation is calculated by a tensor $\mathbf{G}$, and the mapping function is defined as $\boldsymbol{\eta} = \mathbf{G}\mathbf{r}$ at position $\mathbf{r}$. In the local coordinate frame, $\mathbf{G}^k$ is diagonal with elements equivalent to the inverse length-scale along the corresponding cardinal direction, i.e.

$$\mathbf{G}^k = \begin{pmatrix} \dfrac{1}{h_1} & 0 \\ 0 & \dfrac{1}{h_2}. \end{pmatrix} \tag{2.44}$$

The transformation matrix $\mathbf{T}_r^{r\to k}$ and $\mathbf{T}_r^{k\to r}$ are defined to characterize the rotational relationship between $\mathbf{G}^k$ and $\mathbf{G}^r$,

$$\mathbf{T}_r^{r\to k} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}, \ \mathbf{T}_r^{k\to r} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \tag{2.45}$$

where the superscript r and k denote the real-physical frame and the local coordinate frame respectively, θ is the angle between two frames. Consequently, the transformation can be expressed as

$$\mathbf{G}^r = \mathbf{T}_r^{k\to r} \mathbf{G}^k \mathbf{T}_r^{r\to k}. \tag{2.46}$$

The smoothing kernel at position $\mathbf{r}$ in ASPH is then calculated following the transformation

$$W(\boldsymbol{\eta}) = W(\mathbf{G}\mathbf{r}). \tag{2.47}$$

The spatial gradient of the kernel function can be evaluated as

$$\bigtriangledown W(\boldsymbol{\eta}) = \frac{\partial W(\mathbf{Gr})}{\partial \mathbf{r}} = \frac{\partial W}{\partial \boldsymbol{\eta}} \frac{\partial \boldsymbol{\eta}}{\partial \mathbf{r}} = \mathbf{G} \frac{\boldsymbol{\eta}}{\eta} \frac{\partial W}{\partial \eta}. \tag{2.48}$$

Geometry recovery

The particle evolution strategy follows an order of increasing dimension to mesh all the features, i.e. particles sitting on geometry surfaces (denoted as surface particles) are first evolved until an equilibrium state; then the interior particles (denoted as positive phase particles) are solved accordingly. Particles representing geometrical singularities (denoted as singularity particles) remain their positions during the simulation. Moreover, in anisotropic mesh generation, a fourth type of particle is defined and referred as ghost surface particles, which are generated only on geometry surfaces and provide support for positive phase particles as boundary condition.

Since particles are updated in a Lagrangian fashion, the preservation of geometry is not guaranteed if surface particles are always updated following the direction of velocity. In order to ensure the geometry recovery, the contribution from positive phase particles are excluded and only the tangential component of the interaction forces from other surface/singularity particles are considered

$$\mathbf{F}_{*,i} = \mathbf{F}_{*,i} - (\mathbf{N}(\mathbf{r}_i) \cdot \mathbf{F}_{*,i}) \mathbf{N}(\mathbf{r}_i), \tag{2.49}$$

where $\mathbf{N}(\mathbf{r}_i)$ is the unit normal vector on the surface. In each iteration, the updated position of surface particles is projected back onto the surface to further enforce the constraint

$$\mathbf{r}_i = \mathbf{r}_i - \phi_i \mathbf{N}(\mathbf{r}_i), \tag{2.50}$$

where ϕ is the signed distance to the surface.

Post-processing

The main difference between the particle-based mesh generation method and the Delaunay variation-based method is that the optimization procedure doesn't rely on an explicit triangulation of the mesh vertices, i.e. triangulation is only a post-processing step. For isotropic mesh generation, a local Voronoi diagram is first constructed for each particle in parallel and the nearest neighbors are found to form a global graph. Then intersecting edges are removed to construct the final mesh. For anisotropic mesh generation, an additional step of coordinate system transformation is required. I refer to [5][29] for more details.

2.3 The level-set method

The level-set method is first proposed in 1988 [102]. It is one of the most popular interface capturing methods, and has achieved a broad success in various areas, e.g. fluid mechanics, image processing, material science, and shape optimization [103]. By embedding the interface Γ in a signed-distance function $\phi(\mathbf{r})$ as a zero contour, the geometry is characterized implicitly. Consequently, it is easy to describe complex geometries as a level-set function.

2.3.1 The level-set function

In level-set method, the interface Γ is characterized by the $\phi = 0$ contour of the signed-distance function $\phi(\mathbf{r})$, i.e.

$$\Gamma = \{\mathbf{r} \in \mathbb{R}^d | \phi(\mathbf{r}, t) = 0\}, \tag{2.51}$$

where d is the spatial dimension and t is the time. With Γ, the domain Ω is partitioned into the positive region $\Omega^+ = \{\mathbf{r} \in \mathbb{R}^d | \phi(\mathbf{r}, t) > 0\}$ and the negative region $\Omega^- = \{\mathbf{r} \in \mathbb{R}^d | \phi(\mathbf{r}, t) < 0\}$.

2.3.2 Geometry information calculation

Based on the level-set function, the geometric information on the interface, e.g. normal direction and curvature, can be calculated. Assuming the level-set function is defined in a uniform Cartesian grid, the normal direction of the interface can be calculated by

$$\mathbf{n} = \frac{\bigtriangledown \phi}{|\bigtriangledown \phi|}. \tag{2.52}$$

$\bigtriangledown \phi$ can be approximated using a finite difference method, e.g. the derivative $\left(\frac{\partial \phi}{\partial x}\right)_i$ at point i can be computed by

$$\left(\frac{\partial \phi}{\partial x}\right)_i = \frac{\phi_{i+1} - \phi_{i-1}}{2\triangle x}, \tag{2.53}$$

where $\triangle x$ is the spatial resolution.

The mean curvature κ can be computed by

$$\kappa = \bigtriangledown \cdot \left(\frac{\bigtriangledown \phi}{|\bigtriangledown \phi|}\right), \tag{2.54}$$

and the discretized version is

$$\kappa = \frac{\phi_{xx} + \phi_{yy} + \phi_{zz}}{(\phi_x^2 + \phi_y^2 + \phi_z^2 + \varepsilon)^{1/2}} - \frac{\phi_x^2 \phi_{xx} + \phi_y^2 \phi_{yy} + \phi_z^2 \phi_{zz} + 2(\phi_x \phi_y \phi_{xy} + \phi_x \phi_z \phi_{xz} + \phi_y \phi_z \phi_{yz})}{(\phi_x^2 + \phi_y^2 + \phi_z^2 + \varepsilon)^{3/2}}, \tag{2.55}$$

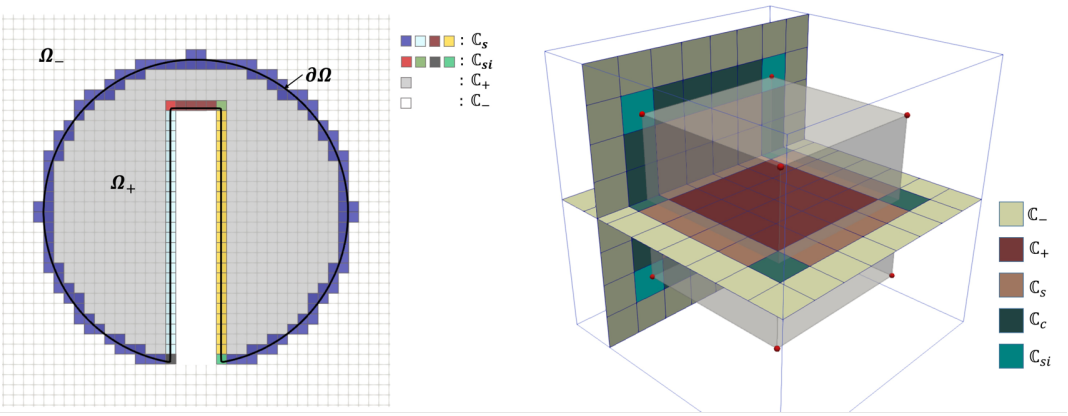
where ε is a small number to avoid division by zero [104].

2.3.3 Geometry definition for mesh generation

For the SPH-based mesh generation method introduced in Section 2.2.3 and Appendix A.3, the level-set function is used for the geometry definition. A multi-resolution Cartesian background grid is used to discretize the level-set function. In order to ensure the interpolation accuracy, the resolution of the background grid is chosen to be 1.5 to 2 time finner than the minimum feature-size of the mesh.

As described before, in level-set method the interface is defined implicitly by a zero contour in the signed-distance function. For complex geometries, multiple sharp edges and singularity points may exist, which cannot be resolved by a single level-set function. Therefore, in order to characterize the under-resolved features, additional inputs are required to incorporate with the level-set function. For singularity points, the positions can be directly imported, see Fig. 2.3 (a) for a 2D example. Sharp edges can be described by piecewise-linear B-splines. Fig. 2.3 (a) is an example of 3D cube, where multiple sharp edges and singularity points are presented.

Moreover, to better distinguish the geometry information defined by explicit inputs and the level-set function, a cell-tag system can be utilized, where a specific tag is assigned to each cell according to the type of feature that intersects with the cell or contains the cell. In Fig. 2.3, each tag is represented by a different color. For more information, I refer to the recent publication and preprint [71][77].



(a) (b)

FIGURE 2.3: Cell tags incorporating with the level-set function for the geometry definition of (a) a 2D Zalesak's disk and (b) a 3D cube.

Chapter 3

Large-scale parallel computing

In this chapter, technical details and challenges regarding the large-scale parallel computing for SPH and particle-based methods are presented. Three fundamental components, i.e. domain decomposition and dynamic load balancing, data communication and the PFNS, are discussed.

3.1 Domain decomposition and dynamic load balancing

The topics of domain decomposition and dynamic load balancing are key components in modern scientific computing. Due to the ever increasing demand for simulating larger systems and the adaptive nature of the underlying physical problems, various discretization techniques, e.g. unstructured or adaptive structured mesh, are employed to achieve better balance between computational accuracy and efficiency. However, this results in a unequal distribution of the computational load. The issue of equally assigning the load to each computer node becomes a scientific problem. Moreover, in heterogeneous clusters, the computational capacity is also not uniform for all the nodes. The domain decomposition method needs to be aware of the infrastructure and should be able to adjust the load-balance criterion accordingly. This issue becomes more critical for some of the latest clusters that feature both GPU and CPU architectures, where task-based load balancing is required to better harness all the available computational resources. The key functionality of the domain decomposition method is to accomplish approximately equal-sized partitioning diagram with minimum neighbor communication patterns [28].

Comparing to static partitioning, achieving dynamic load balancing of the system is equally important for problems involving large variations of mesh topology or particle configuration [105], where computational load needs to be reassigned equally to all the processors periodically. In each repartitioning, the same targets need to be accomplished as in the static domain decomposition, i.e. load balance, locality, minimizing inter-processor communication. In addition, the amount of data migration needs to be controlled too to achieve lower overhead due to the rebalancing procedure. In order to resolve the additional constraint, it is critical for the repartitioning algorithm to achieve a higher incremental property, i.e. minimizing partitioning modifications subject to small topology changes [106]. I refer to [107] and [105] for a comprehensive review regarding recent development of repartitioning approaches.

3.1.1 Target description

For a given computational domain Ω and a point set V, the domain can be discretized by connecting the point set to construct a mesh (mesh-based methods) or directly applying physical attributes to each point (particle-based methods). Each

element in the discretized domain can be treated as a computational unit. Combining all computational unit forms a partitioning of the domain, which can be characterized as a graph $G = (V, E)$, where E denotes the communication relationship between computational units.

In parallel simulation, V is divided into n disjoint subsets denoted as V_1, V_2,..., V_n respectively, and each subset is associated with one MPI task. The mathematical definition of the targets in domain decomposition is [28]:

- $V_1 \cup V_2 ... \cup V_n = V$ and $V_i \cap V_j = \emptyset$ with $i \neq j$;
- $|V_i| \approx \lceil \frac{|V|}{n} \rceil, i = 1, 2, ..., n$;
- $\sum_{i<j} E_{ij}$ is minimum, where $E_{ij} = \{\{u, v\} \in E | u \in V_i \wedge v \in V_j\}$.

Note that only equal mass partitioning is considered in this definition.

As described before, the incremental property is the additional target for dynamic load balancing. The target can be written as

- $\mathbb{E} = \frac{1}{n} \sum_i^n |V_i^{t+} - V_i^{t-}|$ is minimum,

where $t-$ and $t+$ denote before and after the system is rebalanced at time t.

3.1.2 Geometry-based and graph-based methods

Classical domain decomposition method can be roughly categorized as the geometry-based and graph-based method [108].

- *Geometry-based method* uses the physical coordinates of mesh vertices/particles as input and partitions the domain according to the geometrical adjacency of the coordinates with a specific encoding. The Recursive Coordinate Bisection (RCB) method [109] partitions the domain recursively with planes orthogonal to the coordinate axes until the number of sub-regions equals the number of processors. Later the method is extend to the Recursive Inertial Bisection (RIB) method [110], where the principle inertia axises are used to define the cutting planes. RIB improves the performance when the computational load is skew-aligned. The main advantage of the two methods is the high efficiency and simplicity. Another popular geometry-based method uses Space Filling Curves (SFC) [111][108] to map higher-dimension grid into an one-dimensional ordered sequence. The mapping function is designed following a specific order that improves the data locality. Then the one-dimensional sequence is partitioned into desired number of pieces to achieve the goal of domain decomposition. The major disadvantage of the geometry-based method is that the target of minimizing communication volume cannot be achieved without sacrificing the load balance. Despite the disadvantage, the method is widely used in methods where explicit connectivity information is not available.
- *Graph-based method* partitions the domain based on both the mesh elements and the connectivity information. The targets of load balancing and communication optimization can be achieved by Recursive Spectral Bisection (RSB) [112] or a multilevel graph partitioning strategy [113]. Later the concept of Hypergraph partitioning is introduced [114] to resolve the issue where the optimization objective function is not always adequate [28]. For the dynamic load balancing, the graph-based diffusive partitioners are developed to improve the

incremental property [115][116]. The advantage of graph-based method over geometry-based method is obvious, i.e. all the targets in domain decomposition are handled more consistently. The main drawbacks are that strictly-connected partitions cannot be guaranteed in principle [116] and it is computationally much more expensive than the geometry-based method.

3.1.3 The CVP method

Recently, in addition to the above mentioned methods, the CVP [4] method is proposed by combining the concepts of CVT [117] and Voronoi Particle dynamics (VP). As a geometry-based method, CVP optimizes the communication volume implicitly due to the construction of CVT, which guarantees high-level compactness of partitioning sub-domains. Although the communication optimization cannot be proved explicitly, various researches [116][4][28] have demonstrated a direct relation between communication minimization and compactness of the sub-domain.

Moreover, with the VP method, the target of error-controlled load balancing can be achieved, which is difficult for CVT with a small number of Voronoi generators [4]. With an arbitrary density field $\rho^{vp}(\mathbf{r})$ representing the computational load distribution, the total mass m^{vp}, i.e. computational load, in the computational domain Ω can be calculated by $m^{vp} = \int_\Omega \rho^{vp} d\sigma$. For equal-sized partitioning problem, the target mass of Voronoi cell i can be obtained by $m^{vp}_{tg,i} = \frac{m^{vp}}{k}$, where k is the number of partitioning sub-domains. The difference between m^{vp}_i and $m^{vp}_{tg,i}$ can be defined as a pressure, i.e.

$$p^{vp}_i = \frac{m^{vp}_i}{m^{vp}_{tg,i}} = k\frac{\int_{\Omega_i} \rho^{vp} d\sigma}{\int_\Omega \rho^{vp} d\sigma}. \tag{3.1}$$

Then by calculating the forces due to the pressure difference of adjacent cells, the Voronoi particles can be modeled by a fluid relaxation procedure using a Lagrangian governing equation

$$\frac{d\mathbf{v}^{vp}_i}{dt} = -\frac{\int_{\Omega_i} \bigtriangledown p^{vp} d\sigma}{\int_{\Omega_i} \rho^{vp} d\sigma} = -\frac{\int_{\partial\Omega_i} p^{vp} d\mathbf{S}}{m^{vp}_i}, \tag{3.2}$$

where $\mathbf{v}^{vp}$ denotes the velocity vector and $\partial\Omega_i$ the surface of Voronoi cell i. When an equilibrium state is achieved, the target of load balancing is achieved since all forces vanish.

To combine the construction of CVT and the evolution of VP, a two-step time integration scheme is employed. In each iteration substep the Voronoi particles are first moved according to the VP method by

$$\mathbf{r}^{vp,*}_i = \mathbf{r}^{vp,n}_i + \alpha\frac{1}{2}\mathbf{a}^{vp,n}_i\tau_1^2, \tag{3.3}$$

and then updated following CVT construction using the Lloyd method [118][119]

$$\mathbf{r}^{vp,n+1}_i = \mathbf{r}^{vp,*}_i + (1-\alpha)\tau_2(\mathbf{z}^{vp,n}_i - \mathbf{r}^{vp,*}_i), \tag{3.4}$$

where τ_1 and τ_2 are pseudo timestep sizes and $\mathbf{z}^{vp}_i$ is the center of mass of Voronoi cell i. The relaxation parameter α is set as 0.8.

3.1.4 The SPH-based domain decomposition method

A SPH-based domain-decomposition method [28] has been developed too following the same fluid-relaxation methodology as the CVP. A set of colored SPH particles are introduced and each represents a subset of the total computational units. The number of colors equals to the total number of sub-domains. The target of load balancing can be achieved by controlling the total mass, i.e. computational load, within each color. The same governing equation (Eq. 3.2) is discretized using SPH to evolve the particle system to a steady state. Moreover, to optimize the communication volume, an inter-domain surface-tension force is introduced

$$\mathbf{F}_{s,i} = -\sum_j \beta m_j \left(\frac{p_0}{\rho_i^2} + \frac{p_0}{\rho_j^2} \right) \frac{\partial W(r_{ij}, h_{ij})}{\partial r_{ij}} \mathbf{e}_{ij}, \tag{3.5}$$

where β is a coefficient characterizing the strength of surface-tension effect. β is defined as

$$\beta = \begin{cases} 0, & \text{if } C_i = C_j, \\ 3, & \text{overwise}, \end{cases} \tag{3.6}$$

where C_k is the color function for particle k. Due to the surface tension force, particles carrying different colors are separated and form a sharp interface. Moreover, particles carrying the same color tend to concentrate and the interface of neighboring sub-domains is regularized towards a convex and compact shape. As a consequence, the communication volume is optimized.

When an equilibrium state is achieved, the resulting partitioning diagram features highly regularized shapes of sub-domains and each sub-domain encloses approximately the same computational load represented by SPH particles. With a smoothing kernel, the colors of each computational element in the target mesh can be calculated, i.e. the domain decomposition is finished. For more details, I refer to [28].

3.2 Parallel fast neighbor searching

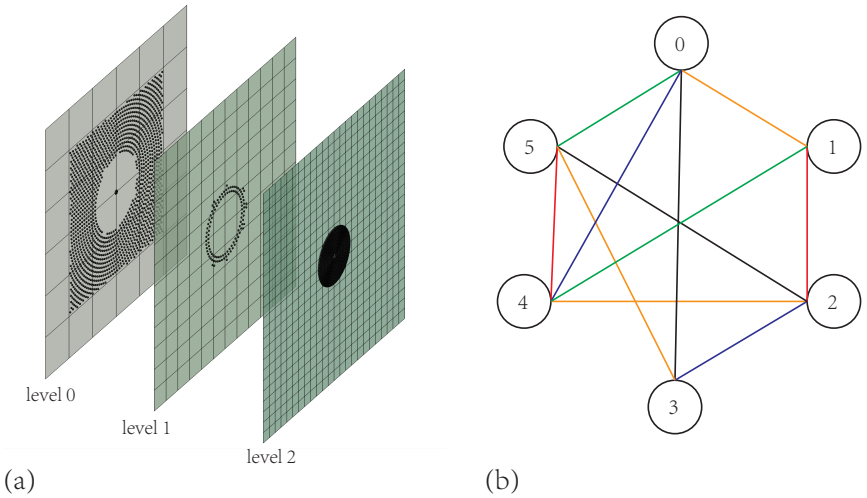


FIGURE 3.1: (a) Illustration of the nested hierarchical data structure with three levels of refinement. (b) Sketch of a communication graph, and the result of applying the edge coloring algorithm.

In SPH, the field quantities and their derivatives are approximated by integrating the contributions from neighboring particles within a certain cutoff range of the current particle. Since the particles generally feature irregular distributions, it is difficult to find the neighboring candidates efficiently. A direct search over all particles (N) results in a computational cost on the order of $\mathcal{O}(N^2)$, which is inapplicable in large-scale parallel computing. More applicable approach is to utilize a background grid that overlays the computational domain, and to register particles inside the cells according to the physical coordinates. Then the neighbors can be found by a search over only neighboring cells, which is easy to obtain from a grid.

Popular data structures for NNS are the CLL [120], the Verlet list (VL) [121] and the hierarchical tree data structure [122]. The CLL divides the domain recursively with uniform cells of size equal to the particle cutoff range. Since the grid is uniform, it is easy to map particles into the corresponding cells. By traversing the closest adjacent cells, the neighboring particles can be located easily. Unlike the the CLL, where neighbors are calculated on-the-fly, the VL stores the neighbors explicitly for each particle. Then the cost for NNS is cheaper comparing to CLL if the information is required multiple times during one timestep. Moreover, one can also inflate the cutoff by a "skin distance" to enclose more potential neighbors [121]. Consequently, the construction of VL can be avoided every timestep. However, the performance of VL highly depends on the construction method [123] and the demand from numerical solvers [124]. The advantage of using CLL or VL is that they can be built in $\mathcal{O}(N)$ and queried in $\mathcal{O}(1)$, whereas the handling of adaptive particle resolution is difficult. For simulations involving variable particle scales, e.g. cosmological simulations in computational astrophysics [125][126], the tree-based structures are mostly adopted, which can be built in $\mathcal{O}(NlogN)$ and queried in $\mathcal{O}(logN)$. The drawback of this data structure is that the tree needs to be maintained dynamically if the topology changes. The efficiency drops heavily when the system features large variations of the computational load, which is typically the case in SPH. Moreover, to find the neighbors, a tree-walk is required on-the-fly, which is less efficient comparing to CLL and VL.

3.2.1 The multi-level nested hierarchical data structure

In [28], a nested hierarchical data structure is proposed based on the idea of CLL, where multiple layers of CLL with different grid sizes are built to characterize particles into different levels. The scale ratio S between two consecutive levels is fixed. The number of levels is defined as

$$N_{level} = log_S \left(\frac{r_{c,max}}{r_{c,min}} \right) + 1, \tag{3.7}$$

where $r_{c,max}$ and $r_{c,min}$ are the maximum and minimum scale of SPH particles. In level l, identical to the procedure in CLL, the domain is recursively divided in each dimension with cell size $r_{c,l} = r_{c,min} \times S^{l_{max}-l}$, where l_{max} is the finest level. For two consecutive level l and $l+1$, the inheritance can be easily calculated. E.g. the $cell(\mathbb{J}, l+1)$ is defined as the child of $cell\left(\lfloor \frac{\mathbb{J}}{S} \rfloor, l\right)$, where $\mathbb{J} = (j_1, j_2, j_3)$ is the index of the cell.

A particle with cutoff $r_{c,i}$ is mapped into level l if

$$r_{c,i} \in (\frac{r_{c,max}}{S^{l+1}}, \frac{r_{c,max}}{S^l}]. \tag{3.8}$$

Then on level l, the particle can be mapped to cell

$$\mathbb{J}_i = [\frac{\mathbf{r}_i}{r_{c,l}}], \tag{3.9}$$

where $\mathbf{r}_i$ is the position of particle i.

Fig. 3.1 (a) shows the data structure in two-dimension with three levels of refinement.

3.2.2 The tag system for PFNS

For NNS within a single level, the same operations as the CLL are performed. To enable the cross-level PFNS, a tag system consisting of several identifiers is constructed [28][1] to mark the status of each cell. The operations can be concluded into two phases. In phase 1, by performing a top-to-bottom and a bottom-to-top traversal of the data structure, the cells that can influence and can be influenced by particles inside the sub-domain are marked. Then the tag systems between two neighboring sub-domains are compared. Cells in remote that can influence the current subdomain are registered as ghost cells. All the particles registered in the ghost cells are sent to the target sub-domain during communication. In phase 2, a bottom-to-top traverse of the data structure is performed to register particle address pointers from finer levels to coarser levels. Since all the potential missing candidates are mapped to the corresponding level with the above operations, one only need to traverse adjacent cells in the current level to find all the neighbors, thus any cross-level search is unnecessary. The traversal of the data structure can be carried out concurrently and in parallel.

3.3 Data communication

In large-scale parallel computing involving sparse data communication pattern, the communication strategy is critical to the parallel efficiency [127][1]. In this section, the graph-based communication strategy and communication frequency optimization for multi-resolution particle-based methods developed in [1] is introduced.

3.3.1 Graph-based communication strategy

Due to limited wave propagation speed in weakly-compressible and compressible fluids, particles only interact with limited amount of neighbors in each timestep. In parallel environment, this characterizes a sparse data communication topology. According to [1], the communication topology can be defined by an undirected simple graph $G_{cm} = (V_{cm}, E_{cm})$, where V_{cm}, E_{cm} denote vertex and edge respectively. Fig. 3.1 (b) shows the sketch of a communication graph. In this graph, each vertex represents a sub-domain, and each edge connects a pair of interacting sub-domains.

To construct the communication graph, the same data structure as introduced in Section 3.2.1 is utilized. According to the SPH particles that reside inside the cell, each cell is assigned with a color-list. Then by performing an “upward projection evaluation" operation in the data structure, the communication topology can be obtained by comparing the overlapping information between each colored region.

3.3.2 Communication frequency optimization

Although the communication topology is built, the problems of potential data racing condition and communication deadlock are still not resolved if an optimized communication sequence is not provided. One solution is to allocate additional buffer memory space to temporarily store the data coming from neighboring subdomains, and merge the collected data afterwards. However, this approach creates additional overhead of memory consumption and operations. Another means is to define a sequential execution order of communication which avoids the data traffic and improves the efficiency. In [1][128], the edge-coloring method is used for the communication frequency optimization. As illustrated in Fig. 3.1 (b), the edge-coring algorithm colors all edges and ensure that no edges sharing the same vertex possess identical color [129]. The communication frequency can then be explicitly characterized by the number of colors. In each communication sub-iteration, only the communication pair possessing the same color is involved. Consequently, the conflicts of data racing condition is eliminated.

Chapter 4

Summaries of publications

In this chapter, the selected publications and main accomplishments during my PhD stage are briefly summarized. The current state-of-the-arts regarding each topic are presented too in each section. For more details, I refer to Appendix A.1, A.2 and A.3 respectively.

4.1 A new multi-resolution parallel framework for SPH

Z. Ji, L. Fu, X. Y. Hu and N.A. Adams (2019)

4.1.1 State of the art

The development for parallel SPH code that supports adaptive smoothing-length originated from researchers in computational astrophysics. Due to the Lagrangian nature of SPH, it is particularly suitable for cosmological simulations where particles can follow the gravitational growth of the structure and the resolution is increased automatically in the central regions of galactic halos [12]. In addition, SPH also satisfies Galilean invariance and conserves mass inherently.

The most popular implementation of SPH in computational astrophysics utilizes the tree data structure, where the domain is divided recursively into a hierarchical representation so that all leaves are either empty or contain a small number of particles. A tree-walk operation is performed to find neighboring particles that fall into the cutoff radius. In 1989, the TreeSPH code is developed [125], and it is successfully applied to various cosmological simulations. Later, the GADGET-2 [12] code is developed based on the main structure of TreeSPH and with massive parallelization capability. In GADGET-2, the domain decomposition is handled based on the Peano–Hilbert curve. A collective hypercube communication strategy is developed to hide communication latency. The code is paralleled by both MPI and OpenMP techniques. It is by far perhaps the most well-established SPH code in computational astrophysics due to ability to scale in a large number of nodes. Other high-performance codes, e.g. Phantom [130], SPHYNX [131], Gasoline/Gasoline2 [126][132] and etc., have also been published in the past decade. Similar structures and algorithms are employed in terms of constructing the parallel environment. More recently, due to the wide application of GPU, there also exist some efforts to extend the tree-based SPH method to GPU with CUDA [133].

Besides the astrophysics community, several other open-source frameworks for different particle-based methods exist too. DualSPHysics [134], designed for free-surface flows in hydrodynamics, is parallelized with OpenMP and CUDA (the publicly available version). LAMMPS [73], maintained and distributed by Sandia National Laboratories and Temple University, is specialized in Molecular Dynamics

(MD) and Dissipative Particle Dynamics (DPD) simulations. Various parallel techniques, e.g. MPI, OpenMP and CUDA are available in the code. For codes that can handle both particle and particle-mesh methods, POOMA [135], PPM [128] and most recently OpenFPM [74] are available with scalable parallel performance. However, the aforementioned codes mainly deal with particles of constant size, and the support for variable resolution is either limited or yet to come.

4.1.2 Summary of the publication

In paper I [72], a new multi-resolution parallel framework for SPH is developed, employing several algorithms from previous work [4][1][28]. The objective is to seek a succinct and unified solution to address some of the existing challenges in the parallel computing of SPH, i.e. domain decomposition, communication optimization and PFNS.

The CVP method [4] is adopted as the domain decomposition method. Comparing to classical geometry-based domain decomposition method, CVP is able to achieve the targets of load balancing while optimize the communication volume simultaneously. Comparing to graph-based method, the parallelization of CVP is straightforward, which is critical for a large number of computational nodes. In [4], CVP is only validated as a static load balancer. In paper I [72], an adaptive criterion and a monitoring system is developed to further integrate CVP as a rebalancer. The adaptive criterion utilizes a time-weighting strategy to dynamically calculate the computational load of each SPH particle. The resulting partitioning features better load-balancing characteristics. Moreover, the proposed monitoring system considers the load imbalance due to the change from both computational and communicational time. Thus the imbalance of the system is captured more accurately. Numerical experiments show that with the proposed algorithm, the target of dynamic load balancing is achieved.

A localized hierarchical CLL-based data structure is developed for the PFNS. Other than using the tree-based data structure, the multi-level CLL-based data structure developed in [1] is tailored into the current framework to achieve higher concurrency and to reduce the cost due to dynamic maintenance. Comparing to the tree data structure, which can be built in $\mathcal{O}(NlogN)$ and queried in $\mathcal{O}(logN)$, the single-level CLL can be built in $\mathcal{O}(N)$ and queried in $\mathcal{O}(1)$. By extending to a hierarchical representation combining with an inter-level mapping of particle pointers, the data structure can be built and updated concurrently, which is well-suited for multi-core CPU and GPU. In paper I [72], the data structure is further modified to incorporate with the framework. First, the data structure is built with only sub-domain information to eliminate the memory bottleneck. Moreover, a safeguard skin area is created to avoid the construction of local data structure every time-step. The performance of PFNS using the local data structure is improved additionally by reducing the number of tag comparison between two neighboring sub-domains.

An optimized communication strategy based on the concept of diffused graph is proposed. Previously in [1] the graph-based communication strategy is developed to handle the sparse data communication among neighboring nodes [127][136]. The graph is originally constructed relying on a global master-slave approach. However, the graph construction does not scale with the partitioning number. Therefore, the runtime contributing to graph construction becomes increasingly significant when the number of partitions grows. In paper I [72], to overcome the serialization bottleneck of graph-construction, the concept of "diffused graph" is proposed. A diffusing distance to the cut-off range of each color is introduced to tune the "density"

of the communication matrix. Consequently, each sub-domain can anticipate potential neighbors within a certain time period in the future due to the limited wave propagation speed constraint. The graph construction at every time-step is avoided. Numerical experiments demonstrate that the time contribution due to graph construction is no longer an issue.

Lastly, the scalability of the framework is measured for both uniform and non-uniform particle distributions. In both cases, the code exhibits good scalability for strong and weak scaling. The weak scaling reveals that the code scales well up to at least 3584 cores. For strong scaling, a good scaling number is achieved for maximum 7168 cores tested.

4.1.3 Individual contributions of the candidate

This article [72] was published in the international peer-reviewed journal *Computer methods in applied mechanics and engineering*. My contribution to this work was the development of the method and the code. I have performed numerical validations, analyzed the results, part of the performance tests, and wrote the manuscript.

4.2 A Lagrangian Inertial Centroidal Voronoi Particle method for dynamic load balancing in particle-based simulations

Z. Ji, L. Fu, X. Y. Hu and N.A. Adams (2019)

4.2.1 State of the art

In simulations featuring large computation-load variation and high anisotropic load distribution in space, e.g. the dambreak problem [137], tsunami simulation [138], and the rotating disk problem in astrophysics [14], significant communication overhead and load imbalance may be introduced to deteriorate the performance of the underlying numerical solver. The issue of load rebalancing becomes critical in the aforementioned scenarios. Comparing to static partitioning, rebalancing needs to meet the same targets, e.g. load balance, locality, optimization for inter-processor communication, but also is subject to further constrains. The key aspect for rebalancing schemes is to minimize partitioning modifications subject to small topology changes, i.e. the incremental property, and to optimize the inter-processor communication at smallest cost [106] [28].

To achieve better incremental property, the load balancer should be aware of load variation in time for the underlying system, instead of taking the load distribution as a new target before every rebalancing. I.e. the repartitioning of the system should be based on the existing partition to achieve lower data redistribution cost [139]. To cure the issue of skew-aligned load distribution, the rebalancer should be able to characterize the anisotropy and dynamically adapt the load-balancing strategy.

Previously, a CVP domain decomposition method [4] has been developed to combine the concepts of CVT and VP. Later, CVP is adopted in paper I [72] as the load balancer for the new multi-resolution parallel framework [72]. An adaptive rebalancing criterion and monitoring system has been proposed to assess the imbalance during simulation and to reassign equivalent load among all processors. Numerical experiments demonstrate that highly compact sub-domains are calculated by CVP and the target of load balancing is achieved with the adaptive rebalancing criterion. Although the CVP method inherently features load balance and communication reduction, there is no explicit treatment in the previous work to handle the additional difficulties encountered in dynamic partitioning.

4.2.2 Summary of the publication

In paper II [140], a Lagrangian Inertial CVP (LICVP) method is developed to extend the CVP method and to improve the performance of particle-based methods in dynamic partitioning problems.

First, a three-step time integration scheme is proposed by introducing a background velocity $\widetilde{\mathbf{v}}_i^{vp}$ to transport Voronoi particles according to local flow details. Originally in CVP, Voronoi particles remain static between two consecutive rebalancing steps, which is unsynchronized with the underlying particle system. Since particles are updated in a Lagrangian fashion, a significant amount of particles may travel in and out of the original Voronoi cell. Consequently, the load variation is not taken into account in the repartitioning of the system. In paper II [140], by introducing $\widetilde{\mathbf{v}}_i^{vp}$, the positions of Voronoi particles are updated following the evolution of the dynamic system and geometry variation of local sub-domains. When the rebalancing subroutine is triggered, the updated Voronoi particle positions are utilized as input for the new partitioning result. Consequently, the incremental property is

preserved. Numerical experiments demonstrate that with the proposed background velocity, the incremental property is improved remarkably comparing to original CVP.

Second, an Inertial CVP is proposed to handle problems featuring skew-aligned computational load and large void regions. By introducing the inertial matrix, the load distribution is characterized by the direction of principle inertia. Based on the load distribution, the motion of the Voronoi particles is constraint on directions where the load distribution is not dominant. The choice of constraint is calculated by a splitting operator C_p. In order to account for the temporal variation, an adaptive filter is proposed to select optimal constraint dynamically according to the development of the underlying particle system. Numerical experiment of a dambreak case demonstrates that with the proposed Inertial CVP method, a speedup of about 6x is achieved comparing to original CVP.

4.2.3 Individual contributions of the candidate

This article [140] was published in the international peer-reviewed journal *Computer physics communications*. My contribution to this work was the development of the method and the code. I have performed numerical validations, analyzed the results, and wrote the manuscript.

4.3 A consistent parallel isotropic unstructured mesh generation method based on multi-phase SPH

Z. Ji, L. Fu, X. Y. Hu and N.A. Adams (2020)

4.3.1 State of the art

Due to the rapidly growing capabilities and capacities of modern supercomputers, numerical methods become more viable for simulating systems consisting up to $\sim 10^{11}$ degrees of freedom. However, the development of parallel mesh generation method is generally lagging behind comparing to the numerical solver that relies on the generated mesh. The topic of developing scalable, stable and high-quality parallel mesh generation methods is still relatively new. Comparing to sequential mesh generator, additional targets arise when the mesh is generated in a distributed memory system. In the previous Section 2.2, a detailed review on the recent progress in unstructured mesh generation method is presented.

To fully harness the computational power of modern DSM architectures, the parallel mesh generator should achieve 1) high concurrency and good data locality in local shared memory system and 2) well-balanced load distribution and low communication cost with remote nodes. Moreover, the choice of domain decomposition should not affect the final mesh quality, which requires the consistency of the underlying method. In this sense, an algorithm that features both coarse-grained and fine-grained parallelism is preferable. Recently, several researches have been published following this direction [100][101] to explore a two-level parallelism.

Comparing to classical mesh generation methods, i.e. AFT, Octree refinement-based method, Delaunay refinement-based method and Delaunay variational-based method, particle-based method naturally satisfies the two-level parallelism, thus making it well-suitable for parallel mesh generation. However, to the best of my knowledge, the topic of parallel particle-based mesh generator in a DSM has not yet been explored.

4.3.2 Summary of the publication

In paper III [71], a consistent parallel mesh generation method with a multi-phase SPH formulation is proposed. With the proposed method, the objectives of partitioning the domain, optimizing communication volume and improving mesh quality are achieved consistently by solving the same set of physics-motivated governing equations.

To combine the objectives of both domain decomposition and mesh generation, a unified target density function $\rho_t = \Phi(\mathbf{r})$ is first introduced, which characterizes the target feature-size of mesh element and provides the distribution of the computational load in the meantime. The target density function can be any smooth scalar field considering various geometrical features and user-defined inputs. By integral over the mesh-generation region, the total mass for mesh generation is obtained and consequently the number of mesh-vertices can be derived. Moreover, with the same integration, the total computational load is obtained too. As a consequence, the target for each sub-domain can be calculated provided the number of MPI tasks.

The key idea of paper III [71] is to merge a parallelization strategy into a set of physics motivated governing equations to relax Lagrangian particles following the target function. A color function is assigned to SPH particles and each color possesses an equal amount of particles, which is obtained from the target computational

load of the corresponding sub-domain. The load-balancing condition is inherently ensured. To handle the additional target of optimizing communication volume, a surface tension model is introduced in addition to the previous work [5]. The mesh generation procedure can be characterized into two stages. During the first stage, the mesh quality is improved in the interior region of each sub-domain. In the meantime, due to the presence of surface tension force, a sharp-interface constraint is preserved, i.e. the communication volume is optimized. Once a steady state is achieved, the second stage starts. The mesh quality near the interface region is optimized by gradually alleviating the surface tension force.

The physics-motivated equations are solved by a multi-phase SPH formulation. The original mesh generator is extended to generating three-dimensional tetrahedral meshes. To better define 3D geometry with sharp edges and to enhance the geometry recovery, piecewise-linear B-splines are supported. The CVP method is utilized to improve the locality property of initial particle seeding.

Numerical results from intensive tests consisting various scales and complexities demonstrate that the calculated sub-domains feature compact shape and high-quality mesh is generated. The communication overhead caused by the optimization of mesh quality near the interface is limited even in cases with complex geometry and large spacial adaptivity. Meanwhile, with the proposed parallel mesh generation method, high quality triangle/tetrahedron mesh can be generated without the need of constructing Delaunay Triangulation/Tetrahedralization explicitly. Since only local information are required during the simulation and the same set of governing equations are solved for all the particles, the proposed method features high consistency and code reusability. Benefiting from the scalable parallel environment designed previously in [72], the mesh generation procedure is able to exploit both fine-grained and coarse-grained parallelization.

4.3.3 Individual contributions of the candidate

This article [71] was published in the international peer-reviewed journal *Computer methods in applied mechanics and engineering*. My contribution to this work was the development of the method and the code. I have performed numerical validations, analyzed the results, the performance tests, and wrote the manuscript.

Chapter 5

Conclusions and outlooks

5.1 Conclusions

SPH, initially as an alternative to the grid-based method, has shown tremendous potentials in the past decade, with significant amount of improvements been made in both numerics and physical capabilities. Greater acceptance of SPH in the general CFD community has been achieved and increasing number of industrial appearances has demonstrated its uniquenesss as a powerfull tool for resolving modern CFD and multi-physics problems. However, the non-negligible higher computational costs comparing to grid-based methods and the lack of efficient algorithms handling adaptive resolution are still issues hinding its acceptance by broader audience.

In this work, several novel numerical methods and algorithms are developed regarding SPH and its HPC. A new and scalable multi-resolution parallel framework for large-scale SPH simulations is develoed to go beyond the limitation of constant resolution in modern clusters with DSM system. The main advantage of the newly developed framework lies in its simplicity and being able to achieve high concurrency in modern multi-core architectures. To improve the incremental property and the performance in problems featuring huge anisotropic load distribution, a Lagrangian Inertial CVP method is proposed to improve the original CVP method in the context of dynamic load balancing. Lastly, a novel consistent parallel mesh generation method is developed to achieve the target of domain decomposition and high-quality mesh generation simultaneously, which is, to the best of the author's knowledge, the first of its kind.

The parallel framework employs several novel methods developed recently in AER. The CVP method [4] is integrated as the domain decomposition method. The CLL-based multi-resolution data structure is tailored for the PFNS, and the graph-based communication strategy is used to characterize the communication matrix and to optimize the communication frequency. The main contribution of paper I [72] is that a set of techniques are proposed to incorporate various functionalities into the framework and to overcome the bottlenecks of efficiency and memory simultaneously. More specifically: (1) An adaptive rebalancing criterion and monitoring system is proposed to integrate the CVP partitioning method as a rebalancer. A set of numerical experiments show that the target of dynamic load balance is achieved with the proposed rebalancing strategy; (2) A localized nested hierarchical data structure is developed to eliminate the bottleneck of memory overhead. The PFNS strategy in [1] is tailored and extended to current data structure. The efficiency is increased further due to the local tree construction. (3) A diffused graph is proposed to improve the efficiency of the graph-based communication strategy. The graph-based communication strategy is employed to handle both the construction

of ghost buffer particle and particle migration. Numerical tests and runtime analyses demonstrate that negligible time is required for graph construction comparing to total runtime. (4) A set of performance tests are performed for both uniform and non-uniform particle distributions, and in both cases, scalable performance is observed.

The Lagrangian Inertial CVP method improves the performance of the original CVP method in terms of dynamic load balancing. The primary objectives of LICVP are to improve the incremental property and to gain the capability of handling problems involving skew-aligned load distribution and large void regions. Two concepts are introduced in this part of the current thesis: (1) By defining a background velocity, Voronoi particles are able to track the motion of local sub-domains and characterize the topological variation of the system with better accuracy. Rebalancing upon the updated Voronoi-particle positions improves the incremental property remarkably. (2) The performance of simulations with extremely anisotropic computation-load distribution is improved utilizing the proposed Inertial CVP method. Due to a splitting operator, the Voronoi-particle motion is insensitive of the load variation along directions of minimum interest. As a consequence, the incremental property is enhanced and convergence is improved. Additionally, an adaptive filter is proposed, which allows the dynamic selection of partitioning strategy according to the evolution of load distribution. The selection procedure guarantees a relative balance between data-redistribution and inter-processor communication cost in extreme situations.

The SPH-based parallel mesh generation method addresses the issue of partitioning the domain, optimizing communication volume, generating an optimized adaptive unstructured mesh in parallel. All the targets are accomplished within the same set of consistent formulations. The main contribution can be summarized as: (1) A unified target density function is defined to characterize the targets of both domain decomposition and mesh generation. The target density function can be any smooth scalar field considering various geometrical features and user-defined inputs. By utilizing a background Cartesian mesh and level-set function, the total number of mesh vertices and target mass for each sub-domain can be determined a priori; (2) A parallelization strategy is developed and a set of physics motivated governing equations is proposed to achieve all the underlying targets consistently. A surface tension model is introduced to previous particle-based mesh generator [5] to handle the additional target of optimizing the communication volume in a parallel environment. During the domain decomposition stage, the mesh quality is improved simultaneously in the interior region of each sub-domain. Once a steady state is achieved, the mesh quality near the interface region is optimized by gradually alleviating the surface tension force. (3) A multi-phase SPH formulation is utilized to solve the governing equations. The previously-developed mesh generator [5] is extended to higher dimensions and parallelized with both MPI and TBB technique. Numerical results demonstrate that the resulting sub-domains feature compact and regularized shape, and high-quality mesh is generated. The communication overhead caused by the optimization of mesh quality near the interface is limited even in cases with complex geometry and large spacial adaptivity; (4) With the proposed parallel mesh generation method, high quality triangle/tetrahedron mesh can be generated without the need of constructing Delaunay Triangulation/Tetrahedralization explicitly. Since only local information are required during the simulation and the same set of governing equations are solved for all the particles, the proposed method features high consistency and code reusability. Benefiting from the scalable parallel environment designed in paper I [72], the mesh generation procedure is able to exploit both

fine-grained and coarse-grained parallelization.

5.2 Outlooks

Although the parallel framework developed in this work achieves a scalable performance, the code itself is not intensively optimized for speed. Several optimization techniques can be applied to improve the performance, e.g. adaptive time-stepping schemes, using structure of arrays (SOA), stream-based parallelism, and etc. In term of the CLL-based multi-resolution data structure, currently a direct grid mapping function is applied to calculate the hash value of each key. This approach is highly efficient, whereas would create a memory bottleneck if most of the domain is filled with void, where a giant list needs to be stored physically. This issue can be mitigated by using spatial hashing, which can represent infinite domains with sufficiently low memory consumptions. Last but not least, extending the current framework with GPU capability will bring a significant leap in performance. The implementation is relatively easy comparing to tree data structures, since most of the algorithms are designed in a way that high concurrency can be achieved. In addition to the performance improvements, more particle-based methods, e.g. Incompressible SPH, SPH for solid mechanics, Molecular Dynamics and Dissipative Particle Dynamics, can be included in future to extend the capability of the current framework.

Regarding to the mesh generation method developed in this work, from a performance point of view, the current form is still considerably more expensive than the state-of-the-art Delaunay-based methods, despite all the unique characteristics and advantages. Future development can be concluded as following: (1) the formulation presented is consistent and can be extended to the parallel anisotropic mesh generation with a rotating tensor; (2) as mentioned in the above paragraph, the proposed mesh generator can be parallelized in GPU, which will result in a speedup of 10X to 100X; (3) more studies on initial particle seeding strategies will be carried out (this issue is studied in my latest preprint [77]); (4) it is also possible to couple the proposed method with other existing meshers or point cloud generators which feature high performance in terms of generating an initial particle distribution, e.g. the Advancing Front Point Generation Method [141].

Bibliography

[1] Lin Fu et al. "Parallel fast-neighbor-searching and communication strategy for particle-based methods". In: *Engineering Computations* 36.3 (2019), pp. 899–929.

[2] Message P Forum. *MPI: A Message-Passing Interface Standard*. Tech. rep. Knoxville, TN, USA, 1994.

[3] Gilberto Contreras and Margaret Martonosi. "Characterizing and improving the performance of intel threading building blocks". In: *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on*. IEEE. 2008, pp. 57–66.

[4] Lin Fu, Xiangyu Y Hu, and Nikolaus A Adams. "A physics-motivated Centroidal Voronoi Particle domain decomposition method". In: *Journal of Computational Physics* 335 (2017), pp. 718–735.

[5] Lin Fu et al. "An isotropic unstructured mesh generation method based on a fluid relaxation analogy". In: *Communications in Computational Physics, in press, 2019* 350 (2019), pp. 396–431. ISSN: 0045-7825.

[6] Martin Jutzi et al. "Modeling asteroid collisions and impact processes". In: *arXiv preprint arXiv:1502.01844* (2015).

[7] MB Davies et al. "A comparison between SPH and PPM: simulations of stellar collisions". In: *Astronomy and Astrophysics* 272 (1993), p. 430.

[8] Johannes Weirather et al. "A Smoothed Particle Hydrodynamics Model for Laser Beam Melting of Ni-based Alloy 718". In: *Computers & Mathematics with Applications* 78.7 (2019), pp. 2377–2394.

[9] Chi Zhang, Xiangyu Y Hu, and Nikolaus A Adams. "A generalized transport-velocity formulation for smoothed particle hydrodynamics". In: *Journal of Computational Physics* (2017).

[10] Leon B Lucy. "A numerical approach to the testing of the fission hypothesis". In: *The astronomical journal* 82 (1977), pp. 1013–1024.

[11] Robert a. Gingold and Joseph J Monaghan. "Smoothed particle hydrodynamics: theory and application to non-spherical stars". In: *Monthly notices of the royal astronomical society* 181.3 (1977), pp. 375–389. ISSN: 00358711. DOI: `http://dx.doi.org/10.1093/mnras/181.3.375`.

[12] Volker Springel. "The cosmological simulation code GADGET-2". In: *Monthly Notices of the Royal Astronomical Society* 364.4 (2005), pp. 1105–1134.

[13] Chia-Yu Hu et al. "SPHGal: smoothed particle hydrodynamics with improved accuracy for galaxy simulations". In: *Monthly Notices of the Royal Astronomical Society* 443.2 (2014), pp. 1173–1191.

[14] Philip F Hopkins. "A new class of accurate, mesh-free hydrodynamic simulation methods". In: *Monthly Notices of the Royal Astronomical Society* 450.1 (2015), pp. 53–110.

[15] C Zhang, X Y Hu, and N A Adams. "A weakly compressible SPH method based on a low-dissipation Riemann solver". In: *Journal of Computational Physics* 335 (2017), pp. 605–620.

[16] Zhe Ji et al. "Numerical simulations of oil flow inside a gearbox by Smoothed Particle Hydrodynamics (SPH) method". In: *Tribology International* 127 (2018), pp. 47–58.

[17] Daniel J Price. "Smoothed particle hydrodynamics and magnetohydrodynamics". In: *Journal of Computational Physics* 231.3 (2012), pp. 759–794.

[18] Joseph Joe J Monaghan. "Simulating free surface flows with SPH". In: *Journal of computational physics* 110.2 (1994), pp. 399–406.

[19] Joseph P Morris, Patrick J Fox, and Yi Zhu. "Modeling low Reynolds number incompressible flows using SPH". In: *Journal of computational physics* 136.1 (1997), pp. 214–226.

[20] S Marrone et al. "δ-SPH model for simulating violent impact flows". In: *Computer Methods in Applied Mechanics and Engineering* 200.13 (2011), pp. 1526–1542.

[21] X Y Hu and Nikolaus A Adams. "A multi-phase SPH method for macroscopic and mesoscopic flows". In: *J. Comput. Phys.* 213.2 (2006), pp. 844–861.

[22] S Adami, X Y Hu, and Nikolaus A Adams. "A new surface-tension formulation for multi-phase SPH using a reproducing divergence approximation". In: *Journal of Computational Physics* 229.13 (2010), pp. 5011–5021.

[23] P W Randles and L D Libersky. "Smoothed particle hydrodynamics: some recent improvements and applications". In: *Computer methods in applied mechanics and engineering* 139.1 (1996), pp. 375–408.

[24] Carla Antoci, Mario Gallati, and Stefano Sibilla. "Numerical simulation of fluid–structure interaction by SPH". In: *Computers & Structures* 85.11 (2007), pp. 879–890.

[25] Luhui Han and Xiangyu Hu. "SPH modeling of fluid-structure interaction". In: *Journal of Hydrodynamics* 30.1 (2018), pp. 62–69.

[26] Abbas Khayyer et al. "Multi-resolution MPS for incompressible fluid-elastic structure interactions in ocean engineering". In: *Applied Ocean Research* 82 (2019), pp. 397–414.

[27] Erwan Bertevas et al. "Smoothed particle hydrodynamics (SPH) modeling of fiber orientation in a 3D printing process". In: *Physics of Fluids* 30.10 (2018), p. 103103.

[28] Lin Fu et al. "A novel partitioning method for block-structured adaptive meshes". In: *Journal of Computational Physics* 341 (2017), pp. 447–473.

[29] Lin Fu, Xiangyu Y Hu, and Nikolaus A Adams. "Adaptive anisotropic unstructured mesh generation method based on fluid relaxation analogy". In: *Communications in Computational Physics* 27 (2019), pp. 1275–1308.

[30] José M Domínguez et al. "New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters". In: *Computer Physics Communications* 184.8 (2013), pp. 1848–1860.

[31] Jose L Cercos-Pita. "AQUAgpusph, a new free 3D SPH solver accelerated with OpenCL". In: *Computer Physics Communications* 192 (2015), pp. 295–312.

[32] Alexis Hérault, Giuseppe Bilotta, and Robert A Dalrymple. "Sph on gpu with cuda". In: *Journal of Hydraulic Research* 48.S1 (2010), pp. 74–79.

[33] David Greif and Markus Ihmsen. "MESHLESS SIMULATION APPROACH FOR WATER MANAGEMENT USING SMOOTHED PARTICLE HYDRODYNAMICS". In: *ASTFE Digital Library*. Begel House Inc. 2019.

[34] Hua Liu et al. "Numerical modelling of oil distribution and churning gear power losses of gearboxes by smoothed particle hydrodynamics". In: *Proceedings of the Institution of Mechanical Engineers, Part J: Journal of Engineering Tribology* 233.1 (2019), pp. 74–86.

[35] Lin Fu and Zhe Ji. "An optimal particle setup method with Centroidal Voronoi Particle dynamics". In: *Computer Physics Communications* 234 (2019), pp. 72–92.

[36] Daniel A Barcarolo et al. "Adaptive particle refinement and derefinement applied to the smoothed particle hydrodynamics method". In: *Journal of Computational Physics* 273 (2014), pp. 640–657.

[37] R Vacondio et al. "Variable resolution for SPH in three dimensions: towards optimal splitting and coalescing for dynamic adaptivity". In: *Computer Methods in Applied Mechanics and Engineering* 300 (2016), pp. 442–460.

[38] Wei Hu et al. "A consistent multi-resolution smoothed particle hydrodynamics method". In: *Computer Methods in Applied Mechanics and Engineering* 324 (2017), pp. 278–299.

[39] Laurent Chiron et al. "Analysis and improvements of Adaptive Particle Refinement (APR) through CPU time, accuracy and robustness considerations". In: *Journal of Computational Physics* 354 (2018), pp. 552–575.

[40] Shu-ichiro Inutsuka. "Reformulation of smoothed particle hydrodynamics with Riemann solver". In: *Journal of Computational Physics* 179.1 (2002), pp. 238–267.

[41] Giuseppe Murante et al. "Hydrodynamic simulations with the Godunov smoothed particle hydrodynamics". In: *Monthly Notices of the Royal Astronomical Society* 417.1 (2011), pp. 136–153.

[42] Kunal Puri and Prabhu Ramachandran. "Approximate Riemann solvers for the Godunov SPH (GSPH)". In: *Journal of Computational Physics* 270 (2014), pp. 432–458.

[43] Holger Wendland. "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree". In: *Advances in computational Mathematics* 4.1 (1995), pp. 389–396.

[44] Joseph J Monaghan and John C Lattanzio. "A refined particle method for astrophysical problems". In: *Astronomy and astrophysics* 149 (1985), pp. 135–143.

[45] Walter Dehnen and Hossam Aly. "Improving convergence in smoothed particle hydrodynamics simulations without pairing instability". In: *Monthly Notices of the Royal Astronomical Society* 425.2 (2012), pp. 1068–1082.

[46] Joe J Monaghan. "Smoothed particle hydrodynamics". In: *Annual review of astronomy and astrophysics* 68.8 (2005), p. 1703. ISSN: 0034-4885. DOI: 10.1088/0034-4885/68/8/R01. arXiv: 0507472v1 [astro-ph].

[47] M B Liu and G R Liu. "Smoothed particle hydrodynamics (SPH): an overview and recent developments". In: *Arch. Comput. Methods Eng.* 17.1 (2010), pp. 25–76.

[48] Chi-Wang Shu and Stanley Osher. "Efficient implementation of essentially non-oscillatory shock-capturing schemes". In: *Journal of Computational Physics* 77.2 (1988), pp. 439–471.

[49] Philip L Roe. "Approximate Riemann solvers, parameter vectors, and difference schemes". In: *Journal of computational physics* 43.2 (1981), pp. 357–372.

[50] Paul Batten et al. "On the choice of wavespeeds for the HLLC Riemann solver". In: *SIAM Journal on Scientific Computing* 18.6 (1997), pp. 1553–1570.

[51] John K Dukowicz. "A general, non-iterative Riemann solver for Godunov's method". In: *Journal of Computational Physics* 61.1 (1985), pp. 119–137.

[52] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[53] Guang-Shan Jiang and Chi-Wang Shu. "Efficient Implementation of Weighted ENO Schemes". In: *Journal of Computational Physics* 126.1 (1996), pp. 202–228.

[54] Bram Van Leer. "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method". In: *Journal of computational Physics* 32.1 (1979), pp. 101–136.

[55] Shang-Hua Teng and Chi Wai Wong. "Unstructured mesh generation: Theory, practice, and perspectives". In: *International Journal of Computational Geometry & Applications* 10.03 (2000), pp. 227–266.

[56] Jonathan Richard Shewchuk. "Unstructured mesh generation". In: *Combinatorial Scientific Computing* 12.257 (2012), p. 2.

[57] Jeffrey Slotnick et al. "CFD vision 2030 study: a path to revolutionary computational aerosciences". In: (2014).

[58] Daming Feng, Andrey N Chernikov, and Nikos P Chrisochoides. "A hybrid parallel Delaunay image-to-mesh conversion algorithm scalable on distributed-memory clusters". In: *Procedia engineering* 163 (2016), pp. 59–71.

[59] Qiang Du and Desheng Wang. "Recent progress in robust and quality Delaunay mesh generation". In: *Journal of Computational and Applied Mathematics* 195.1-2 (2006), pp. 8–23.

[60] Joachim Schöberl. "NETGEN An advancing front 2D/3D-mesh generator based on abstract rules". In: *Computing and visualization in science* 1.1 (1997), pp. 41–52.

[61] Rainald Löhner. "Recent advances in parallel advancing front grid generation". In: *Archives of Computational Methods in Engineering* 21.2 (2014), pp. 127–140.

[62] Mark S Shephard and Marcel K Georges. "Automatic three-dimensional mesh generation by the finite octree technique". In: *International Journal for Numerical methods in engineering* 32.4 (1991), pp. 709–749.

[63] Mark A Yerry and Mark S Shephard. "Automatic three-dimensional mesh generation by the modified-octree technique". In: *International Journal for Numerical Methods in Engineering* 20.11 (1984), pp. 1965–1990.

[64] Jonathan Richard Shewchuk. "Delaunay refinement algorithms for triangular mesh generation". In: *Computational geometry* 22.1-3 (2002), pp. 21–74.

[65] L Paul Chew. "Guaranteed-quality delaunay meshing in 3D (short version)". In: *Proceedings of the thirteenth annual symposium on Computational geometry*. ACM. 1997, pp. 391–393.

[66] Saifeng Ni et al. "Sliver-suppressing tetrahedral mesh optimization with gradient-based shape matching energy". In: *Computer Aided Geometric Design* 52 (2017), pp. 247–261.

[67] Qiang Du, Vance Faber, and Max Gunzburger. "Centroidal Voronoi Tessellations: applications and algorithms". In: *SIAM review* 41.4 (1999), pp. 637–676.

[68] Zichun Zhong et al. "Particle-based anisotropic surface meshing". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 99.

[69] Jonathan R Bronson, Joshua A Levine, and Ross T Whitaker. "Particle systems for adaptive, isotropic meshing of CAD models". In: *Proceedings of the 19th International Meshing Roundtable*. Springer, 2010, pp. 279–296.

[70] Nikos Chrisochoides. "Parallel mesh generation". In: *Numerical solution of partial differential equations on parallel computers*. Springer, 2006, pp. 237–264.

[71] Zhe Ji et al. "A consistent parallel isotropic unstructured mesh generation method based on multi-phase SPH". In: *Computer Methods in Applied Mechanics and Engineering* 363 (2020), p. 112881.

[72] Zhe Ji et al. "A new multi-resolution parallel framework for SPH". In: *Computer Methods in Applied Mechanics and Engineering* 346 (2019), pp. 1156–1178.

[73] Steve Plimpton. "Fast parallel algorithms for short-range molecular dynamics". In: *Journal of computational physics* 117.1 (1995), pp. 1–19.

[74] Pietro Incardona et al. "OpenFPM: A scalable open framework for particle and particle-mesh codes on parallel computers". In: *Computer Physics Communications* (2019).

[75] OpenMP Architecture Review Board. *OpenMP Application Program Interface Vsersion 3.0*. 2008. URL: `http://www.openmp.org/mp-documents/spec30.pdf`.

[76] John Nickolls et al. "Scalable Parallel Programming with CUDA". In: *Queue* 6.2 (2008), pp. 40–53. ISSN: 1542-7730. DOI: `10.1145/1365490.1365500`. URL: `http://doi.acm.org/10.1145/1365490.1365500`.

[77] Zhe Ji et al. "A feature-aware SPH for isotropic unstructured mesh generation". In: *Computer Methods in Applied Mechanics and Engineering* 375 (2021), p. 113634.

[78] Steven J Owen. "A survey of unstructured mesh generation technology." In: *IMR* 239 (1998), p. 267.

[79] Joe F Thompson, Bharat K Soni, and Nigel P Weatherill. *Handbook of grid generation*. CRC press, 1998.

[80] Daming Feng et al. "Scalable 3D hybrid parallel Delaunay image-to-mesh conversion algorithm for distributed shared memory architectures". In: *Procedia Engineering* 124 (2015), pp. 18–30.

[81] L Paul Chew. *Guaranteed-quality triangular meshes*. Tech. rep. Cornell University, 1989.

[82] Nigel P Weatherill and Oubay Hassan. "Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints". In: *International Journal for Numerical Methods in Engineering* 37.12 (1994), pp. 2005–2039.

[83] Jonathan Richard Shewchuk. "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator". In: *Workshop on Applied Computational Geometry*. Springer. 1996, pp. 203–222.

[84] Alper Üngör. "Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations". In: *Latin American Symposium on Theoretical Informatics*. Springer. 2004, pp. 152–161.

[85] Nigel P Weatherill. "Delaunay triangulation in computational fluid dynamics". In: *Computers & Mathematics with Applications* 24.5-6 (1992), pp. 129–150.

[86] William H Frey. "Selective refinement: a new strategy for automatic node placement in graded triangular meshes". In: *International Journal for Numerical Methods in Engineering* 24.11 (1987), pp. 2183–2200.

[87] Qiang Du and Desheng Wang. "Tetrahedral mesh generation and optimization based on Centroidal Voronoi Tessellations". In: *International journal for numerical methods in engineering* 56.9 (2003), pp. 1355–1373.

[88] Saifeng Ni et al. "Field-Aligned and Lattice-Guided Tetrahedral Meshing". In: *Computer Graphics Forum*. Vol. 37. 5. Wiley Online Library. 2018, pp. 161–172.

[89] Bryan Matthew Klingner and Jonathan Richard Shewchuk. "Aggressive tetrahedral mesh improvement". In: *Proceedings of the 16th international meshing roundtable*. Springer. 2008, pp. 3–23.

[90] Jianjun Chen et al. "Domain decomposition approach for parallel improvement of tetrahedral meshes". In: *Journal of Parallel and Distributed Computing* 107 (2017), pp. 101–113.

[91] Miriah D Meyer, Pierre Georgel, and Ross T Whitaker. "Robust particle systems for curvature dependent sampling of implicit surfaces". In: *International Conference on Shape Modeling and Applications 2005 (SMI'05)*. IEEE. 2005, pp. 124–133.

[92] Démian Nave, Nikos Chrisochoides, and L Paul Chew. "Guaranteed-quality parallel Delaunay refinement for restricted polyhedral domains". In: *Computational Geometry* 28.2-3 (2004), pp. 191–215.

[93] Rainald Lohner and Juan R Cebral. "Parallel advancing front grid generation". In: *in International Meshing Roundtable, Sandia National Labs*. Citeseer. 1999.

[94] Jérôme Galtier and Paul Louis George. "Prepartitioning as a way to mesh subdomains in parallel". In: *in 5th International Meshing Roundtable*. Citeseer. 1996.

[95] Leonidas Linardakis and Nikos Chrisochoides. "Delaunay decoupling method for parallel guaranteed quality planar mesh refinement". In: *SIAM Journal on Scientific Computing* 27.4 (2006), pp. 1394–1423.

[96] L Paul Chew, Nikos Chrisochoides, and Florian Sukup. "Parallel constrained Delaunay meshing". In: *ASME APPLIED MECHANICS DIVISION-PUBLICATIONS-AMD* 220 (1997), pp. 89–96.

[97] Jean-François Remacle, Vincent Bertrand, and Christophe Geuzaine. "A two-level multithreaded Delaunay kernel". In: *Procedia Engineering* 124 (2015), pp. 6–17.

[98] Hoby Rakotoarivelo, Franck Ledoux, and Franck Pommereau. "Fine-grained locality-aware parallel scheme for anisotropic mesh adaptation". In: *Procedia engineering* 163 (2016), pp. 123–135.

[99] Georgios Rokos et al. "Thread parallelism for highly irregular computation in anisotropic mesh adaptation". In: *Proceedings of the 3rd International Conference on Exascale Applications and Software*. University of Edinburgh. 2015, pp. 103–108.

[100] Daming Feng, Andrey N Chernikov, and Nikos P Chrisochoides. "Two-level locality-aware parallel Delaunay image-to-mesh conversion". In: *Parallel Computing* 59 (2016), pp. 60–70.

[101] Daming Feng, Andrey N Chernikov, and Nikos P Chrisochoides. "A hybrid parallel Delaunay image-to-mesh conversion algorithm scalable on distributed-memory clusters". In: *Procedia engineering* 163 (2016), pp. 59–71.

[102] Stanley Osher and James A Sethian. "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations". In: *Journal of computational physics* 79.1 (1988), pp. 12–49.

[103] Frederic Gibou, Ronald Fedkiw, and Stanley Osher. "A review of level-set methods and some recent applications". In: *Journal of Computational Physics* 353 (2018), pp. 82–109.

[104] Jian Luo, XY Hu, and Nikolaus A Adams. "A conservative sharp interface method for incompressible multiphase flows". In: *Journal of Computational Physics* 284 (2015), pp. 547–565.

[105] Kirk Schloegel, George Karypis, and Vipin Kumar. *Graph partitioning for high performance scientific simulations*. Army High Performance Computing Research Center, 2000.

[106] Bruce Hendrickson and Karen Devine. "Dynamic load balancing in computational mechanics". In: *Computer methods in applied mechanics and engineering* 184.2 (2000), pp. 485–500.

[107] Ayd\in Buluç et al. "Recent advances in graph partitioning". In: *Algorithm Engineering*. Springer, 2016, pp. 117–158.

[108] Erik Boman et al. "Zoltan 3.0: parallel partitioning, load balancing, and data-management services; user's guide. Sandia National Laboratories, Albuquerque". In: *Rep. SAND2007-4748W* (2007).

[109] Marsha J Berger and Shahid H Bokhari. "A partitioning strategy for nonuniform problems on multiprocessors". In: *IEEE Transactions on Computers* 5 (1987), pp. 570–580.

[110] Horst D Simon. "Partitioning of unstructured problems for parallel processing". In: *Computing Systems in Engineering* 2.2-3 (1991), pp. 135–148.

[111] Hans Sagan. *Space-filling curves*. Springer Science & Business Media, 2012.

[112] Alex Pothen, Horst D Simon, and Kang-Pu Liou. "Partitioning sparse matrices with eigenvectors of graphs". In: *SIAM journal on matrix analysis and applications* 11.3 (1990), pp. 430–452.

[113] George Karypis and Vipin Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs". In: *SIAM Journal on scientific Computing* 20.1 (1998), pp. 359–392.

[114] Umit V Catalyurek and Cevdet Aykanat. "Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication". In: *IEEE Transactions on parallel and distributed systems* 10.7 (1999), pp. 673–693.

[115] Henning Meyerhenke, Burkhard Monien, and Thomas Sauerwald. "A new diffusion-based multilevel algorithm for computing graph partitions of very high quality". In: *2008 IEEE International Symposium on Parallel and Distributed Processing*. IEEE. 2008, pp. 1–13.

[116] Henning Meyerhenke, Burkhard Monien, and Stefan Schamberger. "Graph partitioning and disturbed diffusion". In: *Parallel Computing* 35.10-11 (2009), pp. 544–569.

[117] Atsuyuki Okabe et al. *Spatial tessellations: concepts and applications of Voronoi diagrams*. Vol. 501. John Wiley & Sons, 2009.

[118] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[119] Qiang Du, Maria Emelianenko, and Lili Ju. "Convergence of the Lloyd algorithm for computing Centroidal Voronoi Tessellations". In: *SIAM journal on numerical analysis* 44.1 (2006), pp. 102–119.

[120] Roger W Hockney and James W Eastwood. *Computer simulation using particles*. CRC Press, 1988.

[121] Loup Verlet. "Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules". In: *Physical review* 159.1 (1967), p. 98.

[122] Lars Arge et al. "The priority R-tree: A practically efficient and worst-case optimal R-tree". In: *ACM Transactions on Algorithms (TALG)* 4.1 (2008), p. 9.

[123] Jose M Domínguez et al. "Neighbour lists in smoothed particle hydrodynamics". In: *International Journal for Numerical Methods in Fluids* 67.12 (2011), pp. 2026–2042.

[124] Markus Ihmsen et al. "A parallel SPH implementation on multi-core CPUs". In: *Computer Graphics Forum*. Vol. 30. 1. Wiley Online Library. 2011, pp. 99–112.

[125] Lars Hernquist and Neal Katz. "TREESPH-A unification of SPH with the hierarchical tree method". In: *The Astrophysical Journal Supplement Series* 70 (1989), pp. 419–446.

[126] James W Wadsley, Joachim Stadel, and Thomas Quinn. "Gasoline: a flexible, parallel implementation of TreeSPH". In: *New astronomy* 9.2 (2004), pp. 137–158.

[127] Torsten Hoefler and Jesper Larsson Traff. "Sparse collective operations for MPI". In: *2009 IEEE International Symposium on Parallel & Distributed Processing*. IEEE. 2009, pp. 1–8.

[128] Ivo F Sbalzarini et al. "PPM–A highly efficient parallel particle–mesh library for the simulation of continuum systems". In: *Journal of Computational Physics* 215.2 (2006), pp. 566–588.

[129] Jayadev Misra and David Gries. "A constructive proof of Vizing's theorem". In: *Information Processing Letters* 41.3 (1992), pp. 131–133.

[130] Daniel J Price et al. "Phantom: A smoothed particle hydrodynamics and magnetohydrodynamics code for astrophysics". In: *Publications of the Astronomical Society of Australia* 35 (2018).

[131] Ruben M Cabezón, Domingo Garcia-Senz, and Joana Figueira. "SPHYNX: an accurate density-based SPH method for astrophysical applications". In: *Astronomy & Astrophysics* 606 (2017), A78.

[132] James W Wadsley, Benjamin W Keller, and Thomas R Quinn. "Gasoline2: a modern smoothed particle hydrodynamics code". In: *Monthly Notices of the Royal Astronomical Society* 471.2 (2017), pp. 2357–2369.

[133] Jeroen Bédorf and Simon Portegies Zwart. "Bonsai-SPH: A GPU accelerated astrophysical Smoothed Particle Hydrodynamics code". In: *arXiv preprint arXiv:1909.07439* (2019).

[134] Alejandro J C Crespo et al. "DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH)". In: *Computer Physics Communications* 187 (2015), pp. 204–216.

[135] John V W Reynders et al. "POOMA: A framework for scientific simulations on parallel architectures". In: *Parallel Programming in C+* (1996), pp. 547–588.

[136] Pavan Balaji et al. "MPI on a Million Processors". In: *European Parallel Virtual Machine/Message Passing Interface Users Group Meeting*. Springer. 2009, pp. 20–30.

[137] S Adami, X Y Hu, and N A Adams. "A generalized wall boundary condition for smoothed particle hydrodynamics". In: *Journal of Computational Physics* 231.21 (2012), pp. 7057–7075.

[138] Kohei Murotani et al. "Development of hierarchical domain decomposition explicit MPS method and application to large-scale tsunami analysis with floating objects". In: *Journal of Advanced Simulation in Science and Engineering* 1.1 (2014), pp. 16–35.

[139] Chris Walshaw, Mark Cross, and Martin G Everett. "Parallel dynamic graph partitioning for adaptive unstructured meshes". In: *Journal of Parallel and Distributed Computing* 47.2 (1997), pp. 102–108.

[140] Zhe Ji et al. "A Lagrangian Inertial Centroidal Voronoi Particle method for dynamic load balancing in particle-based simulations". In: *Computer Physics Communications* 239 (2019), pp. 53–63. ISSN: 0010-4655.

[141] Rainald Löhner and Eugenio Onate. "An advancing front point generation technique". In: *Communications in numerical methods in engineering* 14.12 (1998), pp. 1097–1108.

Appendix A

Original journal papers

Here, the peer-reviewed journal publications of the present work are attached.

A.1 Paper I

Zhe Ji, Lin Fu, Xiangyu Y. Hu, Nikolaus A. Adams

A new multi-resolution parallel framework for SPH

In *Computer Methods in Applied Mechanics and Engineering*, Volume 346, 2019, pp. 1156-1178, DOI: https://doi.org/10.1016/j.cma.2018.09.043.

Contribution: My contribution to this work was the development of the method and the corresponding computer code for its implementation. I performed simulations and analyzed the results, and wrote the manuscript for the publication.

Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 346 (2019) 1156–1178
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A new multi-resolution parallel framework for SPH

Zhe Ji, Lin Fu, Xiangyu Y. Hu*, Nikolaus A. Adams

Chair of Aerodynamics and Fluid Mechanics, Department of Mechanical Engineering, Technical University of Munich, 85748 Garching, Germany

Received 23 March 2018; received in revised form 27 September 2018; accepted 27 September 2018
Available online 10 October 2018

Abstract

In this paper we present a new multi-resolution parallel framework, which is designed for large-scale SPH simulations of fluid dynamics. An adaptive rebalancing criterion and monitoring system is developed to integrate the CVP partitioning method as rebalancer to achieve dynamic load balancing of the system. A localized nested hierarchical data structure is developed in cooperation with a tailored parallel fast-neighbor-search algorithm to handle problems with arbitrarily adaptive smoothing-length and to construct ghost buffer particles in remote processors. The concept of "diffused graph" is proposed in this paper to improve the performance of the graph-based communication strategy. By utilizing the hybrid parallel model, the framework is able to exploit the full parallel potential of current state-of-the-art clusters based on Distributed Shared Memory (DSM) architectures. A range of gas dynamics benchmarks are investigated to demonstrate the capability of the framework and its unique characteristics. The performance is assessed in detail through intensive numerical experiments at various scales.

Keywords: Smoothed Particle Hydrodynamics; Compressible fluid dynamics; Centroidal Voronoi Particle method; High-performance parallel computing; Fast neighbor search; Edge coloring

1. Introduction

As a Lagrangian method, Smoothed Particle Hydrodynamics (SPH) [1,2] ensures Galilean invariance and conserves mass, momentum and energy inherently. It has been explored and demonstrated for a wide range of applications. Moreover, since SPH solves fluid-dynamics equations on discretized particles carrying specific mass, the space resolution can be adapted which is particularly interesting for compressible fluid dynamics [3]. Due to simple coupling with a gravitational solver, SPH is a preferred approach in computational astrophysics [4,3,5]. Other effects, such as radiation fields and magnetic fields, can also be included and help to explore the formation and evolution of galaxies [6]. For incompressible or weakly compressible hydrodynamics, recently there has been an increasing interest to employ multi-resolution SPH to locally resolve fluid details of interest in a more efficient way [7–9].

Several open-source frameworks exist for the large-scale parallel simulation of particle-based methods. DualSPHysics [10] is a high performance code developed mainly for free-surface flow phenomena in weakly-compressible

* Corresponding author.
E-mail addresses: zhe.ji@tum.de (Z. Ji), lin.fu@tum.de (L. Fu), xiangyu.hu@tum.de (X.Y. Hu), nikolaus.adams@tum.de (N.A. Adams).

https://doi.org/10.1016/j.cma.2018.09.043
0045-7825/

fluid dynamics. The publicly available code is parallelized with OpenMP [11] and CUDA [12] aimed at shared-memory platforms or single-GPU systems. In Molecular Dynamics (MD) and Dissipative Particle Dynamics (DPD) simulations, LAMMPS [13] is a well-established and massively optimized code. LAMMPS features parallelism in both shared-memory and distributed-memory systems using various parallel techniques, e.g. Message Passing Interface (MPI) [14], OpenMP and CUDA. Regarding frameworks for particle and particle-mesh method in distributed memory systems, both POOMA [15] and PPM [16] have been developed and established with scalable parallel performance. More recently, as the successor to the PPM Library, OpenFPM [17] provides higher-level abstractions for simulations using particles only and hybrid particle-mesh, and is optimized intensively to achieve high-performance in modern architectures.

Some preliminary work has been published to tackle the difficulties encountered in extending codes with adaptive-resolution capability, e.g. neighbor lists for adaptive resolution [18], and variable resolution algorithms for particle-based simulations [19,8]. However, the support for fully parallelized adaptive-resolution in DSM systems is generally still limited in the aforementioned codes. To deal with adaptive-smoothing-length SPH in high-performance parallel computing, a versatile and flexible framework is required to handle a set of common issues, e.g. domain decomposition, adaptive data structure, parallel fast neighbor search, data communication, etc. A state-of-the-art code with adaptive-resolution capability is the tree-based SPH solver GADGET-2 [4]. GADGET-2 provides a hierarchical representation of particles using the tree method. The domain decomposition is handled based on the Peano–Hilbert curve. A collective hypercube communication strategy is developed to hide communication latency. In the past decades, this method has become increasingly popular for SPH as well as particle-based N-body methods.

In this paper, we follow an alternative approach by introducing a new multi-resolution parallel framework employing several algorithms from previous work [20–22]. The objective is to seek a succinct yet unified solution to address the existing challenges. In order to integrate the aforementioned algorithms as a consistent and efficient system, several new techniques are developed in this paper to overcome the difficulties encountered and to optimize the parallel performance.

The Centroidal Voronoi Particle (CVP) method [20] is adopted in our framework instead of a conventional domain decomposition method, such as graph- and geometry-based approaches [23,24]. First, due to the construction of Centroidal Voronoi Tessellation (CVT) [25], the partitioning results in convex, strictly-connected subdomains of small aspect-ratio, which facilitate communication reduction. Second, with a tailored equation of state, both equal- and nonequal-sized partitioning can be achieved straightforwardly with controlled load-balance error by appropriately assigning the target mass for each subdomain. The load balance of each sub-domain is then achieved iteratively by solving a physics-driven Voronoi Particle dynamics (VP). The targets of both load balance and communication reduction are achieved simultaneously by a two-step time integration scheme to incorporate both CVT and VP. Moreover, since only the physical coordinate of the mesh-element is required as input of the CVP method, it is a mesh-element-independent method and can be implemented easily. In our previous work, the CVP method is only applied in static partitioning problems. However, in dynamic systems involving large deformations, the computational load in each subdomain can vary rapidly, therefore, the imbalance of the system will deteriorate the parallel efficiency significantly. In this paper, we propose an adaptive criterion and monitoring system to employ the CVP method as a rebalancer. The adaptive criterion utilizes a time-weighting strategy to dynamically calculate the computational load of each SPH particle. The resulting partitioning features better load-balancing characteristics. Moreover, the proposed monitoring system considers the load imbalance due to the change from both computation and communication consumption. Thus the imbalance of the system is captured more accurately.

One of the computationally most expensive components of the SPH method is to find all stencils, i.e. pairwise interacting particles, within the cut-off radius of each discretization particle. We recently have developed a parallel fast-neighbor-search strategy, which allows to find the pairwise interaction partners with arbitrary smoothing-length precisely and efficiently both on local and remote processors [21]. A table-based multi-level nested hierarchical data structure is tailored to facilitate fast neighbor search. The neighboring particles on remote processors are identified by simply comparing the tag system constructed on the data structure between two neighboring subdomains. In our previous work, the parallel fast-neighbor-search strategy requires that each subdomain saves the data structure of the entire computational domain. The memory requirement quickly becomes a bottleneck when the number of MPI tasks increases. In this paper, we propose a localized hierarchical data structure, where only required information is stored for each subdomain, and the maintenance of the data structure is entirely localized. Therefore, the bottleneck of memory requirement is resolved, and the data management is simplified as well. Moreover, a safeguard skin area is

created to avoid the construction of each local data structure every time-step. The parallel fast-neighbor-search strategy for the local data structure is slightly modified. The cost for tag comparison between two neighboring subdomains is reduced by considering only cells in the overlap region.

Another issue that affects the performance of large-scale simulations is to handle sparse data communication among neighboring nodes [26–28]. According to previous work [29,30,16], the communication network can be characterized by a simple undirected graph, and the edge coloring algorithm can be utilized to reduce the communication latency. In this paper, the communication graph for a multi-level data structure is constructed following [21]. When we apply the communication strategy to the framework, the same issue arises as in the data structure construction. Since a global master–slave approach is required [21], the graph construction procedure does not scale with the partitioning number. Therefore, the runtime due to graph construction at every time-step becomes increasingly significant when the number of partitions grows. To reduce the time consumption of graph construction and overcome the serialization bottleneck, we propose a concept of "diffused graph". The "density" of the communication matrix is tuned by introducing a diffusing distance to the cut-off range of each color. By applying this concept, each subdomain can anticipate potential neighbors within a certain time period in the future. Consequently, the graph construction at every time-step can be avoided. Numerical experiments demonstrate that the time contribution due to graph construction is no longer an issue. Nevertheless, in this paper, the graph-based communication strategy is also applied to optimize particle migration frequency. A similar master–slave strategy is proposed to construct a migration graph in order to achieve high performance.

The current framework is designed to be independent of the specific SPH solver. We demonstrate its application to compressible fluid dynamics, where the resolution varies significantly following the evolution of flow field. A well established Godunov SPH method is implemented, and several benchmarks in gas dynamics are investigated.

The remainder of the paper is organized as follows. (i) In Section 2, we first briefly review some newly developed methods including the CVP method, the parallel fast-neighbor-search and communication strategy for particle-based methods with adaptive smoothing-length. (ii) The main contribution in this paper is presented in Section 3. Several new techniques are elaborated and the overall framework is presented. (iii) Four typical categories of compressible gas dynamics simulations are given in Section 4 to verify the accuracy of the framework and show the dynamic property of the CVP method and the communication strategy. (iv) In Section 5, we carry out a parameter study of the dynamic load balancing strategy and demonstrate the performance of the framework via numerical experiments involving various scales. (v) Concluding remarks are given in the last section.

2. Brief review of the CVP method, parallel fast-neighbor-search and graph-based communication strategy

Before moving on to the detailed description of the framework, in this section we first give an overview of three employed algorithms, i.e. the CVP method [20], the parallel fast-neighbor-search and the graph-based communication strategy [21,22].

2.1. Centroidal Voronoi Particle (CVP) partitioning method

By combining two concepts, i.e. CVT [25] and Voronoi Particle dynamics (VP), CVP is able to achieve high-level compactness of partitioning subdomains and error-controlled load balance simultaneously [20].

With the CVP method, an equilibrium state is calculated iteratively to achieve the aforementioned targets. A two-step time integration scheme is developed to combine these two methods and to achieve better convergence. To compute the CVT diagram [31–33] efficiently, the well-established Lloyd method [34,35] is employed. Voronoi Particle dynamics is then applied to relax the particles towards achieving load balance. The governing equation in Lagrangian form is

$$\frac{d\mathbf{v}_i}{dt} = -\frac{\int_{\Omega_i} \nabla\phi d\sigma}{\int_{\Omega_i} \rho d\sigma} = -\frac{\int_{\partial\Omega_i} \phi d\mathbf{S}}{L_i}, \tag{1}$$

where $\mathbf{v}$ denotes the velocity vector, ϕ a force potential, ρ the density, L_i the mass of Voronoi particle i and $\partial\Omega_i$ the Voronoi cell surface. The force potential of a Voronoi particle is defined as

$$\phi_i = \frac{L_i}{L_{tg,i}}, \tag{2}$$

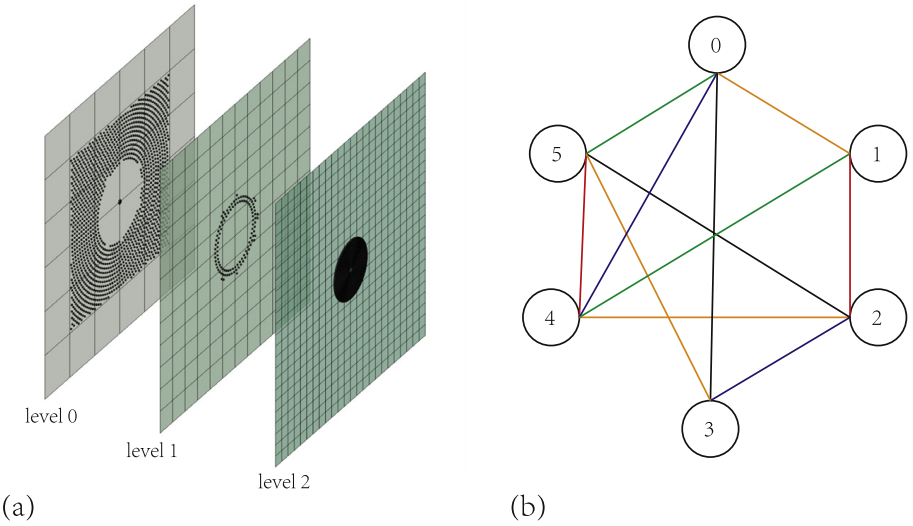


Fig. 1. Data structure: (a). Illustration of the multi-resolution based nested hierarchical data structure comprising three levels; (b). Sketch of a communication graph, and the result of applying the edge coloring algorithm.

where $L_{tg,i}$ is the target mass defined by users with respect to various purposes. For equal-sized partitioning, one can set $L_{tg,i} = \frac{\int_\Omega \rho d\sigma}{k}$, where k equals to the number of partitioning subdomains. The potential at the surface between two neighboring cells is computed with second-order approximation. Note that all variables in Eqs. (1) and (2) are non-dimensional.

The CVP method can be parallelized with the Threading Building Blocks (TBB) [36] and MPI techniques. For more details, the reader is referred to [20].

2.2. Data structure and parallel fast neighbor search (FNS)

A multi-resolution based nested hierarchical data structure [21] is employed in this framework (see Fig. 1(a)). The number of levels can be dynamically defined according to the fluid field information, e.g. multiple levels may be created with respect to the increasing resolution jumps, and vice versa. The number of levels is defined as

$$N_{level} = log_S \left(\frac{r_{c,max}}{r_{c,min}} \right) + 1, \tag{3}$$

where S represents the scale ratio between two levels, $r_{c,max}$ and $r_{c,min}$ are the maximum and minimum scale of SPH particles. At each level, the computational domain is divided recursively with equal-sized cells, and cell linked list (CLL) is built to mark mapped particles. The $cell(j, l+1)$, where j is the index of the cell and $l+1$ denotes the level index, is defined as the child of $cell\left(\lfloor \frac{j}{S} \rfloor, l\right)$.

The parallel fast-neighbor-search algorithm in [21] is adopted. This algorithm utilizes the above data structure. A tag system consisting of several identifiers is constructed within this data structure to facilitate the neighbor search. There are two phases dealing with the neighbor search: (1) construct buffer particles from neighboring subdomains to fill the stencil missing in the local subdomain; (2) build a hierarchical CLL for neighbor search. To construct ghost buffer particles, the data structure is traversed twice, i.e. a “downward prediction evaluation” as well as an ”upward projection evaluation”, to define the region that may affect and be affected by particles in the current subdomain. The tag systems between two neighboring subdomains are compared to construct the ghost buffer particle list. For the second phase, the algorithm is repeated with a single ”upward projection evaluation”, which registers particle address pointers from finer levels to coarser levels. To find real neighbors, one only needs to compare particle information within the current level, thus any cross-level calculation is unnecessary.

2.3. Communication strategy

The communication topology in the SPH method can be defined as a sparse graph [21]. We construct an undirected simple graph $G_0 = (V, E)$, where V, E denote vertex and edge respectively, to characterize the sparse data communication relationship, see Fig. 1(b). Each vertex represents a CPU node, and each edge connects a pair of interacting neighbors. The communication frequency is optimized via utilizing the edge coloring method [30,16] to color all edges so that no edges sharing the same vertex possess identical color [37]. The communication frequency can be explicitly characterized by the number of colors. After applying the edge coloring algorithm, the communication processes are initiated whose number equals the color number, and in each communication sub-iteration only one edge color is involved to avoid conflicts. In the current paper, the edge coloring method is computed utilizing the Boost Graph Library [38].

3. The parallel framework

In this section, details regarding to the proposed technique are presented accordingly. An overview of the entire framework is provided at the end of this section.

3.1. Dynamic load balancing

The computational load distribution in the CVP method is characterized as a density function $\rho(x)$. Each SPH particle, which is referred as interaction particle in the CVP method, is assigned with the physical property of mass proportional to its computational cost. Based on the density field, a certain number of Voronoi particles, identical to the partitioning number, is generated in the computational domain. The mass L_i for each Voronoi particle can be computed by

$$L_i = \int_{\Omega_i} \rho(\mathbf{x}) d\sigma = \sum_{j=0}^{N_i-1} l_j, \tag{4}$$

where Ω_i denotes the region corresponding to Voronoi particle i, N_i the number of interaction particles included in the Voronoi cell i and $d\sigma$ the volume differential. l_j is the computational load of specific interaction particle j.

In Eq. (4), the definition of l_j can be specified by users with respect to different concerns, which will basically result in different CVP diagrams and performance. Moreover, the computational effort, i.e. l_j, of each interaction particle can vary dynamically during the simulation, which may cause load imbalance and therefore decrease parallel performance.

In order to achieve load balance of the system, an accurate evaluation of l_j is important. Assuming that a system is comprised of M computationally intensive subroutines, l_j can be calculated considering the contribution from each subroutine,

$$l_j = \sum_{k=0}^{M} \epsilon_k \bar{l}_{j,k}, \tag{5}$$

where $\bar{l}_{j,k} = \frac{l_{j,k}}{\sum_{j=0}^{N} l_{j,k}}$ is the normalized computational load of SPH particle j with respect to subroutine k. N is the total number of SPH particles. The weight ϵ_k should always sum to 1, i.e. $\sum_{k=0}^{M} \epsilon_k = 1$.

With the SPH method, the major part of the runtime is contributed mainly by two subroutines, i.e. the *neighbor search* (*NS*) and the *calculation of inter-particle forces* (*CF*). We calculate l_j from:

$$l_j = \epsilon \bar{l}_{j,NS} + (1 - \epsilon) \bar{l}_{j,CF}, \tag{6}$$

To assess the computational load due to NS, we set $l_{j,NS}$ equal to the number of distance calculations in nearest neighbor search for each particle. In this scenario, $l_{j,NS}$ may change during the simulation according to the variation of particle scale. Moreover, we set $l_{j,CF} = 1$. Since an iterative algorithm is utilized in this paper for the calculation of smoothing-length, and the actual number of inter-particle interactions is approximately the same for each particle, we assume the runtime contributing to CF is equal for each particle.

To combine the aforementioned two components, we calculate the weight ϵ by an adaptive approach,

$$\epsilon = \frac{\Delta t_{NS}}{\Delta t_{NS} + \Delta t_{CF}}, \tag{7}$$

where $\Delta t_{\{\cdot\}}$ denotes the net runtime elapsed for corresponding subroutines since last load-balance estimate. Upon substituting Eqs. (6) and (7) into Eq. (4), the total mass L_i in subdomain i is obtained. The imbalance error due to computational load change ($E_{L,i}$) can be evaluated straightforwardly by comparing the relative difference between L_i and the target mass $L_{tg,i}$ at the same instance, i.e. $E_{L,i} = \frac{L_i - L_{tg,i}}{L_{tg,i}}$. In practice, the calculation of $L_{tg,i}$ requires global collection among all processors, and extra time is required. Since the total computational load remains approximately the same during a shot time span, we can replace $L_{tg,i}$ with the initial mass $L_{0,i}$ after each repartitioning.

For a dynamic system, the change of communication volume can also cause severe imbalance [39]. With the SPH method, besides the computational load defined in each region Ω_i, the communication load changes as well owing to the deformation of Ω_i and the variation of particle scale in the fringe of Ω_i during the simulation. Thus we further propose another criterion to eliminate the performance deterioration caused by communication imbalance. We define the number of ghost buffer particles in each subdomain to represent the communication load (denoted as Lc_i). The imbalance error is characterized as $E_{Lc,i} = \frac{Lc_i - Lc_{0,i}}{Lc_{0,i}}$, where $Lc_{0,i}$ is the initial value after each repartitioning.

By combining both imbalance indicators, we define an imbalance monitoring tag as

$$R = \begin{cases} 1 & \text{if } E_{Lc,max} > e_{Lc,max} \text{ or } E_{L,max} > e_{L,max} \\ 0 & \text{if else,} \end{cases} \tag{8}$$

$$E_{Lc,max} = max(E_{Lc,0}, \ldots, E_{Lc,n-1}), \tag{9}$$

$$E_{L,max} = max(E_{L,0}, \ldots, E_{L,n-1}), \tag{10}$$

where $e_{Lc,max}$ and $e_{L,max}$ are user defined error tolerance respectively. n is the total number of Voronoi particles. The CVP method is triggered for $R = 1$. In actual simulations, imbalance monitoring is performed every 10 to 20 physical timesteps in order to save time. The performance of the dynamic load balancing strategy is discussed in Section 5.1.

3.2. Local data structure and tag system

The main memory consumption of the data structure is the hierarchical cell-linked list and the tag system. The cell-linked list is constructed at each level with size equal to the total cell number on that level. In each cell, a dynamic container of "pointer" type is utilized to construct the registered particle list. The tag system consists of several arrays of "char" type with the same size of the total cell number in the tree. For large-scale simulation involving thousands of CPU nodes, the memory requirement will become a bottleneck if each subdomain stores all tree information.

To cope with the memory limitation we develop a local hierarchical data structure (referred as "local tree" for simplicity) that only retains local data for physical evolution within the next few timesteps, Fig. 2(a), (b). With the new data structure, the bottleneck encountered from memory requirement is eliminated.

The local tree can be built with minimal effort compared with the total simulation time. It is constructed from the coarsest level and bounded by a box that confines all the particles within this CPU node and buffer particles from remote processors. The tree is built from the coarsest to finest level as described in Section 2.2. Moreover, due to the locality property gained from the domain decomposition strategy, the local tree can be extended with a safeguard skin wrapped around the perimeter, thus the construction procedure can be executed every few timesteps. The length of the time interval is defined by the size of the safeguard skin and the physical wave propagation speed. Fig. 2(b) illustrates the local tree with a safeguard skin for subdomain 8 in Fig. 2(a). The blue area with cell labeled "1" means that this cell is registered with SPH particles, and the others are cells without particles. In this figure, besides the top and right edge that are defined as global domain boundary, the left and bottom edges are both extended one cell further away to avoid local tree reconstruction at every physical timestep.

Furthermore, the efficiency of the neighbor-search algorithm can be further increased as the amount of tag comparisons is greatly reduced due to the local tree construction. In Fig. 2(c), a sketch of local projection and prediction operation for NS is illustrated. The "Influence area" of each subdomain is identified locally through these operations. In [21], the "influence area" is denoted as cells with tag $Q_{j,l} = 1$, where j is the cell index at level l.

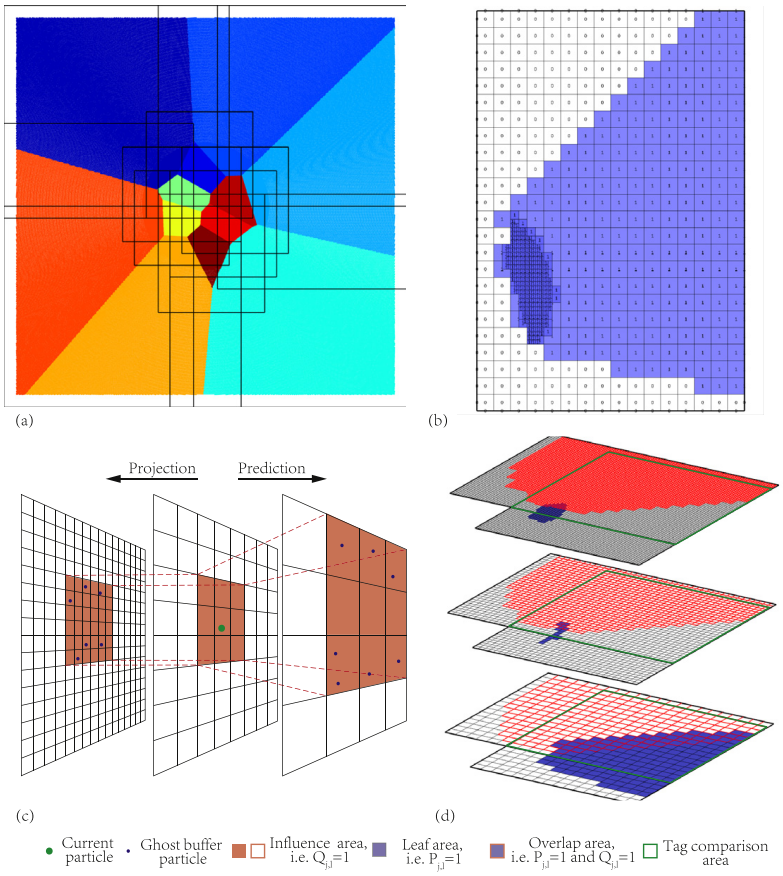


Fig. 2. Data structure: (a) A domain that is partitioned by 12 subdomains, the black lines are the boundaries of each local tree. (b) Local tree with a safe area for subdomain 8 in sub-figure (a) above; (c) Local projection and prediction operation for NS. (d) Illustration of tag comparison between two neighboring subdomains.

Assuming subdomain A interacts with subdomain B, one can find the ghost buffer particles in B by comparing $Q_{j,l}$ from A (area with red color in Fig. 2(d)) with the "leaf area", which corresponds to cells with tag $P_{j,l} = 1$, in subdomain B (area with blue color in Fig. 2(d)). The particles registered in the overlap region of "leaf" and "influence area", i.e. cells with tag $Q_{j,l} = 1$ and $P_{j,l} = 1$, are defined as ghost buffer particles of A registered in B. Fig. 2(d) illustrates the "overlap area" between A and B. It is observed that the tag comparison operation is bounded within the intersecting region between neighboring subdomains, denoted as the "tag comparison area". Due to the local tree construction, the range of "tag comparison area" can be calculated straightforwardly by comparing the start and end cell index of each subdomain. Consequently, for constructing ghost buffer particles, only the traverse within the "tag comparison area" is required, and the number of tag comparison is significantly reduced.

3.3. *Diffused graph*

The graph-based communication strategy is highly efficient since the communication is scheduled by solving the edge coloring problem. In real simulation, the communication topology may change according to the variation of the particle scale, and previous algorithms require to construct G_0 at each timestep. As described above, the graph-construction procedure requires collecting information from Slave nodes, and often the number of edges added is several times larger than the vertex number, which becomes time consuming for a large number of partitioning subdomains.

However, due to the limited wave propagation speed and the locality feature through the CVP method, the communication topology remains bounded within certain number of timesteps. Based on this observation, we propose a diffused graph that can reduce the graph-construction frequency to achieve better performance.

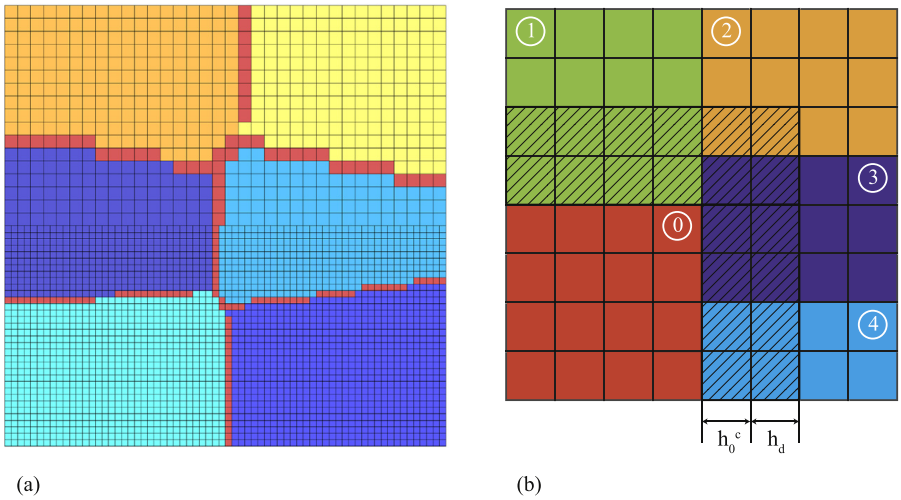


Fig. 3. Graph based communication strategy: (a). The color map used for constructing graph. A domain of size 1×1 is partitioned with 6 MPI tasks (filled with different colors), and the red color is the area where neighboring subdomains overlap; (b). Illustration of constructing diffused graph in a simplified case. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The strategy is to diffuse the interaction area of each colored cell to anticipate potential neighboring subdomains that are not current neighbors. Similarly as with the strategy elaborated in Section 3.2, an "upward projection evaluation" operation is performed in the tree to map the color list from children cell to father cell. The cut-off range of each color is relaxed by

$$h_i^{c,d} = h_i^c + h_d, \tag{11}$$

where h_i^c denotes the cut-off range of color i, $h_i^{c,d}$ the diffused cut-off range of color i, and h_d the diffused distance. Note that Eq. (11) is similar to the concept of "skin distance" defined in constructing the Verlet list [40]. Finally, the tree is traversed again to find each potentially interacting subdomains, i.e. edge connecting corresponding vertices, using the relaxed cut-off radius $h_i^{c,d}$. In Fig. 3 (a), a sketch of the color map containing 6 MPI tasks is illustrated. Each subdomain pair, which mutually interacts within the cut-off range h^c, is defined as a communication edge. Fig. 3(b) illustrates the procedure of constructing a diffused graph. The shaded area is the relaxed cut-off range of subdomain 0, i.e. $h_0^{c,d}$, which is one cell larger than h_0^c. Initially, the subdomain 0 does not consider subdomain 2 as a neighbor. However, after relaxation, subdomain 2 is within $h_0^{c,d}$, and counts as communication neighbor of subdomain 0.

By the diffused graph strategy the communication graph remains intact throughout several iterations. However, the communication matrix becomes denser, i.e. more edges are added to the graph. In our framework, we diffuse the interaction area by one more cell further in each direction, i.e. $\triangle_d = \triangle_l$, and $\triangle_l$ is the cell size at level l where the color is currently located. Consequently, the graph is constructed every 10 timesteps. Performance will be discussed in most detail in Section 5.

3.4. Particle migration

For dynamic load balancing, relocation or migration of SPH particles normally is required after each CVP partitioning to achieve the rebalance of the system. In our framework, the graph-based communication strategy is not only utilized for constructing ghost buffer particles from neighboring subdomains, but also applied to optimize the particle-migration frequency after every rebalancing of the system. It is worth noticing that the particle-migration topology is not necessarily the same with the communication topology, thus G_0 cannot be employed directly to handle the particle migration between subdomains. Similarly with G_0, we characterize the migration topology by introducing a migration graph $G_1 = (V, E)$.

G_1 can be constructed straightforwardly with a Master/Slave communication strategy. After CVP partitioning, every subdomain may possess SPH particles with colors distinct from current subdomain. Such particles are registered as migrating particles. All colors in each subdomain are registered in a color list and the "Master" node collects all

the color lists from corresponding "Slave" nodes. The migration graph G_1 is then constructed in the "Master" node. Moreover, the migration sequence can be optimized by the edge coloring method as well to avoid bandwidth conflicts.

3.5. SPH solver

In this paper, we mainly focus on applications in compressible gas dynamics. Other approaches, such as incompressible [41] or weakly-compressible SPH [42,43], can be paralleled analogously with the current framework. In this section, we mainly discuss a Godunov SPH solver following Ref. [44–46].

The density is calculated via summation by

$$\rho_i = \sum_j m_j W(\left|\mathbf{r}_i - \mathbf{r}_j\right|, h_i), \tag{12}$$

where ρ denotes density, m particle mass, W the kernel function, $\left|\mathbf{r}_i - \mathbf{r}_j\right|$ the distance between particle i and j and h the smoothing length. The Gaussian kernel is adopted for all test cases in Sections 4 and 5.

Unlike in Ref. [44] and [46], where h_i is calculated from a three step process, we calculate h_i from an iterative approach following [47]. A safeguard factor η is defined to set $h_{nominal,i} = \eta h_i$ before calculating density, then particle i is mapped to the tree using $h_{nominal,i}$. Since in resolved flow regions h_i remains approximately constant, η can be adapted according to the variation of the smoothing length from the previous iteration step (n-1) following

$$\eta^n = max(\eta_0, \theta h^n / h^{n-1}), \tag{13}$$

where n is the timestep index, $\eta_0 = 1.03$ and $\theta = 1.2$ are user defined constants. h_i is iterated using the Newton–Raphson method [47].

The discretized governing equation in Lagrangian form for GSPH is

$$\frac{d\mathbf{v}_i}{dt} = -\sum_j m_j p^*_{ij} \left(\frac{1}{(\rho_i)^2} \nabla W_{ij}(h_i) + \frac{1}{(\rho_j)^2} \nabla W_{ij}(h_j) \right), \tag{14}$$

$$\frac{de_i}{dt} = -\sum_j m_j p^*_{ij} \left(\mathbf{v}^*_{ij} - \mathbf{v}^*_i \right) \cdot \left(\frac{1}{(\rho_i)^2} \nabla W_{ij}(h_i) + \frac{1}{(\rho_j)^2} \nabla W_{ij}(h_j) \right), \tag{15}$$

where e_i is the specific internal energy per unit mass for particle i. For ideal gas, the equation of state is defined as

$$p = (\gamma - 1)\rho e, \tag{16}$$

where γ denotes the ratio of specific heats. The $\mathbf{v}^*_i = \mathbf{v}_i + \frac{\delta t}{2} \frac{d\mathbf{v}_i}{dt}$ represents the time centered velocity for particle i. The starred quantity p^*_{ij} and $\mathbf{v}^*_{ij}$ are the intermediate states computed by solving a Riemann problem.

In GSPH, the intermediate state is determined by solving one-dimensional Riemann problems at the imaginary interface along the line joining the pair-wise particles. We employ approximate non-iterative Riemann solvers for better efficiency. Ref. [46] introduces various Riemann solvers for the Lagrangian framework, and the results are compared in detail with different benchmarks. In this paper, two approximate Riemann solvers, the LLF and the Ducowicz solver, are implemented, and the results are demonstrated in Section 4. As input of the Riemann solver the right and left states are approximately reconstructed. Following [44], we use piecewise-linear reconstruction of primitive variables to achieve second-order accuracy in smooth regions. Furthermore, to enforce the monotonicity near discontinuities, the harmonic gradient averaging based limiter proposed by Van Leer [48] is applied.

3.6. Overview of the framework

A detailed flowchart of the developed framework is attached in the Appendix to summarize all the algorithms elaborated before. All the components integrated in this framework are rendered with different colors in order to have a better perspective of the work flow. Since each component can be packed as independent module and offers specific functionality, the framework is highly flexible for developing new algorithms and simple to extend to other particle-based methods as well. Furthermore, in order to achieve high level maintainability and better performance, the code is written in C++ using several open source libraries, e.g. BOOST [49], Voro++ [50] and TBB [36].

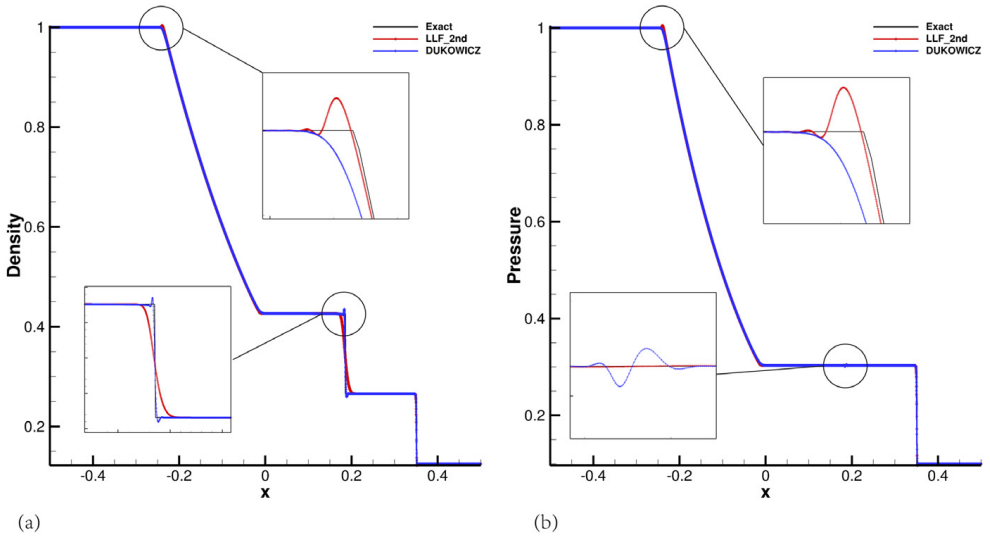


Fig. 4. Sod problem: (a) Density and (b) Pressure distribution at 0.2 s. "Exact" denotes the theoretical solution, and "2nd" represents the second order reconstruction.

4. Numerical validation

In this section, a set of validation benchmarks is considered to verify accuracy and performance of the framework. Since the basic numerical algorithms are well-established, we test all the cases at high resolution to demonstrate convergence. The dynamic property of the CVP method and the graph-based communication strategy is discussed. All cases in this section and in Section 5 are simulated on the facilities provided by Leibniz-Rechenzentrum (LRZ).

4.1. 1D and 2D shock wave problems

The first four cases involve shock waves and verify the accuracy of the solver and the capability of the current framework to handle problems with arbitrary resolution jumps.

One-dimensional shock tube problems are well-suited for verifying the shock-capturing capability of the solver. Two typical cases, i.e. the Sod problem [51] and the Lax problem [52], are simulated here ($\gamma = 1.4$).

The initial condition for the Sod problem is

$$(\rho, u, p) = \begin{cases} (1, 0, 1) & \text{if } -0.5 \leqslant x < 0 \\ (0.125, 0, 0.1) & \text{if } 0 \leqslant x < 0.5. \end{cases} \tag{17}$$

Simulation time is 0.2 s. A total number of 9000 particles is used. The results are shown in Fig. 4. Both results agree well with the theoretical solution. Slight oscillations can be observed near the contact discontinuity when using the Ducowicz solver. The combination of the LLF solver and second order reconstruction produces a slight overshoot at the expansion wave.

The initial condition for the Lax problem is

$$(\rho, u, p) = \begin{cases} (0.445, 0.698, 3.528) & \text{if } -0.5 \leqslant x < 0 \\ (0.5, 0, 0.5710) & \text{if } 0 \leqslant x < 0.5. \end{cases} \tag{18}$$

Simulation time is 0.14 s. 8000 particles with equal mass are used in this simulation. Fig. 5 gives the density and pressure distribution of the result at 0.14 s. Both results fit well with the "Exact" solution. It is also observed that using second order reconstruction results in less dispassion while generating more oscillations at the contact discontinuity.

Second, two types of two-dimensional circular blast wave problems, i.e. blast wave problem EP1 and EP2 [53], are simulated to validate our framework in high dimensions ($\gamma = 1.4$).

The initial condition for blast wave problem EP1 is

$$(\rho, u, v, p)(\mathbf{r}, 0) = \begin{cases} (1.0, 0, 0, 1.0) & \text{if } \|\mathbf{r}\| \leqslant 0.5 \\ (0.125, 0, 0, 0.1) & \text{otherwise.} \end{cases} \tag{19}$$

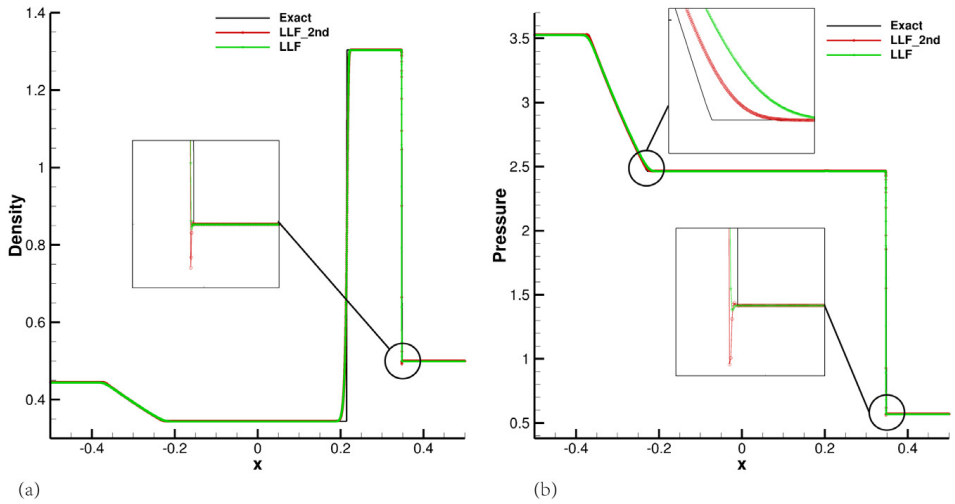


Fig. 5. Lax problem: (a) Density and (b) Pressure distribution at 0.14 s. "Exact" denotes the theoretical solution, and "2nd" represents the second order reconstruction.

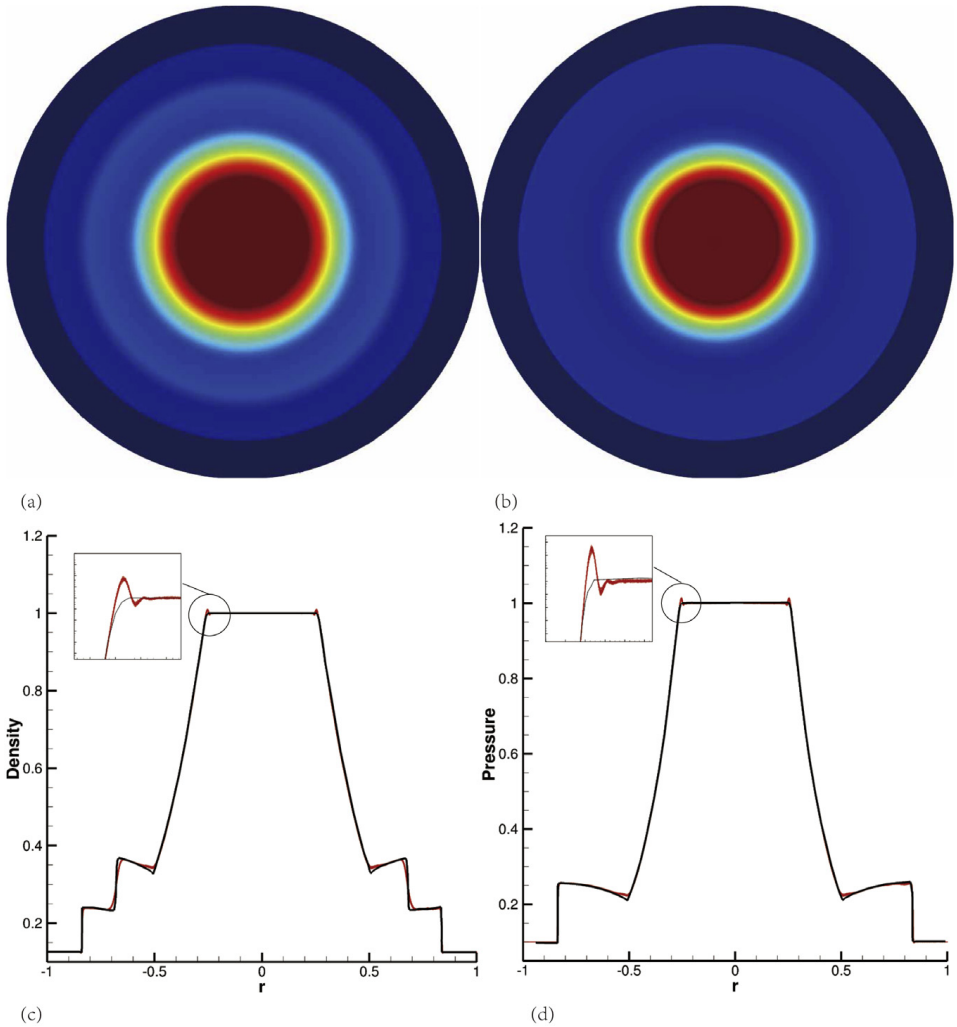


Fig. 6. Circular blast wave problem EP1: (a) Density and (b) Pressure results at $t = 0.2$ s. The black line denotes reference solution, and red dot is the result calculated using LLF solver and second order reconstruction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

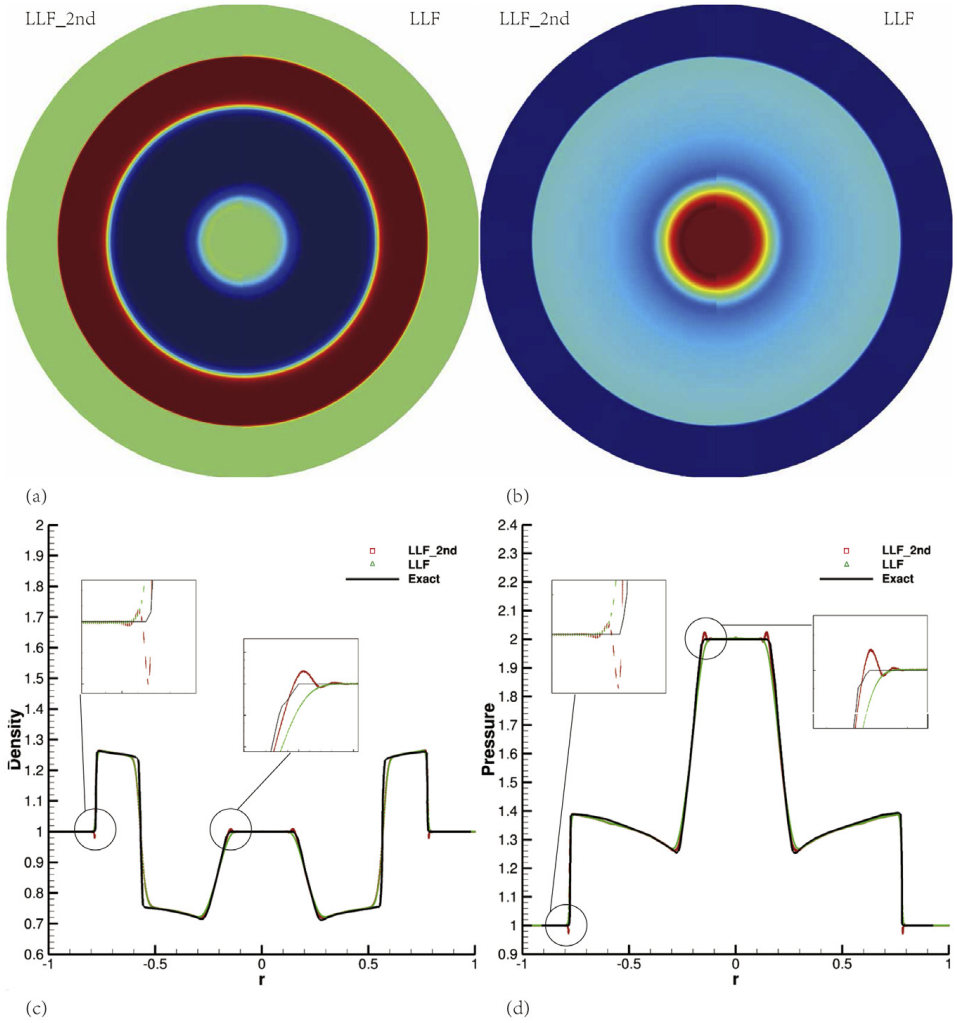


Fig. 7. Circular blast wave problem EP2: (a) Density and (b) Pressure results at $t = 0.2$ s. The black line denotes reference solution. The green/red dot is the result calculated using LLF solver and first/second order reconstruction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Simulation time is 0.2 s. A total number of 5 873 447 particles are simulated. LLF solver plus second order reconstruction is employed and the result is illustrated in Fig. 6 comparing with reference solution. The overall result is satisfactory even though a slight oscillation appears at the front of the expansion wave.

The initial condition for blast wave problem EP2 is

$$(\rho, u, v, p)(\mathbf{r}, 0) = \begin{cases} (1.0, 0, 0, 2.0) & \text{if } \|\mathbf{r}\| \leqslant 0.5 \\ (1.0, 0, 0, 1.0) & \text{otherwise.} \end{cases} \tag{20}$$

Simulation time is 0.2 s. The model uses 5 768 974 SPH particles. We calculate the problem using the LLF solver with both first and second order reconstruction (Fig. 7). The cloud plots in Fig. 7 give a comparison of both results. The first order reconstruction exhibits more dissipation and less oscillation comparing with high order reconstruction. The same conclusion can be drawn from the density and pressure curves at the bottom of Fig. 7.

4.2. The 2D Noh problem

The 2D Noh problem [54] is considered ($\gamma = 5/3$). This is a challenging case to test shock-capturing algorithms, where a uniform inwards radial inflow generates an infinitely strong shock wave moving outwards at a constant speed. The computational domain is a cylinder with radius $\mathbf{r} = 1$, and the initial condition is described as $(\rho, u, v, p) = (1, -x/r, -y/r, 10^{-6})$. The simulation time is 0.6 s. The initial particle distribution is generated using

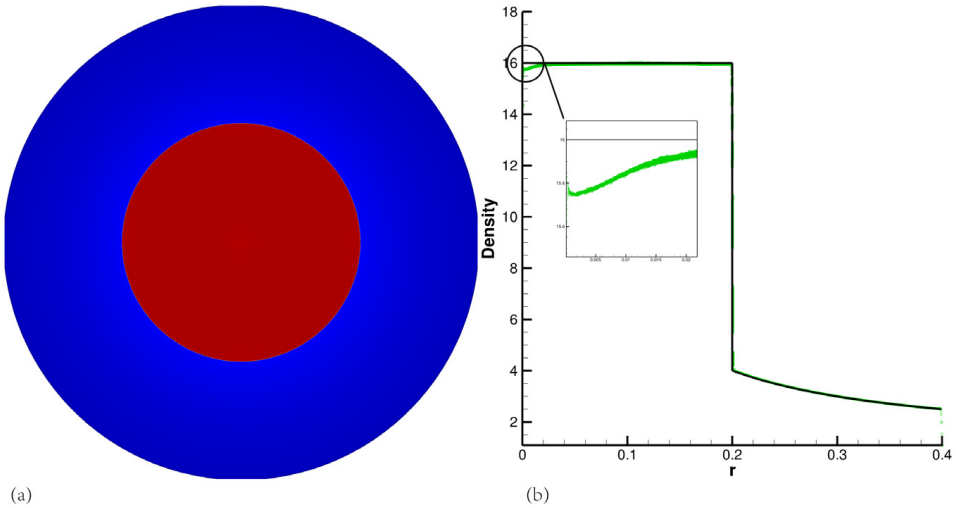


Fig. 8. The 2D Noh problem: (a) Density cloud from simulation result and (b) Density distribution (green dot) comparing with theoretical solution (black line). Numerical result is calculated using LLF solver plus first order reconstruction.

n concentric circles initialized from the center following Ref. [55]. The total number of particles is 19 226 200. This case is also utilized to study the dynamical characteristics of the CVP method and the graph based communication strategy. A total number of 112 MPI tasks each containing 1 TBB thread are launched. The total runtime is 79 427 s (2×10^{-7} second/timestep/particle).

As demonstrated in Fig. 8, an exactly cylindrical shock is generated and the position is recovered accurately. The shock strength agrees with theoretical solution well, and a slight wall heating (with minimum value of 15.6) appears at the center.

The evolution of the load balance error, Fig. 9(c), indicates that the system is repartitioned 11 times during the entire simulation with current threshold (10%), and the imbalance due to computational load change is more pronounced than communication error. Both $E_{Lc,max}$ and $E_{L,max}$ decrease to zero after each repartitioning and keep climbing towards the threshold until the CVP method is triggered again. High locality is observed in Fig. 9(a)(b) as the converging flow moving towards the center, and the partitioning topology remains highly analogous. Benefiting from the CVP method, the communication topology is approximately constant. The total number of edges in the communication graph is around 300 (Fig. 9(d)), which is approximately three times the number of MPI tasks. Moreover, after applying edge coloring, the communication frequency is reduced to 9, which is much smaller than the edge number.

4.3. Rayleigh–Taylor instability

We consider the inviscid Rayleigh–Taylor instability case proposed in Ref. [56]. The initial condition is

$$(\rho, u, v, p) = \begin{cases} (2, 0, 0, 1) & \text{if } y < 0 \\ (2, 0, -0.025c\ cos(8\pi x), 1 + 2y) & \text{if } 0 \le y < 1/2 \\ (1, 0, -0.025c\ cos(8\pi x), y + 3/2) & \text{if } 1/2 \le y < 1 \\ (1, 0, 0, 2.5) & \text{if } 1 \le y, \end{cases} \tag{21}$$

where $c = \sqrt{\gamma \frac{p}{\rho}}$ (with $\gamma = 5/3$) is the sound speed. The computational domain is $[0, 1] \times [-0.05, 1.05]$ and the mesh is generated on lattice vertices. Reflective boundary conditions are used on the left and right, and primitive variables are prescribed on the upper ($1 \le y$) and bottom ($y < 0$). The Ducowicz solver and second order reconstruction are adopted. Equal-mass particles are used, and the total number of particles is 5 453 394. 112 MPI tasks each containing 1 TBB thread are initialized. The total runtime is 42 213 s (2.9×10^{-7} second/timestep/particle).

As illustrated in Fig. 10 (top row), the main structures of the instability are well captured from the simulation results at four instants. The bottom row in Fig. 10 shows the distribution of the subdomains. The partitioning topology

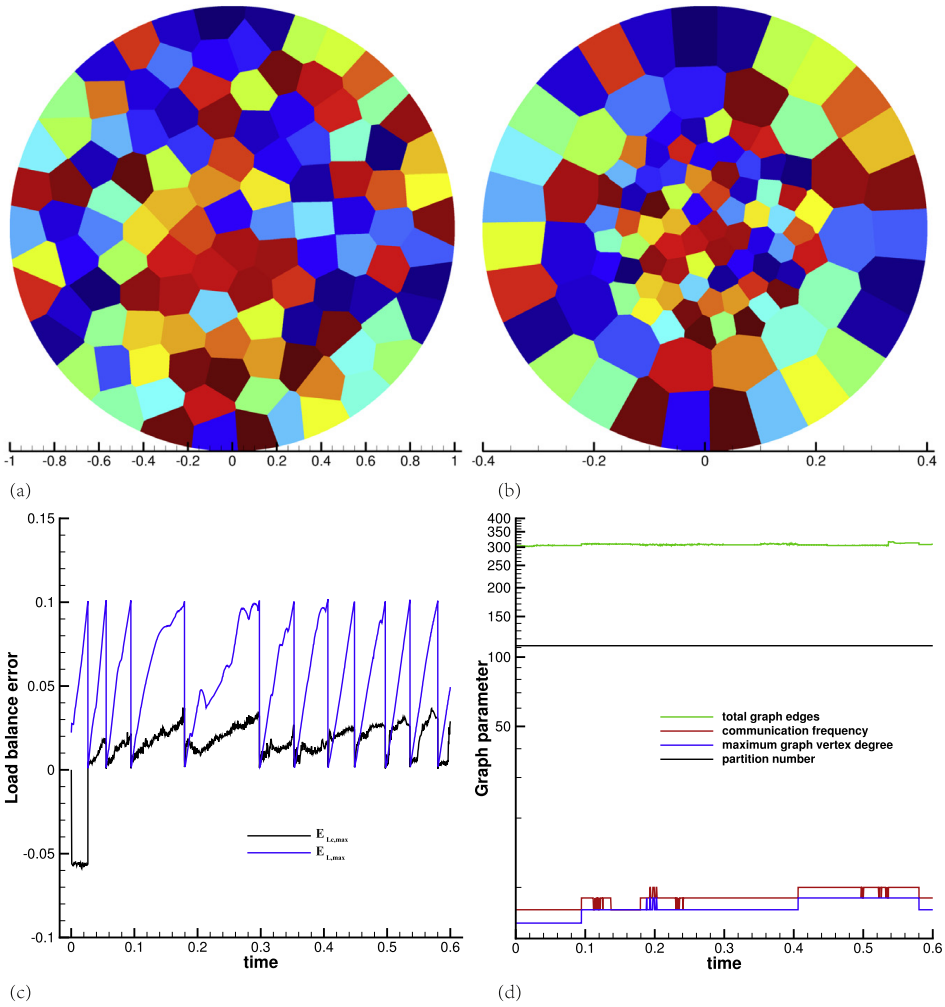


Fig. 9. The 2D Noh problem: (a) CVP partitioning result at $t = 0$ s, each color represents one subdomain. (b) Evolution of the CVP partitioning at $t = 0.6$ s. (c) History of load balance error versus time. (d) History of graph parameters versus time.

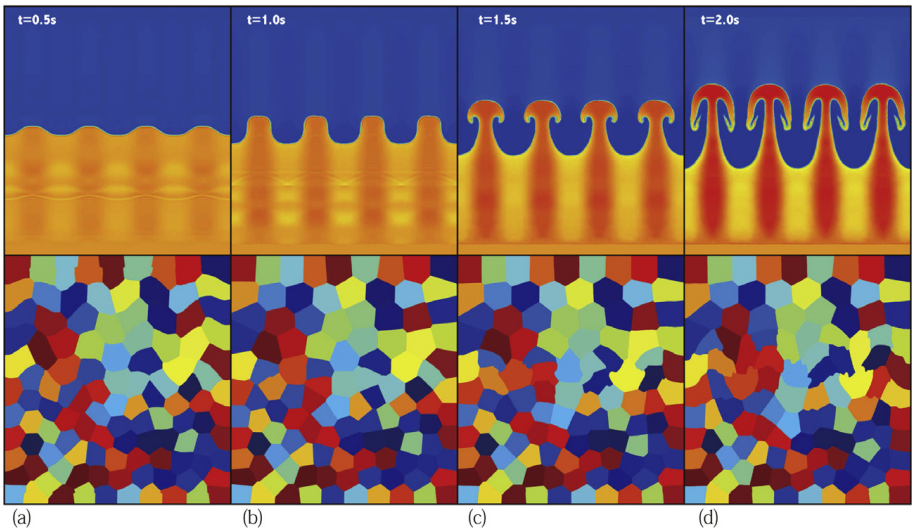


Fig. 10. RT instability problem: Density field (top row) and subdomain distribution (bottom row) at $t = 0.5$ s, 1.0 s, 1.5 s, 2.0 s.

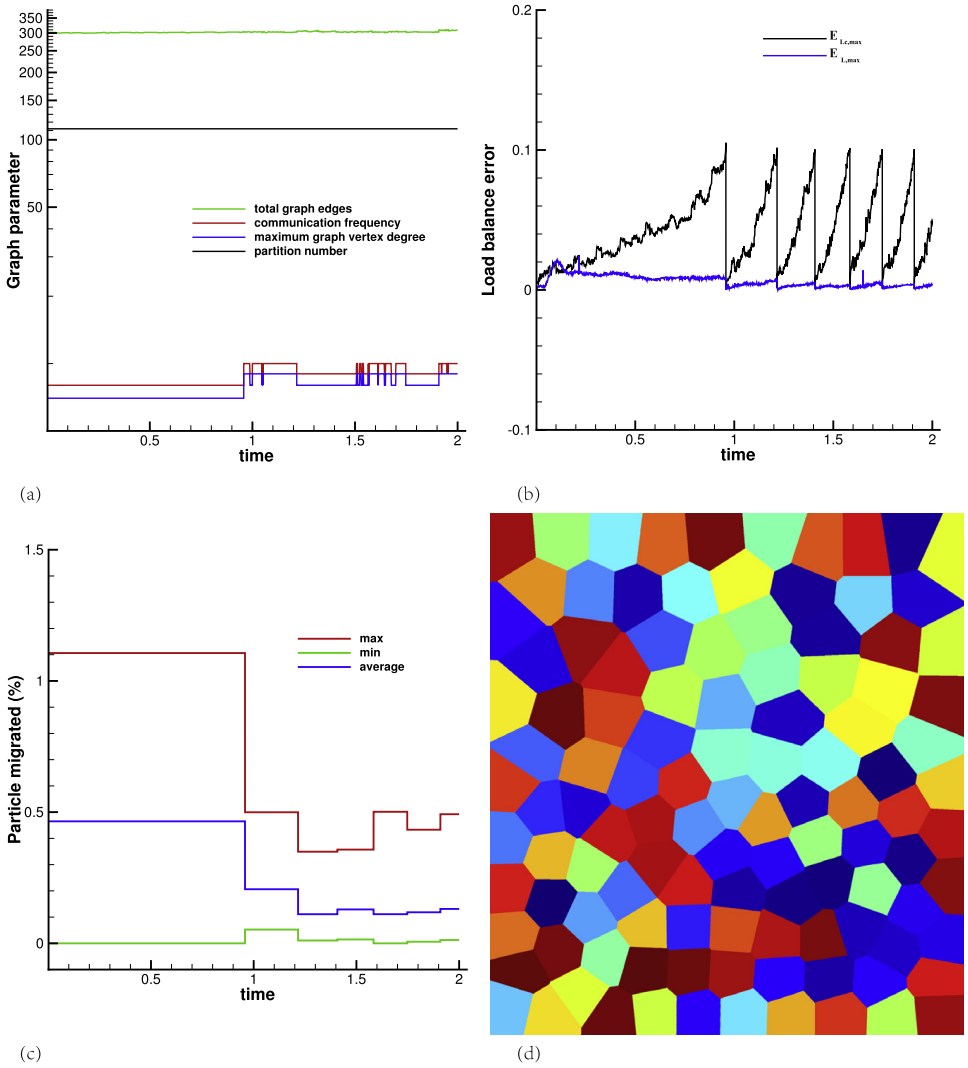


Fig. 11. RT instability problem: (a) History of graph parameters versus time. (b) History of load balance error versus time. (c) Maximum, average and minimum percentage of migrated particles among all subdomains after partitioning. (d) CVP diagram at $t = 0$ s.

remains homogeneous where the flow field shows no strong variation. In the shear regions the topology exhibits larger variations, since the deformation of the subdomains is more pronounced. The total number of communication edges during the simulation is between 310 and 320, and the communication frequency is reduced to 9 or 10 after applying edge coloring method (see Fig. 11(a)). The computational domain is repartitioned 6 times with the maximum error set as 10%. Unlike the 2D Noh problem, $E_{Lc,max}$ grows much faster than $E_{L,max}$ after each repartitioning. Moreover, we calculate the percentage of migrated particles after each CVP partitioning, and show the result in Fig. 11(c). The averaged fraction of migrated particles after second repartition is approximately 15%, indicating that the locality property is well maintained using the current partitioning method.

5. Performance and discussions

In this section, the performance of our framework is discussed. First, the dynamic load balancing strategy proposed in Section 2.2 is assessed with respect to load balance in each subdomain. Then the scalability is calculated with two scaled-size problems in three dimension. Last, we measure the runtime of each module in our framework to validate the optimization strategies presented before.

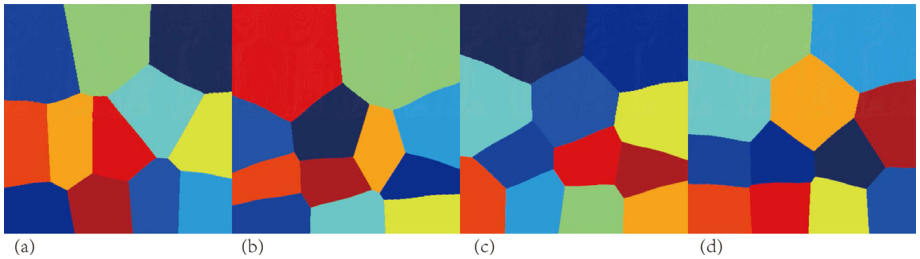


Fig. 12. Distribution of each subdomain with different parameters: (a) Partition with $l_j = \bar{l}_{j,CF}$. (b) Partition with $l_i = \bar{l}_{j,NS}$. (c) Partition with $\epsilon_0 = 0.5$ in Eq. (6). (d) Partition with $\epsilon_0 = 0.1$ in Eq. (6).

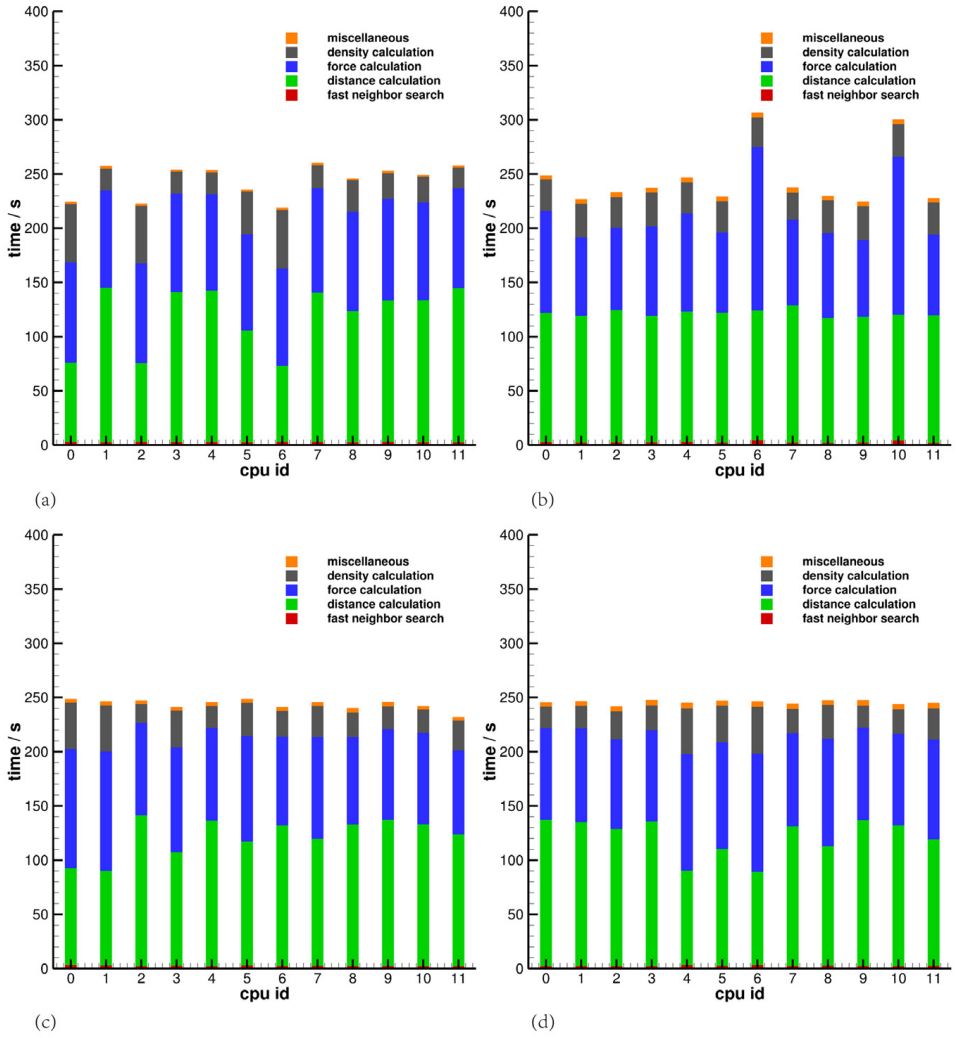


Fig. 13. Measure of elapse time of different subroutines with different criterion: (a) Partition with $l_j = \bar{l}_{j,CF}$. (b) Partition with $l_j = \bar{l}_{j,NS}$. (c) Partition with $\epsilon_0 = 0.5$ in Eq. (6). (d) Partition with $\epsilon_0 = 0.1$ in Eq. (6).

5.1. Dynamic load balancing strategy

We use the Rayleigh–Taylor instability to test the performance of the dynamic partitioning strategy. Since there is an initial resolution jump at $y = 0.5$, and the computational load for each particle may be different for both fluids, the

test case is well suited for our purpose. The case description is the same as Section 4.3 except that the particle number is altered to 134 888. We partition the domain into 12 subdomains, i.e. 12 MPI tasks, and each task has 1 TBB thread. We measure the runtime of all "non-communication-related" subroutines, e.g. simulation, fast neighbor search, and etc., to evaluate the load balance.

Four numerical experiments are set up to verify the performance. In the first two experiments, we set $l_j = \bar{l}_{j,CF}$ and $l_j = \bar{l}_{j,NS}$ respectively to account for only one hot spot of the code. In the last two tests, we employ Eq. (6) to calculate the computational load for each particle. Two initial values, i.e. $\epsilon_0 = 0.5$ and 0.1, are set for the adaptive weight ϵ for the sake of comparison. Different diagrams are obtained with respect to different parameters (see Fig. 12). It is observed that the results from test 1 and 2 differ significantly, and Voronoi cells are more condensed towards the bottom of the domain in Fig. 12(b). The result of test 3 and 4, Fig. 12(c) and (d), however, exhibit an intermediate subdomain distribution.

The runtime measurement, plotted in Fig. 13, presents more comprehensive statistics to evaluate the performance. For test 1, the time fractions contributing to force and density calculations are approximately well-balanced, since by the employed density evaluation the neighbor number for each particle does not vary much among neighboring particles [5]. However, the time fraction of pairwise distance calculation is unbalanced since the computational load is not considered. For test 2, the time fraction for the pairwise distance calculation is balanced, but the others are not. The last two tests, Fig. 13(c) and (d), present overall much better results by merging both factors. The fraction for each subroutine is not equal among all subdomains, but the total runtime is optimized and global load balance is achieved. Moreover, only marginal differences can be observed upon comparing Fig. 13(c) and (d), indicating that the adaptation strategy can dynamically optimize the load balance without being sensitive to the initial value.

As mentioned before, different solvers or test cases may require different criteria. For the current formulation, Eq. (6) is desirable for achieving dynamic load balance. The extra effort to calculate the imbalance error is trivial, as the error is checked every 10–20 iterations.

5.2. Scalability

Two 3D Rayleigh–Taylor instability tests are considered in this section to evaluate the scalability of the framework. Weak scaling S, strong scaling s and efficiency e are calculated from

$$S(N_{proc}) = \frac{t(1)}{t(N_{proc})} \cdot \frac{N(N_{proc})}{N(1)}, \tag{22}$$

$$s(N_{proc}) = \frac{t(1)}{t(N_{proc})}, \tag{23}$$

$$e(N_{proc}) = \frac{S(N_{proc})}{N_{proc}} \text{ or } \frac{s(N_{proc})}{N_{proc}}, \tag{24}$$

where $N(N_{proc})$ and $t(N_{proc})$ are problem size and runtime on N_{proc} nodes, and $N(1)$ and $t(1)$ are problem size and runtime on 1 node.

Both uniform and non-uniform particle distribution are tested to evaluate the scalability of the framework. For uniform particle distribution, particles are assigned with constant scale so that particle mass is different; For non-uniform particle distribution, we initialize particles with constant mass to vary particle scale.

We test the weak scaling with increased domain size to ensure that each MPI task possesses the same number of particles. Two scales, 1M particles and 4M particles per MPI task, are considered to evaluate the performance. Within each MPI task, 7 and 14 TBB threads are initialized respectively for each scale.

The CVP partitioning results for both tests with particle scale 4M/MPI are shown in Fig. 14. The Voronoi cells for both tests feature high-level compact shape. Whereas the partitioning diagrams are significantly different between uniform and non-uniform particle distribution. Weak scaling and efficiency are calculated and plotted in Fig. 15. The maximum number of cores utilized for particle scale 1M/MPI and 4M/MPI are 1792 and 3584 respectively. Our framework exhibits good scalability for both cases. Meanwhile, very similar performances are achieved between the uniform and non-uniform particle distribution, which demonstrates that current framework can handle adaptive resolution efficiently. Moreover, when the number of particles in each MPI task increases, the parallel efficiency raises

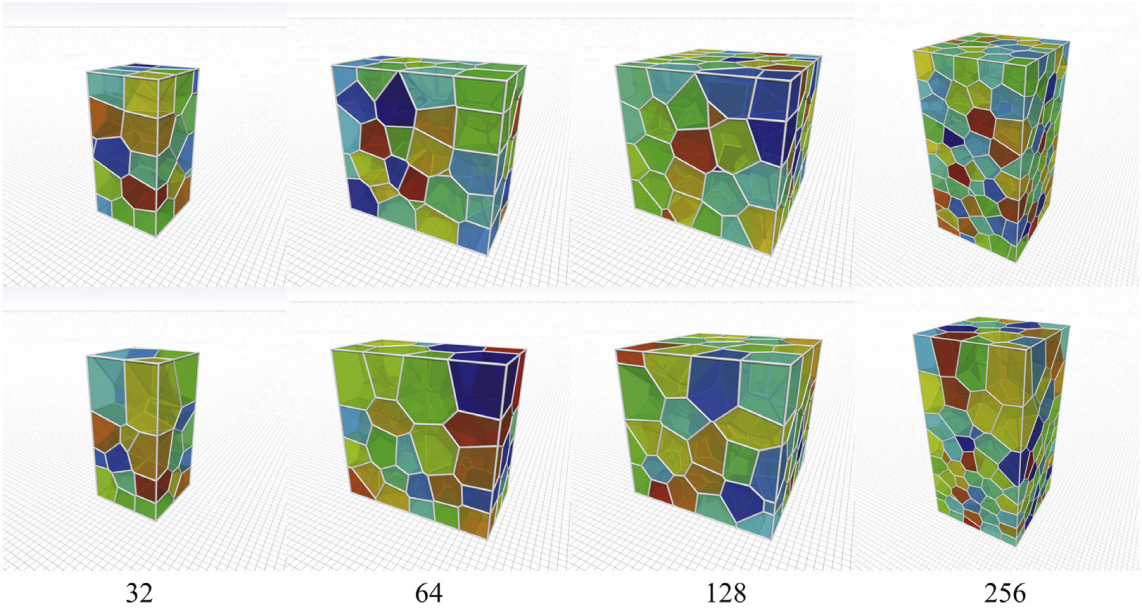


Fig. 14. CVP partitioning result for the uniform (top row) and non-uniform (bottom row) particle distribution with different (32, 64, 128, 256) partitions with scale 4M/MPI.

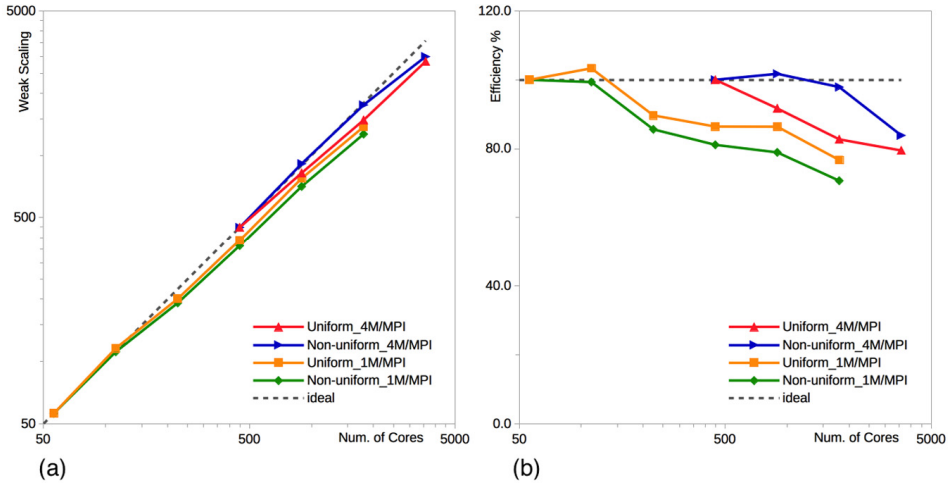


Fig. 15. (a) Weak scaling and (b) Efficiency for uniform and non-uniform particle distributions with scale 1M/MPI and 4M/MPI.

accordingly. An efficiency of 79.5% and 83.7% are achieved for test 1 and test 2 with particle number 4M/MPI when 3584 threads are utilized.

The strong scaling is tested with 6 scales, i.e. 16, 32, 64, 128, 256, 512 million particles. The strong scaling and efficiency for both uniform and non-uniform particle distributions are plotted in Fig. 16. The maximum number of cores utilized is 7168. For all the tests, each MPI task is initialized hosting 7 TBB threads. The results for each scale are calculated from different stating point due to the memory-allocation limitation. Similarly with the weak scaling test, the framework exhibits good scaling for both uniform and non-uniform particle distribution. Due to the limited resources available currently, only three points are tested for the scale of 512M particles. However, the efficiency remains high after increasing the number of cores four times (103% and 97% for uniform and non-uniform particle distributions respectively), and no remarkable decrease is observed.

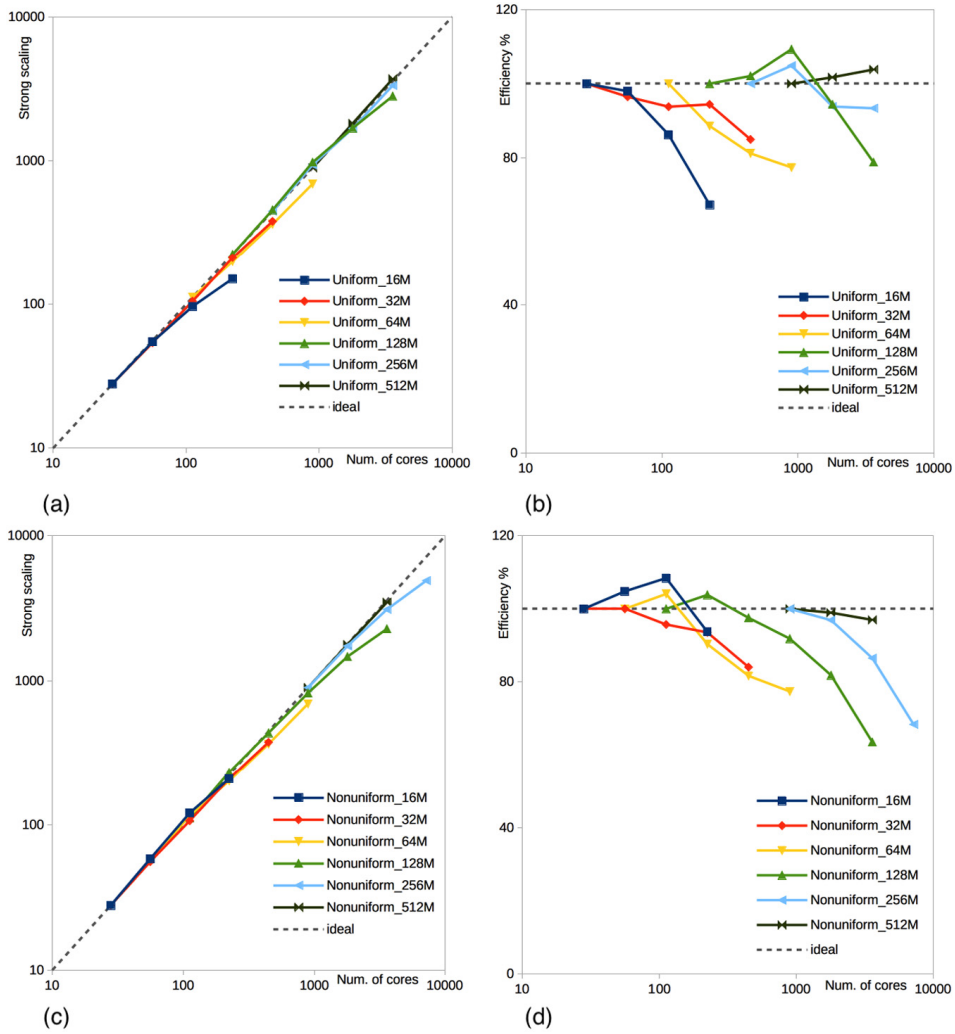


Fig. 16. (a) Strong scaling and (b) Efficiency for uniform particle distribution in different scales. (c) Strong scaling and (d) Efficiency for non-uniform particle distribution in different scales.

5.3. Runtime analysis

The elapsed time of main subroutines in our framework are measured and listed in Table 1 for non-uniform particle distribution with particle number 1M/MPI in weak scaling test. Several observations can be made from this table: (1) The majority of time are dedicated to particle evolution, and the fraction decreases as the number of threads increases. (2) The graph construction takes negligible time (less than 1% of total runtime) owing to the optimization strategy of the diffused graph. (3) The parallel fast neighbor search is highly efficient (less than 0.5% of total runtime). Moreover, due to the local tree construction, the runtime is almost invariant with increasing number of partitions. (4) The time for the CVP partition as well as the particle migration are negligible. (5) The time consumption of communication between neighboring subdomains ranges from 8.03% to 15.25% as the MPI task number varies from 8 to 256.

The measured runtime (second per particle per iteration) for fixed-scale (128M) non-uniform particle distribution is listed in Table 2. The same setup with the strong scaling test is used . The simulation time decreases significantly as the number of CPU cores increases. The total runtime for evolving one particle one timestep is on the order of 10^{-7} second using 3584 CPU cores.

Table 1
Measurement of elapsed time for scaled-size (1M/MPI) non-uniform problem.

Thread number	Graph construction	CVP partition	Graph based communication	Particle migration	Simulation	Fast neighbor search
56	0.07	1.06	8.03	0.48	90.37	0.43
112	0.11	1.21	10.28	0.35	88.05	0.39
224	0.13	1.39	13.21	0.50	84.77	0.42
448	0.16	2.79	12.33	0.69	84.03	0.43
896	0.23	2.91	15.25	0.65	80.96	0.43
1792	0.35	3.18	14.74	0.97	80.76	0.42

Table 2
Measurement of wall-clock time for fixed-scale (128M) non-uniform problem.

Thread number	112	224	448	896	1792	3584
Wall-clock time (10^{-7} s/particle/timestep)	31.1	15.0	7.98	4.28	2.38	1.53

6. Conclusion

In this paper we propose a new parallel framework designed for the SPH method with adaptive smoothing-length. Several new algorithms from our previous work are employed. A set of techniques are proposed to incorporate various functionalities into the framework and to overcome the bottlenecks of efficiency and memory simultaneously. The main contributions are summarized as follows:

1. The framework is able to handle problems with arbitrarily adaptive smoothing-length.

2. An adaptive rebalancing criterion and monitoring system is proposed to integrate the CVP partitioning method as a rebalancer. A set of numerical experiments show that the target of dynamic load balance is achieved with the proposed rebalancing strategy.

3. A localized nested hierarchical data structure is developed to eliminate the bottleneck of memory overhead. The parallel fast-neighbor-search strategy in [21] is tailored and extended to current data structure. The efficiency is increased further due to the local tree construction.

4. A diffused graph is proposed to improve the efficiency of the graph-based communication strategy. The graph-based communication strategy is employed to handle both the construction of ghost buffer particle and particle migration. Numerical tests and runtime analyses demonstrate that negligible time is required for graph construction comparing to total runtime.

5. The scalability of the framework is measured for both uniform and non-uniform particle distributions. The weak scaling reveals that the code scales well up to at least 3584 cores. Good efficiency is achieved for strong scaling tests at various scales.

Although the scalability of the current framework is demonstrated with the performance tests shown above, there are still several places where the code needs to be explored and improved in the future in terms of performance, e.g. adaptive time-stepping schemes, vectorization, task scheduling, pipeline parallelism, and etc. Meanwhile, more particle-based methods, e.g. Incompressible SPH, Molecular Dynamics and Dissipative Particle Dynamics, will be included in future versions to extend the capability of current framework. Moreover, we are also working on improving the portability and work flow in order to facilitate the users for better experience. The code will be available in the future for academic usage once the package is fully tested.

Acknowledgments

The first author is partially supported by China Scholarship Council (No. 201506290038). The second author is partially supported by China Scholarship Council (No. 201206290022). The first author would like to acknowledge Mr. Jiayu Liu for assistance in the scalability tests. The computational resources are provided by Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, Munchen (LRZ).

Appendix. Flowchart of multi-resolution parallel framework

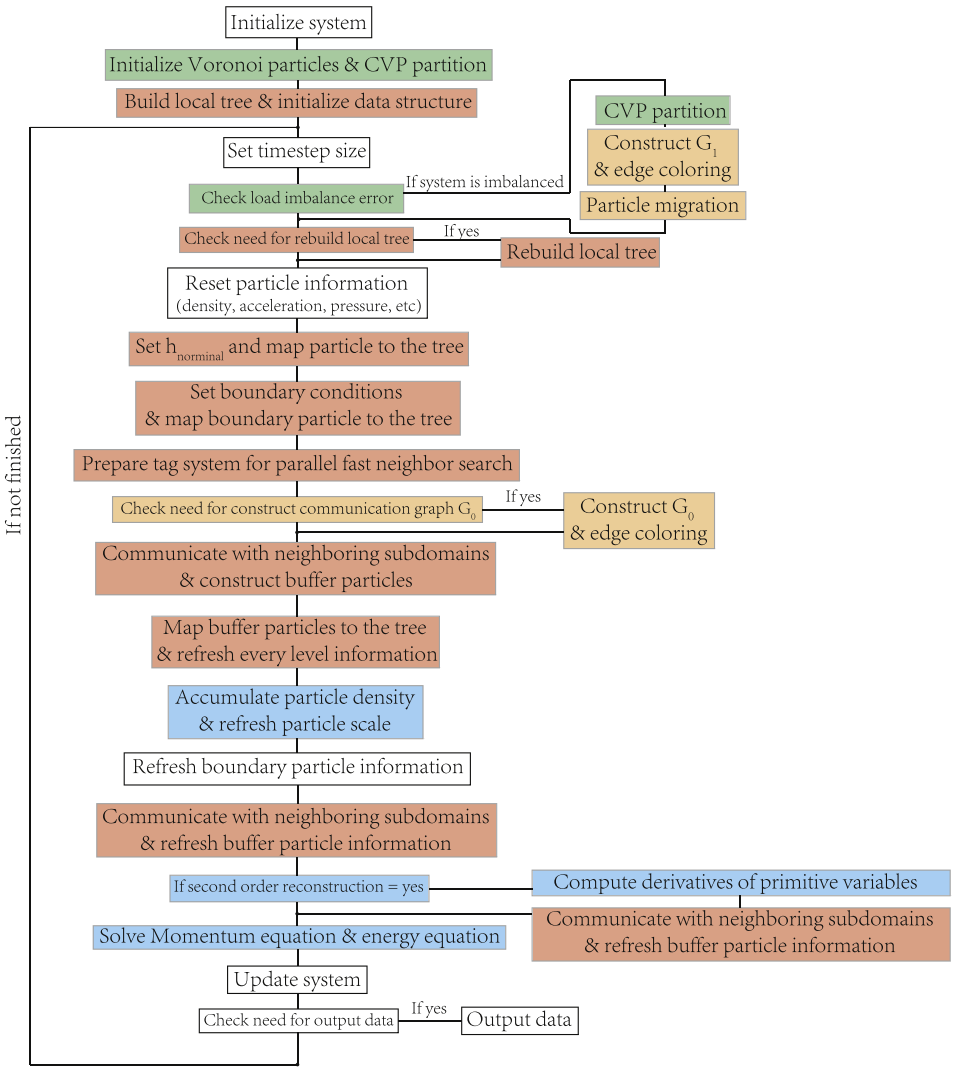


Fig. 17. Flowchart of multi-resolution parallel framework.

References

[1] R.a. Gingold, J. Monaghan, Smoothed particle hydrodynamics: theory and application to non-spherical stars, Mon. Not. R. Astron. Soc. (1977) 375–389, http://dx.doi.org/10.1093/mnras/181.3.375.
[2] J.J. Monaghan, Smoothed particle hydrodynamics, Rep. Progr. Phys. 68 (2005) 1703–1759, http://dx.doi.org/10.1088/0034-4885/68/8/R01. arXiv:0507472v1.
[3] C.-Y. Hu, T. Naab, S. Walch, B.P. Moster, L. Oser, SPHGal: smoothed particle hydrodynamics with improved accuracy for galaxy simulations, Mon. Not. R. Astron. Soc. 443 (2) (2014) 1173–1191.
[4] V. Springel, The cosmological simulation code GADGET-2, Mon. Not. R. Astron. Soc. 364 (4) (2005) 1105–1134.
[5] P.F. Hopkins, A new class of accurate, mesh-free hydrodynamic simulation methods, Mon. Not. R. Astron. Soc. 450 (1) (2015) 53–110.
[6] V. Springel, Smoothed particle hydrodynamics in astrophysics, arXiv preprint arXiv:1109.2219.
[7] D.A. Barcarolo, D. Le Touzé, G. Oger, F. De Vuyst, Adaptive particle refinement and derefinement applied to the smoothed particle hydrodynamics method, J. Comput. Phys. 273 (2014) 640–657.
[8] R. Vacondio, B. Rogers, P. Stansby, P. Mignosa, Variable resolution for SPH in three dimensions: towards optimal splitting and coalescing for dynamic adaptivity, Comput. Methods Appl. Mech. Engrg. 300 (2016) 442–460.
[9] R. Vacondio, B. Rogers, P. Stansby, P. Mignosa, J. Feldman, Variable resolution for SPH: a dynamic particle coalescing and splitting scheme, Comput. Methods Appl. Mech. Engrg. 256 (2013) 132–148.

[10] A.J. Crespo, J.M. Domínguez, B.D. Rogers, M. Gómez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, O. García-Feal, DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH), Comput. Phys. Comm. 187 (2015) 204–216.
[11] O.A.R. Board, OpenMP Application Program Interface Vsersion 3.0 (may 2008) URL http://www.openmp.org/mp-documents/spec30.pdf.
[12] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable parallel programming with CUDA, Queue 6 (2) (2008) 40–53, http://dx.doi.org/10.1145/1365490.1365500. URL http://doi.acm.org/10.1145/1365490.1365500.
[13] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1) (1995) 1–19.
[14] M.P. Forum, MPI: A Message-Passing Interface Standard, Tech. rep., Knoxville, TN, USA (1994).
[15] J.V. Reynders, P.J. Hinker, J.C. Cummings, S.R. Atlas, S. Banerjee, W.F. Humphrey, S.R. Karmesin, K. Keahey, M. Srikant, M.D. Tholburn, et al., POOMA: A framework for scientific simulations on parallel architectures, in: Parallel Programming in C+, 1996, pp. 547–588.
[16] I.F. Sbalzarini, J.H. Walther, M. Bergdorf, S.E. Hieber, E.M. Kotsalis, P. Koumoutsakos, PPM–A highly efficient parallel particle–mesh library for the simulation of continuum systems, J. Comput. Phys. 215 (2) (2006) 566–588.
[17] P. Incardona, A. Leo, Y. Zaluzhnyi, R. Ramaswamy, I.F. Sbalzarini, OpenFPM: A scalable open framework for particle and particle-mesh codes on parallel computers, arXiv preprint arXiv:1804.07598.
[18] O. Awile, F. Büyükkeçeci, S. Reboux, I.F. Sbalzarini, Fast neighbor lists for adaptive-resolution particle simulations, Comput. Phys. Comm. 183 (5) (2012) 1073–1081.
[19] S. Reboux, B. Schrader, I.F. Sbalzarini, A self-organizing lagrangian particle method for adaptive-resolution advection-diffusion simulations, J. Comput. Phys. 231 (9) (2012) 3623–3646.
[20] L. Fu, X.Y. Hu, N.A. Adams, A physics-motivated Centroidal Voronoi Particle domain decomposition method, J. Comput. Phys. 335 (2017) 718–735.
[21] L. Fu, Z. Ji, X.Y. Hu, N.A. Adams, A parallel fast neighbor search and communication strategy for particle-based methods with adaptive smoothing-length, submitted for publication (2018).
[22] L. Fu, S. Litvinov, X.Y. Hu, N.A. Adams, A novel partitioning method for block-structured adaptive meshes, J. Comput. Phys. 341 (2017) 447–473.
[23] E. Boman, K. Devine, R. Heaphy, B. Hendrickson, V. Leung, L.A. Riesen, C. Vaughan, U. Catalyurek, D. Bozdag, W. Mitchell, et al. Zoltan 3.0: parallel partitioning, load balancing, and data-management services; users guide. Sandia National Laboratories, Albuquerque, Rep. SAND2007-4748W.
[24] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1) (1998) 359–392.
[25] A. Okabe, B. Boots, K. Sugihara, S.N. Chiu, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, Vol.. 501, John Wiley & Sons, 2009.
[26] T. Hoefler, J.L. Traff, Sparse collective operations for MPI, in: IEEE International Symposium on Parallel & Distributed Processing, 2009. IPDPS 2009, IEEE, 2009, pp. 1–8.
[27] P. Balaji, D. Buntinas, D. Goodell, W. Gropp, S. Kumar, E. Lusk, R. Thakur, J.L. Träff, Mpi on a million processors, in: European Parallel Virtual Machine/Message Passing Interface Users Group Meeting, Springer, 2009, pp. 20–30.
[28] A. Ovcharenko, D. Ibanez, F. Delalondre, O. Sahni, K.E. Jansen, C.D. Carothers, M.S. Shephard, Neighborhood communication paradigm to increase scalability in large-scale dynamic scientific applications, Parallel Comput. 38 (3) (2012) 140–156.
[29] G.L. Djordjević, M.B. Tošić, A heuristic for scheduling task graphs with communication delays onto multiprocessors, Parallel Comput. 22 (9) (1996) 1197–1214.
[30] D. Durand, R. Jain, D. Tseytlin, Parallel I/O scheduling using randomized, distributed edge coloring algorithms, J. Parallel Distrib. Comput. 63 (6) (2003) 611–618.
[31] Q. Du, V. Faber, M. Gunzburger, Centroidal Voronoi Tessellations: applications and algorithms, SIAM Rev. 41 (4) (1999) 637–676.
[32] M. Iri, K. Murota, T. Ohya, A fast Voronoi-diagram algorithm with applications to geographical optimization problems, in: System Modelling and Optimization, Springer, 1984, pp. 273–288.
[33] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, C. Yang, On Centroidal Voronoi Tessellationenergy smoothness and fast computation, ACM Trans. Graph. (ToG) 28 (4) (2009) 101.
[34] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[35] Q. Du, M. Emelianenko, L. Ju, Convergence of the Lloyd algorithm for computing Centroidal Voronoi Tessellations, SIAM J. Numer. Anal. 44 (1) (2006) 102–119.
[36] G. Contreras, M. Martonosi, Characterizing and improving the performance of intel threading building blocks, in: IEEE International Symposium on Workload Characterization, 2008. IISWC 2008, IEEE, 2008, pp. 57–66.
[37] J. Misra, D. Gries, A constructive proof of Vizing's theorem, Inform. Process. Lett. 41 (3) (1992) 131–133.
[38] J.G. Siek, L.-Q. Lee, A. Lumsdaine, Boost Graph Library: User Guide and Reference Manual, Pearson Education, 2001.
[39] K.D. Devine, E.G. Boman, R.T. Heaphy, B.A. Hendrickson, J.D. Teresco, J. Faik, J.E. Flaherty, L.G. Gervasio, New challenges in dynamic load balancing, Appl. Numer. Math. 52 (2–3) (2005) 133–152.
[40] L. Verlet, Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules, Phys. Rev. 159 (1) (1967) 98.
[41] X. Hu, N.A. Adams, An incompressible multi-phase SPH method, J. Comput. Phys. 227 (1) (2007) 264–278.
[42] J.J. Monaghan, Smoothed particle hydrodynamics, Annu. Rev. Astron. Astrophys. 30 (1992) 543–574.
[43] S. Adami, X. Hu, N.A. Adams, A transport-velocity formulation for smoothed particle hydrodynamics, J. Comput. Phys. 241 (2013) 292–307.
[44] S.-i. Inutsuka, Reformulation of smoothed particle hydrodynamics with Riemann solver, J. Comput. Phys. 179 (1) (2002) 238–267.

[45] G. Murante, S. Borgani, R. Brunino, S.-H. Cha, Hydrodynamic simulations with the Godunov smoothed particle hydrodynamics, Mon. Not. R. Astron. Soc. 417 (1) (2011) 136–153.
[46] K. Puri, P. Ramachandran, Approximate Riemann solvers for the Godunov SPH (GSPH), J. Comput. Phys. 270 (2014) 432–458.
[47] D.J. Price, Smoothed particle hydrodynamics and magnetohydrodynamics, J. Comput. Phys. 231 (3) (2012) 759–794.
[48] B. Van Leer, Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method, J. Comput. Phys. 32 (1) (1979) 101–136.
[49] B. Schäling, The boost C++ libraries, Boris Schäling, 2011.
[50] C. Rycroft, Voro++: A three-dimensional Voronoi cell library in C++, Lawrence Berkeley National Laboratory, 2009.
[51] G.A. Sod, A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws, J. Comput. Phys. 27 (1) (1978) 1–31.
[52] P.D. Lax, Weak solutions of nonlinear hyperbolic equations and their numerical computation, Comm. Pure Appl. Math. 7 (1) (1954) 159–193.
[53] D. Avesani, M. Dumbser, A. Bellin, A new class of Moving-Least-Squares WENO–SPH schemes, J. Comput. Phys. 270 (2014) 278–299.
[54] W.F. Noh, Errors for calculations of strong shocks using an artificial viscosity and an artificial heat flux, J. Comput. Phys. 72 (1) (1987) 78–120.
[55] L.D.G. Sigalotti, H. López, A. Donoso, E. Sira, J. Klapp, A shock-capturing SPH scheme based on adaptive kernel estimation, J. Comput. Phys. 212 (1) (2006) 124–149.
[56] Z. Xu, C.-W. Shu, Anti-diffusive flux corrections for high order finite difference WENO schemes, J. Comput. Phys. 205 (2) (2005) 458–485.

A.2 Paper II

Zhe Ji, Lin Fu, Xiangyu Y. Hu, Nikolaus A. Adams

A Lagrangian Inertial Centroidal Voronoi Particle method for dynamic load balancing in particle-based simulations

In *Computer Physics Communications*, Volume 239, 2019, pp. 53-63, DOI: `https://doi.org/10.1016/j.cpc.2019.01.011`.

Contribution: My contribution to this work was the development of the method and the corresponding computer code for its implementation. I performed simulations and analyzed the results, and wrote the manuscript for the publication.

Computer Physics Communications 239 (2019) 53–63

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

A Lagrangian Inertial Centroidal Voronoi Particle method for dynamic load balancing in particle-based simulations

Zhe Ji, Lin Fu, Xiangyu Y. Hu *, Nikolaus A. Adams

Technical University of Munich, Department of Mechanical Engineering, Chair of Aerodynamics and Fluid Mechanics, 85748 Garching, Germany

ARTICLE INFO

Article history:
Received 14 February 2018
Received in revised form 23 November 2018
Accepted 17 January 2019
Available online 28 January 2019

Keywords:
Inertial Centroidal Voronoi Particle
Centroidal Voronoi Particle
Dynamic load balance
SPH
Particle simulation

ABSTRACT

In this paper we develop a Lagrangian Inertial Centroidal Voronoi Particle (LICVP) method to extend the original CVP method (Fu et al., 2017) to dynamic load balancing in particle-based simulations. Two new concepts are proposed to address the additional problems encountered in repartitioning the system. First, a background velocity is introduced to transport Voronoi particles according to the local fluid field, which facilitates data reuse and lower data redistribution cost during rebalancing. Second, in order to handle problems with skew-aligned computational load and large void space, we develop an inertial-based partitioning strategy, where the inertial matrix is utilized to characterize the load distribution, and to confine the motion of Voronoi particles dynamically adapting to the physical simulation. Intensive numerical tests in fluid dynamics simulations reveal that the underlying LICVP method improves the incremental property remarkably without sacrifices on other objectives, i.e. the inter-processor communication is optimized simultaneously, and the repartitioning procedure is highly efficient.

1. Introduction

Large scale parallel computing is essential for a wide range of scientific applications. The objective of the domain decomposition method is to facilitate the algorithms to harness computational resources more efficiently [1]. In Computational Fluid Dynamics (CFD), the configuration of discretization-elements (mesh or particle) may evolve in time [2], which requires partitioning algorithms to reassign computational load to processors periodically, i.e., achieve dynamic load balance. Compared to static partitioning, rebalancing needs to meet the same targets, e.g. load balance, locality, optimization for inter-processor communication, but also is subject to further constrains. The key aspect for rebalancing schemes is to minimize partitioning modifications subject to small topology changes, i.e. the so-called incremental property, and to optimize the inter-processor communication at smallest cost [3,4]. The reader may refer to [5] and [2] for a comprehensive review regarding recent developments of repartitioning approaches.

For particle simulations, the discretization-elements, e.g. SPH particles, move according to the corresponding dynamic system, which may cause deformation and relocation of computational sub-domains. To achieve the incremental property it is crucial to maximize the data reuse during rebalancing process, namely, the rebalancer should be aware of the details of the underlying system, and repartition the system incorporating the existing partition to achieve lower data redistribution cost [6]. An algorithm with better incremental property features less migration percentage during each rebalancing. Another problem encountered in particle simulation is that in some cases, e.g. the dambreak problem [7], tsunami simulation [8], and the rotating disk problem in astrophysics [9], the computational load distributes anisotropically in the computational domain, and the configuration of discretization-elements varies rapidly. The skew alignment and rapid change of computational load may result in void space which requires zero computing resources. Such a scenario is more problematic for partitioning algorithms with respect to satisfying all constrains simultaneously [3,10].

In our previous paper [11], a CVP domain decomposition method based on physical analogy has been developed. The load balance target is achieved by solving a Voronoi Particle (VP) evolution model equation. Centroidal Voronoi Tessellation (CVT) is utilized for communication reduction by optimizing the compactness of partitioning sub-domains [10]. The CVP method is verified by various static partitioning tests, independently of the mesh-element type. Later, we have integrated the CVP method as dynamic partitioner and develop a new multi-resolution parallel framework for the SPH method [12]. An adaptive rebalancing criterion and monitoring system has been proposed to assess the imbalance during simulation and reassign equivalent load among all the processors.

Although the CVP method inherently features load balance and communication reduction, there is no explicit treatment in our previous work regarding to the handling of the aforementioned

* Corresponding author.
E-mail addresses: zhe.ji@tum.de (Z. Ji), lin.fu@tum.de (L. Fu), xiangyu.hu@tum.de (X.Y. Hu), nikolaus.adams@tum.de (N.A. Adams).

https://doi.org/10.1016/j.cpc.2019.01.011
0010-4655/

additional difficulties that will be encountered in dynamic partitioning. The objective of the current paper is to extend the CVP method and improve the performance of particle-based methods in the dynamic partitioning problems. Two concepts are proposed in this paper: a Lagrangian background velocity and an inertial-based partitioning strategy. The newly developed method is called Lagrangian Inertial CVP (LICVP) method. In the following two paragraphs the basic idea of LICVP method is introduced.

With the CVP method each Voronoi particle possesses physical properties either integrated from meshes enclosed within its Voronoi cell region or calculated from neighboring cells. Generally speaking, the CVP method can be viewed as a coarse-grained model of underlying mesh elements. Based on this observation we can introduce a background velocity to advect partitioning generators, i.e. Voronoi particles, between two consecutive rebalancing steps. If the background velocity is given properly, the positions of the Voronoi particles will be updated following the evolution of the dynamic system and geometry variation, e.g. mass center, of local sub-domains. When the rebalancing subroutine is triggered, the updated positions of Voronoi particle are utilized as input and initial condition for the new partitioning result. Since the equilibrium is calculated globally, the target of load balance and communication reduction is guaranteed by the CVP method. Moreover, since the new partitioning diagram is calculated in awareness of the original partitioning, data reuse is improved, i.e. incremental property can be achieved.

To handle problems with skew-aligned computational load and large void space, we propose another extension of the CVP method. The idea is to constrain the motion of Voronoi particles according to the load distribution in space during the rebalancing procedure. Similarly with the Recursive Inertial Bisection (RIB) method [13], we choose the inertia matrix to characterize the load distribution, such that the skewness of the computational load can be obtained accordingly. A splitting operator is proposed to take the trajectory of Voronoi particles as the input and outputs the resultant vector subject to a certain type of constraint. Moreover, an adaptive filter is proposed, which selects a proper constraint dynamically adapting to the development of the underlying particle system. The filter ensures restoration of the original CVP method when the computational load is distributed homogeneously. With this method Voronoi particles are insensitive of the load variation along the confined direction, thus the convergence and incremental property are improved. We refer the new method as Inertial CVP.

In this paper, we focus on a specific version of particle-based method, the Smoothed Particle Hydrodynamics method. The validation of our new algorithm is performed with the code we have developed previously [12]. The remainder of this paper is arranged as follows. In Section 2 the CVP method and imbalance monitoring strategy for dynamic load balancing is briefly reviewed. The concept and model equations of the Lagrangian Inertial CVP method are introduced in Section 3. Numerical algorithms and boundary conditions are elaborated in Section 4. Section 5 gives five numerical tests to verify our method.

2. Brief review of the CVP method

We briefly review the model equations of the CVP method and the imbalance monitoring strategy from previous work [11,12].

2.1. Model equations

The key concept of the CVP method is to combine CVT [14] and Voronoi Particle dynamics (VP) to achieve high-level compactness of partitioning sub-domains and error-controlled load balance simultaneously. The equilibrium is calculated iteratively utilizing a two-step time integration scheme. The CVT diagram is constructed employing the Lloyd method [15,16]. The model equation for VP is

$$\mathbf{a}_i^{vp} = \frac{d\mathbf{v}_i^{vp}}{dt} = -\frac{\int_{\Omega_i} \nabla p d\sigma}{\int_{\Omega_i} \rho d\sigma} = -\frac{\int_{\partial\Omega_i} p d\mathbf{S}}{m_i^{vp}}, \tag{1}$$

where $\mathbf{a}^{vp}$ denotes the acceleration, $\mathbf{v}^{vp}$ the velocity vector, p the pressure, ρ the density, Ω_i the region corresponding to Voronoi particle i and $\partial\Omega_i$ the Voronoi cell surface. The pressure of the Voronoi particle is defined as

$$p_i^{vp} = \frac{m_i^{vp}}{m_{tg,i}^{vp}}, \tag{2}$$

where $m_{tg,i}^{vp}$ is the target mass. The pressure at the surface between two neighboring cells is computed by second-order approximation $p_{ij}^{vp} = (p_i^{vp} + p_j^{vp})/2$. The scale h_i^{vp} for Voronoi particle i is defined as the average distances from all neighboring particles.

In each iteration substep the Voronoi particles first are moved according to the VP method by

$$\mathbf{x}_i^{vp,*} = \mathbf{x}_i^{vp,n} + \alpha \frac{1}{2} \mathbf{a}_i^{vp,n} \tau_1^2, \tag{3}$$

and then updated following CVT construction as

$$\mathbf{x}_i^{vp,n+1} = \mathbf{x}_i^{vp,*} + (1-\alpha)\tau_2(\mathbf{z}_i^{vp,n} - \mathbf{x}_i^{vp,*}), \tag{4}$$

where τ_1 and τ_2 are pseudo timestep sizes. The relaxation parameter α is set as 0.8.

2.2. Imbalance monitoring strategy

To facilitate dynamic load balancing in the current framework, we develop an imbalance monitoring system. Two criteria are constructed to indicate the imbalance of a computation: (1) imbalance caused by the load change of SPH particles in Ω_i, defined as m_i^{vp}; (2) imbalance due to the change of communication, i.e. the number of ghost buffer particles constructed in Ω_i, defined as mc_i^{vp}.

The computational load for each SPH particle $l_{j,i}^{sp}$ is estimated by considering two key operations, the *neighbor search* (*NS*) and the *calculation of inter-particle forces* (*CF*),

$$l_{j,i}^{sp} = \epsilon \bar{l}_{j,i,NS}^{sp} + (1-\epsilon)\bar{l}_{j,i,CF}^{sp}, \tag{5}$$

where $\bar{l}_{\{\cdot\}}^{sp}$ denotes the normalized computational load. The adaptive weight ϵ is obtained by

$$\epsilon = \frac{\Delta t_{NS}}{\Delta t_{NS} + \Delta t_{CF}}, \tag{6}$$

where $\Delta t_{\{\cdot\}}$ denotes the net runtime elapsed for different subroutines since the last load-balance estimate. The total computational-load for Ω_i is calculated by

$$m_i^{vp} = \int_{\Omega_i} \rho(\mathbf{x}) d\sigma = \sum_{j=0}^{N_i-1} l_{j,i}^{sp}, \tag{7}$$

where N_i is the number of SPH particles included in the Voronoi cell i.

To combine the two criteria, an imbalance monitoring tag R is defined in Eq. (8). The CVP method is triggered when $R = 1$.

$$R = \begin{cases} 1 & \text{if } E_{mc^{vp},max} > e_{mc^{vp},max} \text{ or } E_{m^{vp},max} > e_{m^{vp},max} \\ 0 & \text{if else,} \end{cases} \tag{8}$$

$$E_{mc^{vp},max} = max(E_{mc^{vp},0}, \ldots, E_{mc^{vp},k-1}), \tag{9}$$

$$E_{m^{vp},max} = max(E_{m^{vp},0}, \ldots, E_{m^{vp},k-1}), \tag{10}$$

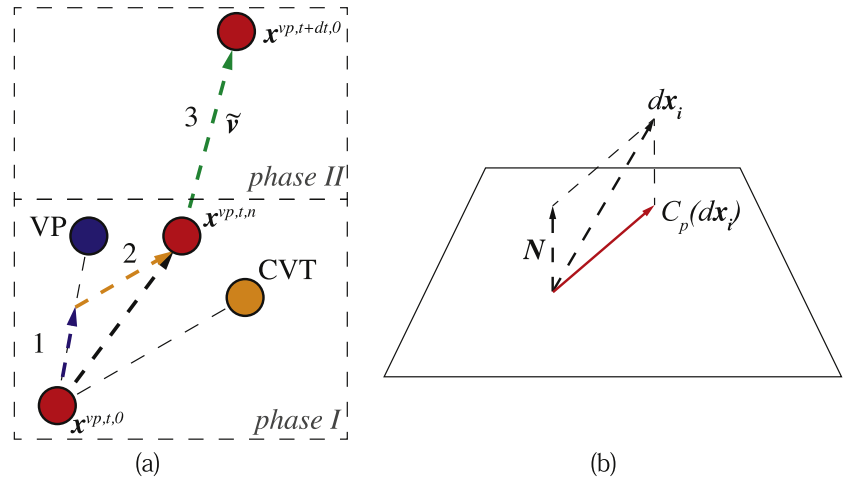


Fig. 1. (a) Demonstration of three-step time integration scheme. (b) Demonstration of the operator $C_p(\mathbf{A})$ if partitioning on the plane $\perp \mathbf{N}$.

where $E_{mc^{vp},i} = \dfrac{mc_i^{vp} - mc_{0,i}^{vp}}{mc_{0,i}^{vp}}$ and $E_{m^{vp},i} = \dfrac{m_i^{vp} - m_{0,i}^{vp}}{m_{0,i}^{vp}}$, with $i = 0, \ldots, k-1$ are local errors in each sub-domain. $mc_{0,i}^{vp}$ and $m_{0,i}^{vp}$ are initial values set after each partitioning. $e_{mc^{vp},max}$ and $e_{m^{vp},max}$ are user defined error tolerance respectively. In current framework, we set $e_{mc^{vp},max} = e_{m^{vp},max} = 0.1$.

3. Lagrangian inertial CVP (LICVP) method

In this section, the detailed formulation of the LICVP method is developed. First, a three-step time integration scheme is proposed by introducing a background velocity for advecting Voronoi particles. The Inertial CVP is described in Section 3.2. A splitting operator and an adaptive filter is defined to optimize and select partitioning strategy dynamically.

3.1. Background velocity for Voronoi particles

We first define two phases in dynamic load balancing using the CVP method, illustrated in Fig. 1(a). The first is the partitioning phase and the second is the load imbalance monitoring phase. In the first phase, the partitioning subroutine is triggered. The initial position of Voronoi particle i at time t, denoted as $\mathbf{x}_i^{vp,t,0}$, is evolved using a two-step integration scheme given by Eq. (3) and (4). The partition operation is terminated when equilibrium is achieved after n pseudo time-steps. The final position of Voronoi particle i is $\mathbf{x}_i^{vp,t,n}$, and the computational domain is repartitioned accordingly. In the second phase, the imbalance monitoring system is launched. The imbalance errors $E_{mc^{vp},i}$ and $E_{m^{vp},i}$ are examined during the computation. The error accumulates until the threshold is reached, i.e. $R = 1$, then phase 1 is activated again. The final position of Voronoi particle i in phase 2 is marked as $\mathbf{x}_i^{vp,t+\Delta t,0}$.

For phase I, we develop a two-step time integration scheme to achieve both load balance and communication reduction targets. To increase the data reuse in rebalancing, we propose a third step to advect Voronoi particles in phase II (see Fig. 1(a)). The third time integration step can be defined as

$$\mathbf{x}_i^{vp,t+\Delta t,0} = \mathbf{x}_i^{vp,t,n} + \tilde{\mathbf{v}}_i^{vp} \cdot \Delta t, \tag{11}$$

where Δt is the timestep size identical to the underlined dynamic system, e.g. the SPH flow model. $\tilde{\mathbf{v}}_i^{vp}$ is a background velocity for Voronoi particle i, which can be given arbitrarily. However, to preserve the incremental property, we suggest that $\tilde{\mathbf{v}}_i^{vp}$ is correlated to the motion of Ω_i.

Several options for calculating $\tilde{\mathbf{v}}_i^{vp}$ can be considered. The first is to simply set the background velocity of Voronoi particle i as identical to the mean fluid velocity within the current subdomain.

$$\tilde{\mathbf{v}}_i^{vp} = \frac{\sum_{j=0}^{N_i-1} \mathbf{v}_{j,i}^{sp}}{N_i}, \tag{12}$$

where $\mathbf{v}_{j,i}^{sp}$ is the velocity of SPH particle j in subdomain Ω_i.

Another option is to consider the influence of computational load distribution. The mass center of Ω_i is given as

$$\mathbf{z}_i^{vp} = \frac{\int_{\Omega_i} \rho(\mathbf{x})\mathbf{x}d\sigma}{\int_{\Omega_i} \rho(\mathbf{x})d\sigma} = \frac{\sum_{j=0}^{N_i-1} l_{j,i}^{sp}\mathbf{x}_{j,i}^{sp}}{m_i^{vp}}, \tag{13}$$

where $d\sigma$ denotes the volume differential, and $\mathbf{x}_{j,i}^{sp}$ the coordinates of specific SPH particle j in Ω_i. Then $\tilde{\mathbf{v}}_i^{vp}$ can be defined by the time differential of mass center,

$$\tilde{\mathbf{v}}_i^{vp} = \frac{d\mathbf{z}_i^{vp}}{dt} = \frac{\sum_{j=0}^{N_i-1} l_{j,i}^{sp}\mathbf{v}_{j,i}^{sp}}{m_i^{vp}}. \tag{14}$$

In practice, instead of updating the Voronoi particles according to the velocity of the mass center of every timestep, we simply set the mass center as the position of Voronoi particles before repartitioning.

We consider the aforementioned two choices in this paper. The performance and detailed comparison will be given in Section 5.

3.2. Inertial CVP

In order to constrain the motion of a Voronoi particle, we define a splitting operator, which operates on the time-integration scheme in phase I. We can rewrite Eq. (3) and (4) as

$$\mathbf{x}_i^{vp,n+1} = \mathbf{x}_i^{vp,n} + \Delta\mathbf{x}_i^p, \tag{15}$$

and

$$\Delta\mathbf{x}_i^p = C_p(\Delta\mathbf{x}_i), \tag{16}$$

where $\Delta\mathbf{x}_i^p$ is the resulting displacement. The splitting operator C_p functions via manipulating $\Delta\mathbf{x}$, where $\Delta\mathbf{x}_i = \alpha\dfrac{1}{2}\mathbf{a}_i^{vp,n} \triangle \tau_1^2 + (1-\alpha) \triangle \tau_2(\mathbf{z}_i^{vp,n} - \mathbf{x}_i^{vp,*})$.

For a general input vector $\mathbf{A}$ and a given direction of constraint, C_p returns a resultant vector defined as

$$C_p(\mathbf{A}) = \begin{cases} \mathbf{A}_\parallel, \text{where} \mathbf{A}_\parallel \parallel \mathbf{N} & \text{if constrained along } \mathbf{N} \\ \mathbf{A}_\perp, \text{where} \mathbf{A}_\perp \perp \mathbf{N} & \text{if constrained on the plane } \perp \mathbf{N} \\ \mathbf{A} & \text{if no constraint,} \end{cases} \tag{17}$$

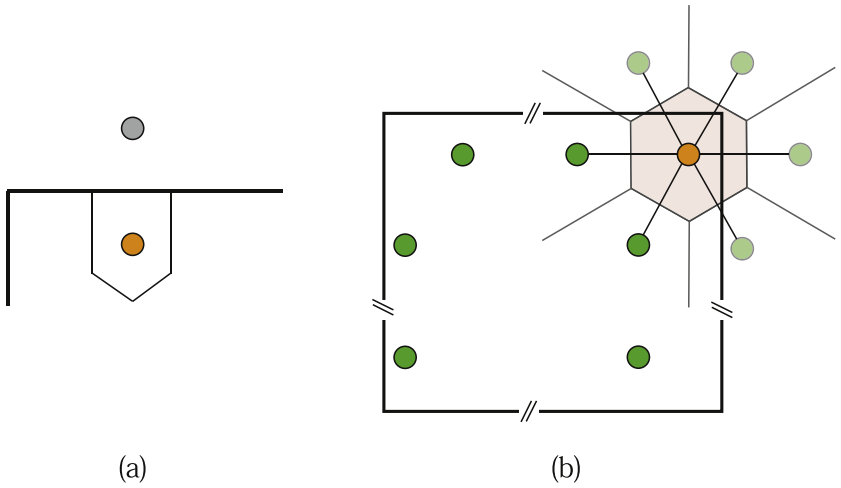


Fig. 2. Sketch of (a) symmetric and (b) periodic boundary conditions. The orange particles indicate the boundary Voronoi particles; the gray particle is the mirrored particles constructed to enforce symmetric boundary condition; the solid green particles are neighbors of the current (orange) Voronoi particle in a periodic box, and the translucent particles are mapped neighbor particles . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where $\mathbf{A}_{\parallel}$ and $\mathbf{A}_{\perp}$ denote the vector parallel and perpendicular to $\mathbf{N}$, respectively. $\mathbf{N}$ is the input that defines the direction of constraint. As illustrated in Fig. 1(b), the particle motion is constrained on a plane where $\mathbf{N}$ denotes its normal direction. In this case, $C_p(\mathbf{A})$ returns $\mathbf{A}_{\perp}$ that is the projection of $\mathbf{A}$ into the plane.

To calculate $\mathbf{N}$, we need to characterize the computational load distribution. In the RIB method [13], the inertia matrix is utilized to calculate the principle inertia axis [3]. Inspired by the RIB method, we can find the direction of constraint by the same means.

The global mass center of simulation is calculated by

$$\overline{\mathbf{Z}} = \frac{\int_{\Omega} \rho(\mathbf{x})\mathbf{x}d\sigma}{\int_{\Omega} \rho(\mathbf{x})d\sigma} = \frac{\sum_{i=0}^{N-1} m_i^{vp}\mathbf{z}_i^{vp}}{\sum_{i=0}^{N-1} m_i^{vp}}. \tag{18}$$

Then the inertial matrix is obtained as

$$\mathbf{J} = \sum_{k=0}^{n-1} l_k^{sp}(\mathbf{x}_k^{sp} - \overline{\mathbf{Z}})(\mathbf{x}_k^{sp} - \overline{\mathbf{Z}})^T, \tag{19}$$

where n denotes the total number of SPH particles simulated. The eigenvalue λ (Eq. (20)) and eigenvector ξ (Eq. (21)) of $\mathbf{J}$ indicate the value and direction of principle inertia, respectively. The eigenvalues are sorted by increasing eigenvector.

$$\lambda = \{\lambda_1, \lambda_2, \lambda_3\}, \tag{20}$$

$$\xi = \{\xi_{\lambda_1}, \xi_{\lambda_2}, \xi_{\lambda_3}\}. \tag{21}$$

The direction of the constraint can be calculated accordingly. We propose an inertial-based adaptive filter defined as

$$\mathbf{N} = \begin{cases} \xi_{\lambda_1} & \text{if } \overline{\lambda_3} > \lambda_{max}, \text{ i.e. constrained along } \mathbf{N} \\ \xi_{\lambda_3} & \text{if } \overline{\lambda_1} < \lambda_{min}, \text{ and } \overline{\lambda_2} > \lambda_{min} - \overline{\lambda_1}, \\ & \text{i.e. constrained on the plane } \perp \mathbf{N}, \end{cases} \tag{22}$$

where $\overline{\lambda_i} = \dfrac{\lambda_i}{\sum_{i=0}^{2} \lambda_i}$. λ_{max} and λ_{min} are user defined thresholds. The selection procedure is to compare the normalized eigenvalues with predefined thresholds, and determine an appropriate strategy. The filter is adaptive as well, which allows for dynamic optimization of the partitioning strategy via choosing different constraints. For instance, if $\overline{\lambda_3} > \lambda_{max}$, the load along the first principle inertial axis, i.e. ξ_{λ_1}, is considerably larger than the combination of the other two axis. Thus, the degrees of freedom in ξ_{λ_2} and ξ_{λ_2} are confined, and only motion along ξ_{λ_1} is allowed. During the computation, if the other condition is satisfied, the strategy will be altered accordingly. If both conditions are not satisfied, the original CVP method is restored.

3.3. Intermediate conclusions

In Sections 3.1 and 3.2, we propose two extensions of CVP method. Both algorithms are mutually independent and compatible since they function at different phases of the computation, thus they can be integrated into a single framework. The objective of the LICVP method is to optimize the incremental property in rebalancing without violating the other constraints in static partitioning. In general, the Lagrangian property improves the data reuse when SPH particles move following the fluid field, while the Inertial CVP guarantees that partitioning is aware of the global load distribution and insensitive of the load variation along directions of minimum interest. LICVP methods can restore to original CVP method under certain conditions, and the implementation requires trivial modification. Moreover, The additional cost due to the extension is minimum since updating Voronoi particles is localized within subdomains and the global inertial matrix is only updated once the repartitioning is triggered.

4. Numerical algorithms

4.1. Partitioning boundary conditions

Symmetric and periodic boundary conditions can be applied on the partitioning domain, see Fig. 2. The ghost particle method is employed to enforce both boundary conditions.

Ghost particles for symmetric boundary conditions (SBC) are constructed following [11]. SBC is employed when the computational domain is symmetric itself or is defined as open boundary.

Periodic boundary conditions (PBC) are enforced in a similar way. In the first step, Voronoi particles that have neighbors on the other side of the periodic boundary are identified, and marked as boundary particles. In the second step, all boundary particles are mapped by adding/subtracting the domain length in the periodic direction, and the Voronoi diagram is constructed again including ghost particles. All physical variables of the ghost particles are identical to the mapped boundary particles. When the background velocity is introduced, PBC is enforced for phase II as well, which facilitates Voronoi particles for better tracing the local subdomains. The benefit of PBC is twofold: (1) SPH particles that are transported across the boundary during simulation cost no extra

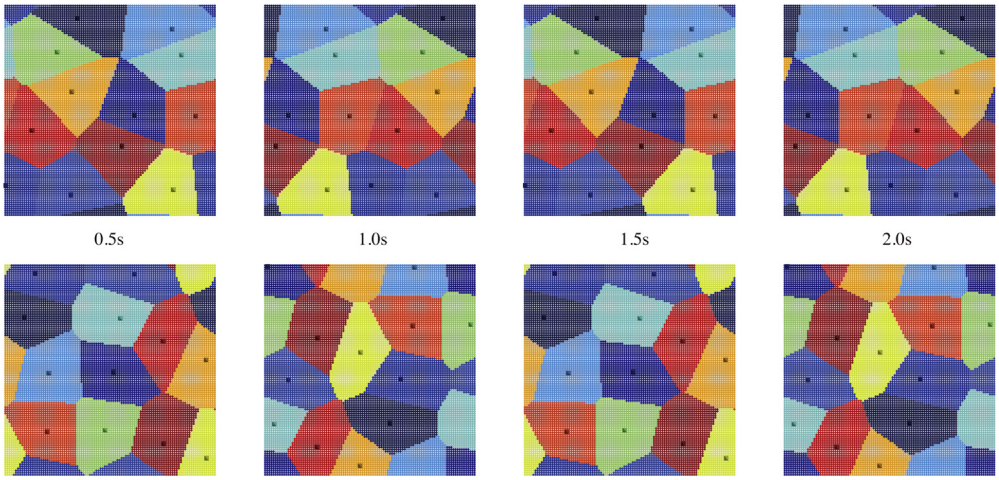


Fig. 3. Steady advection test: the partitioning result of fluid with horizontal (upper row) and diagonal (bottom row) velocity at four instants. SPH particles are rendered with sub-domain index, and octahedra are positions of Voronoi particles.

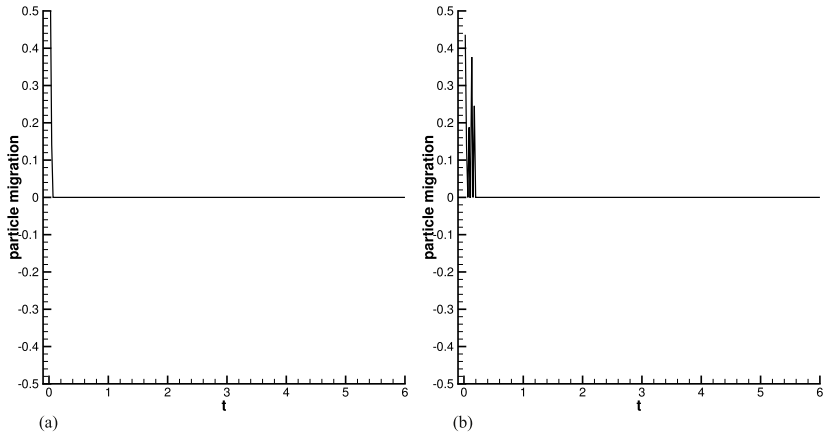


Fig. 4. Steady advection test: time history of averaged particle migration. (a) Fluid with horizontal velocity, (b) fluid with diagonal velocity.

inter-processor data relocation; (2) the rebalancing is calculated allowing Voronoi particles to move across the boundary, thus data reuse is better satisfied.

4.2. Flowchart

The detailed algorithm for LICVP method is summarized in Algorithm 1. For simplicity, we only list the flowchart for one simulation timestep. Other correlative algorithms, e.g. CVP method, data structure, communication strategy, etc., are not elaborated here. One can refer to our previous work [11,12,17] for a comprehensive overview.

5. Numerical validation

In this section, five test cases are considered to validate the proposed LICVP method. The underlying Lagrangian property of LICVP method is demonstrated via case 1, case 2 and case 3, where the Inertial CVP method degenerates to the original CVP method. The feasibility and performance of Inertial CVP method are then discussed in cases 4 and 5. Case 3 is tested on the mpp2 cluster provided by Leibniz-Rechenzentrum (LRZ), which is constructed by 28-way Haswell-EP nodes with Infiniband FDR14 interconnect. The other cases are carried out all on the same workstation with 12 Intel(R) Xeon(R) CPU E5-2630 v2 cores (64G memory and 2.6 GHz) and Scientific Linux system (Release 6.8).

Algorithm 1 Flowchart of LICVP method for one timestep

1: **if** current timestep count is divisible by 20 **then** ▷ *the imbalance monitoring system is activated every 20 timesteps*
2: calculate imbalance error $E_{mc^{vp},i}$ (Eq. (9)) and $E_{m^{vp},i}$ (Eq. (10));
3: Check whether repartitioning is required, i.e. $R = 1$ (Eq. (8));
4: **if** $R = 1$ **then** ▷ *the system will be rebalanced*
5: calculate the inertial matrix J (Eq. (19)), and find its eigenvalue and eigenvector (Eq. (20) and (21));
6: calculate direction of constraint, i.e. $\mathbf{N}$ (Eq. (22)):
7: **while** the partitioning error is not converged **do**
8: construct Voronoi diagram and solve the model equation Eq. (2);
9: calculate the resulting displacement $\Delta\mathbf{x}_i^p$ according to Eq. (16) and Eq. (17);
10: update Voronoi particles;
11: check partitioning error;
12: **end while**
13: migrate SPH particles to target processor according to the new partitioning result;
14: **end if**
15: update data structure and construct ghost buffer particles;
16: solve SPH governing equations;
17: calculate the background velocity $\tilde{\mathbf{v}}_i^{vp}$ for transporting Voronoi particles using Eq. (12) or (14);
18: update the position of Voronoi particles according to Eq. (11);
19: **end if**

Before moving to the results, we first define two measurements to assess the incremental property and optimization for

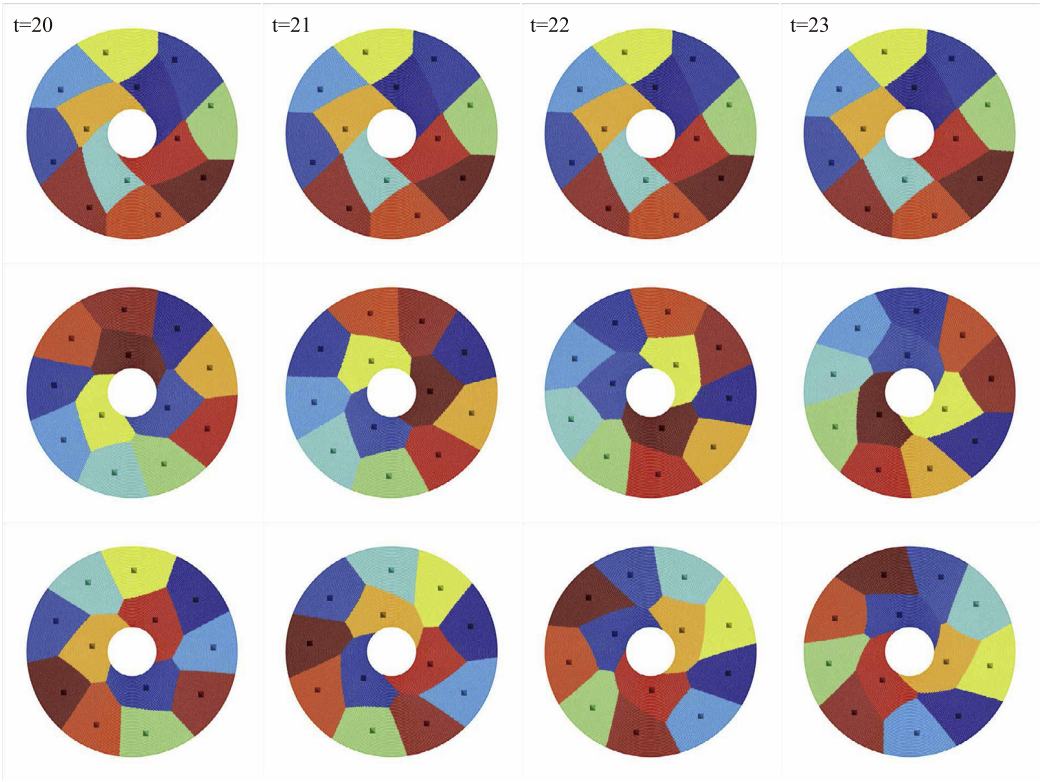


Fig. 5. 2D Keplerian disk problem: comparison between non-moving Voronoi particles (upper row), moving Voronoi particles with velocity of mass center (middle row) and mean fluid velocity (bottom row). Four snapshots are illustrated, i.e. t = 20, 21, 22 and 23. SPH particles corresponding to different sub-domains are assigned with distinct colors, and the octahedrons denote the Voronoi particles.

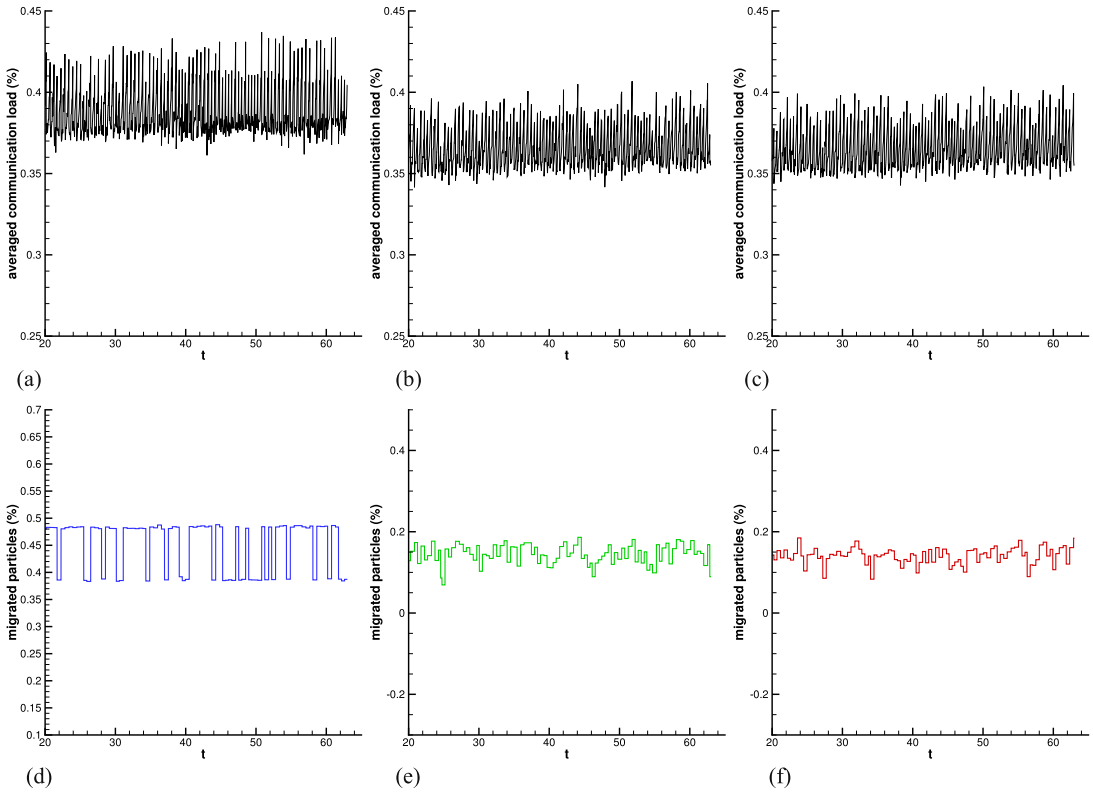


Fig. 6. 2D Keplerian disk problem: The time history of averaged communication load percentage using non-moving Voronoi particles (a), moving Voronoi particles with velocity of mass center (b) and mean fluid velocity (c). The time history of averaged migration percentage using non-moving Voronoi particles (d), moving Voronoi particles with velocity of mass center (e) and mean fluid velocity (f).

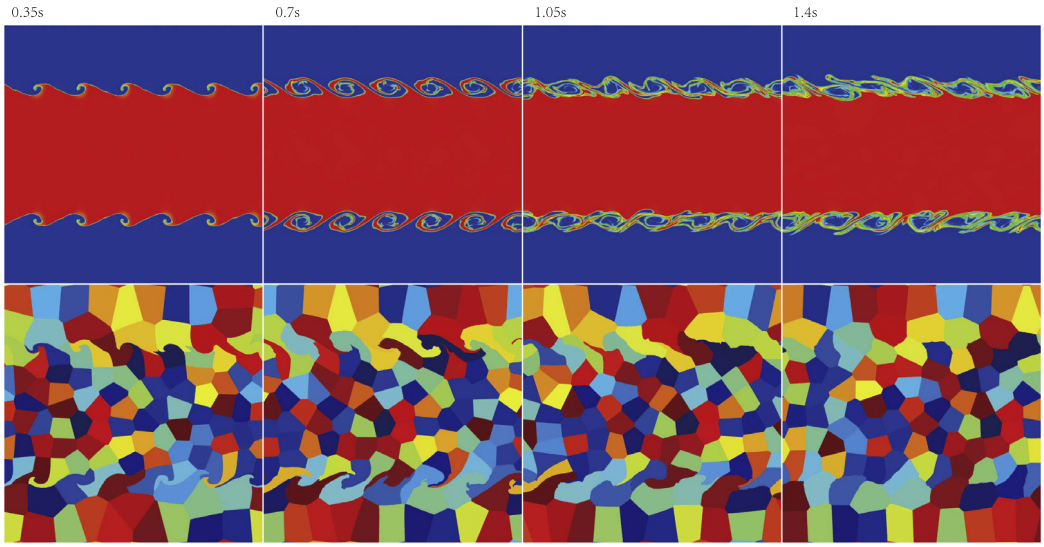


Fig. 7. 2D KH instability: the density field (upper row) and partitioning diagram (bottom row) with respect to four instants.

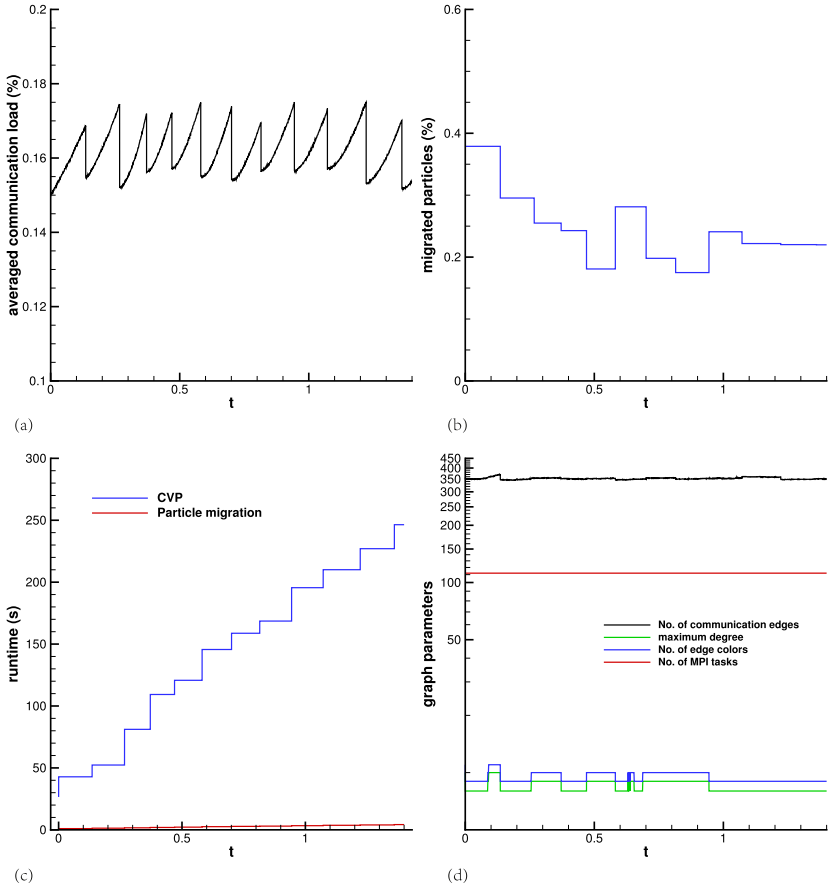


Fig. 8. 2D KH instability: (a) The time history of averaged communication load percentage. (b) The time history of averaged particle migration percentage. (c) Runtime of CVP method and particle migration during simulation. (d) Time history of graph parameters.

communication reduction, i.e. averaged communication load (S_c) and averaged particle migration (S_m). S_c and S_m are defined as

$$S_c = \frac{\sum_{i=0}^{N-1} \frac{N_i^{ghost}}{N_i}}{N}, \tag{23}$$

$$S_m = \frac{\sum_{i=0}^{N-1} \frac{N_i^{migrated}}{N_i}}{N}, \tag{24}$$

where N_i^{ghost} and $N_i^{migrated}$ denote the number of ghost buffer particles constructed and the number of migrated particles respectively.

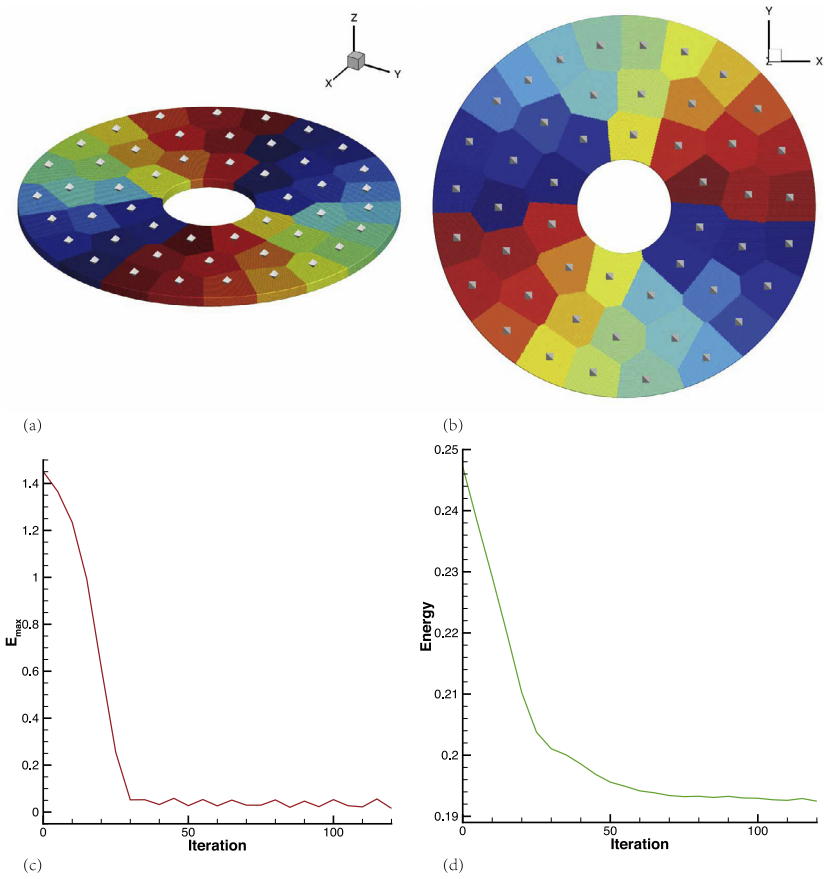


Fig. 9. 3D Keplerian disk problem: the disk is initialized on x–o–y plane. Partitioning result (a)(b). Time history of partitioning error (c) and energy (d).

5.1. Case 1

In the first case, we consider a steady advection test. We initialize a 2D periodic box of unit length. The fluid is assigned with constant density ($\rho = 1000$), and advected with uniform bulk velocity. A weakly compressible SPH solver is employed following [7]. The number of particles simulated is 10 000, and 12 MPI tasks are launched. The partitioning domain is set with equal size of the fluid field and PBC is enforced in both directions.

Since the fluid is advected with constant velocity, the relative position of SPH particles remains identical in entire simulation. With the background velocity, the topology of Voronoi particles should be invariant as well, consequently, after rebalancing, we should obtain the same partitioning diagram every time, and inter-processor particle migration is zero, i.e. data reuse is maximized.

Two bulk velocities, i.e. horizontal ($\{u, v\} = \{1, 0\}$) and diagonal ($\{u, v\} = \{1, 1\}$) velocity, are studied. It is noticed that all the computational load for SPH particle is the same, and the mean velocity of the fluid coincides with the velocity of mass center. We rebalance the simulation every 100 timesteps. The partitioning results at four instants are illustrated in Fig. 3. As anticipated, the topology of the partitioning diagram remains unaltered, and repeats every time unit. The averaged particle migration (see Fig. 4) is zero during entire simulation, except for a small oscillation at the beginning of the second case.

5.2. Case 2

We consider a 2D cold Keplerian disk problem. It is a typical case in astrophysics, where gas orbits a central point mass subjecting to the equilibrium of gravity, centrifugal force and pressure force [18]. We initialize a uniform density disk following [19]. A compressible SPH solver proposed by [20] is employed, where artificial viscosity and conductivity are switched off. To stabilize the flow, a damping term is added to the radial component of particle acceleration [19]. A total number of 47 500 particles are simulated with 12 MPI tasks.

Since the flow is shearing, different orbiting velocities will cause the deformation of sub-domains and an increase of communication load. The proposed two background velocities are chosen to compare with simulation without setting background velocity, i.e. $\widetilde{\mathbf{v}}_i^{vp} = \mathbf{0}$. The snapshot with regard to three situations at four instants are illustrated in Fig. 5. The top row gives the result of $\widetilde{\mathbf{v}}_i^{vp} = \mathbf{0}$. It is observed that the positions of Voronoi particles remain approximately constant after rebalancing and the topology of partitioning result is exactly the same. Conversely, if the Voronoi particles are advected with the background velocity, both results exhibit shifted partitioning sub-domains according to the local flow, and slight topology alteration in a long run. The discrepancy of the result is due to the essential difference between the original CVP and the proposed Lagrangian CVP. In Lagrangian CVP, Voronoi particles are advected according to the main feature of the flow structure to keep track of the existing partitioning sub-domains. If sub-domain Ω_i deforms between each repartitioning, the corresponding Voronoi particle i will be advected according to the averaged velocity or mass center of Ω_i. In this case, as demonstrated in Fig. 5 Voronoi Particles orbit the central point. Conversely, in the original CVP method Voronoi Particles are not updated. Since the resolution in this case is uniform, the resultant partitioning diagram remains almost invariant.

Statistics comparison is manifested in Fig. 6. All three cases demonstrate that after rebalancing, the communication load decreases instantly, and recovers the same value. Moreover, S_c

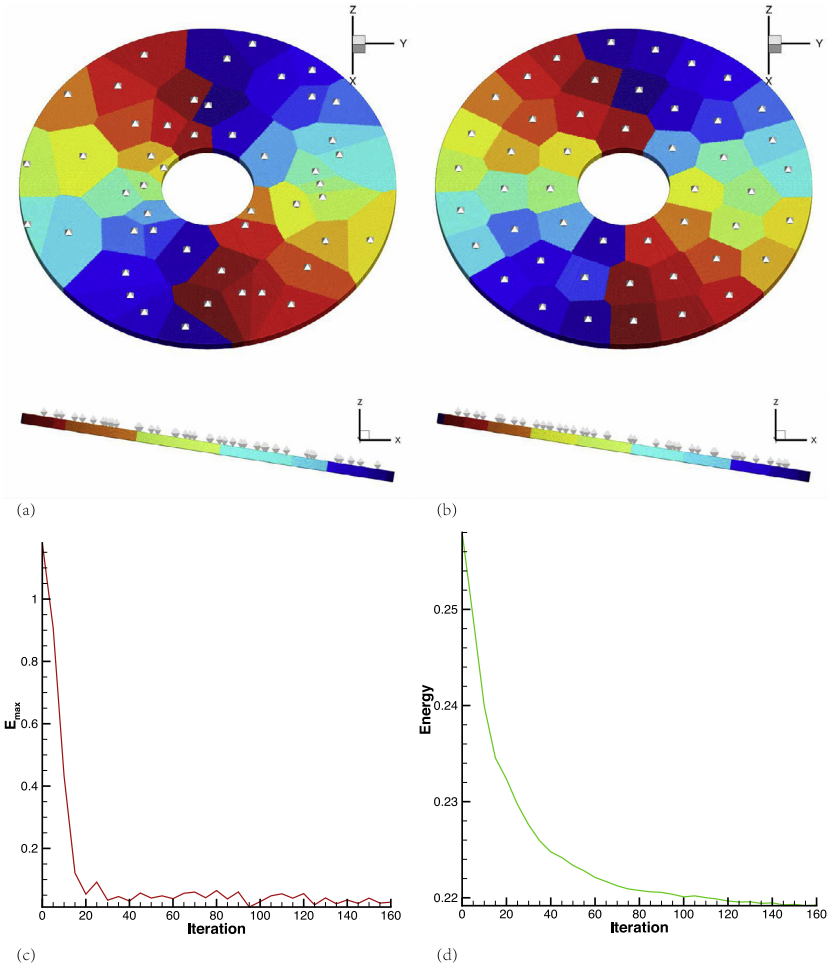


Fig. 10. 3D Keplerian disk problem: the disk is initialized on plane with angle $\pi/4$ to x–o–y plane. Initial condition and Voronoi particle distribution (a). Partitioning result (b). Time history of partitioning error (c) and energy (d).

exhibits a slightly larger value when the Voronoi particles are not updated compared to the other two situations. The S_m for the first case shows that a portion of 39% to 48% particles are being migrated after each rebalancing. This number drops to approximately 15% for the other two cases. It can be concluded that with the background velocity, the incremental property of CVP method is improved. The results from two underlying background velocities exhibit slight differences with respect to performance.

5.3. Case 3

We consider a 2D Kelvin–Helmholtz(KH) instability problem following [20]. A Godunov SPH with second order reconstruction solver is employed [12,21]. PBC is enforced at the boarder of partitioning domain. Eq. (12) is utilized to calculate the background velocity. 6 290 560 SPH particles are simulated on 112 processors. Each processor has 1 TBB thread.

The 2D KH instability problem combines two features from *Case 1* and *Case 2*, where in the smooth region the flow velocity varies slowly, while in regions with strong gradients the flow is dominated by shear stress. The main objective is to test whether the underlying incremental and locality property can still be preserved in a more complex environment. Fig. 7 presents the simulation result as well as the partitioning diagram at four instants. The topological layout of sub-domains in smooth region manifests superior similarity, and sub-domains are transported with the local flow. In regions with discontinuity, due to the development of instability, the partitioning result exhibits topological alteration in a long run. In the simulation duration, i.e. 1.4 s, the system is repartitioned 11 times, and S_c drops immediately after each rebalancing (see Fig. 8(a)). The averaged data migration drops from 40% at the beginning to approximately 25% after 0.5 s (see Fig. 8 (b)), which is consistent with the result obtained in *Case 2* using background velocity. The runtime for CVP method as well as data migration (illustrated in Fig. 8(c)) is negligible compared to the total simulation time (60559s). The graph parameter, which is introduced in Ref. [12] and characterizes the sparse communication relationship, demonstrates that the total number of communication relation is bounded and is about three times larger than the MPI task number (see Fig. 8(d)). The communication finishes within 10 sub-steps during the entire simulation.

5.4. Case 4

The 3D Keplerian disk problem is employed to demonstrate the feasibility of proposed Inertial CVP method. We use the same setup in case 2, and extend the height of the disk to 0.1 in the z direction with identical particle pitch. The dynamic evolution of this case gives the same conclusions as in case 2, thus we only focus on the initial partitioning. Two configurations are calculated, where the disk is initialized on x–o–y plane and plane with angle $\pi/4$ to x–o–y plane respectively. The calculation is carried out with 12 MPI tasks. The Voronoi particles are initialized on disk plane with constant angular separation and a random radius ranging

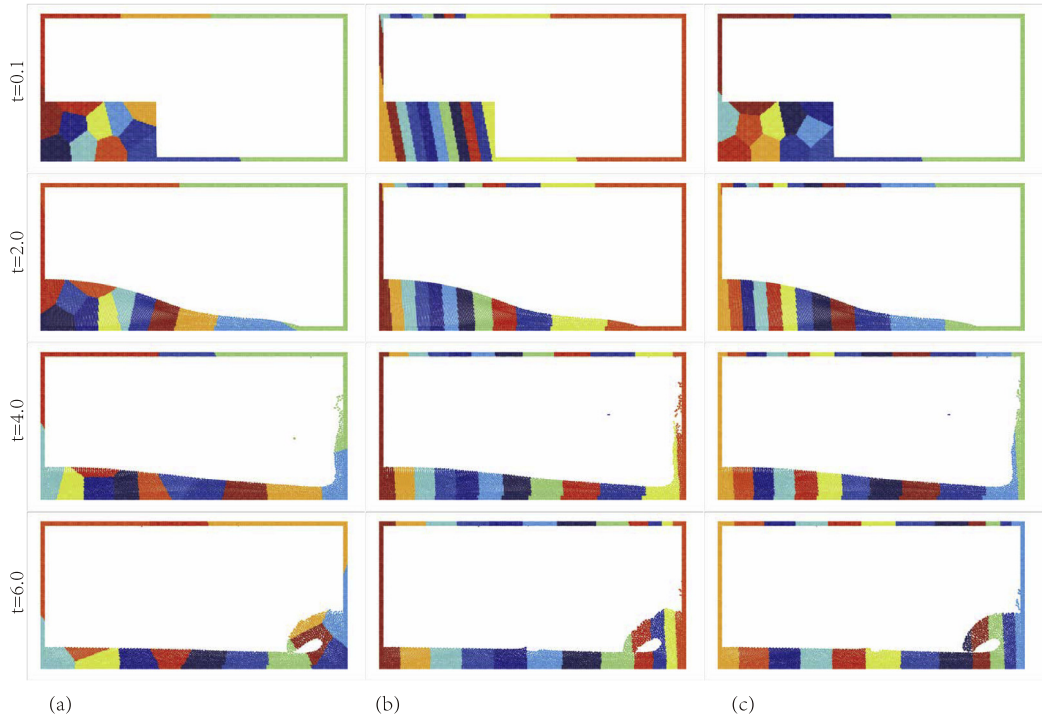


Fig. 11. 2D dam-break: snapshots of simulation result at four instants using (a) original CVP method with background velocity, (b) Inertial CVP with adaptive filter switched off and (c) Inertial CVP with adaptive filter.

between the outer circle and the inner circle. We set $\lambda_{max} = 0.9$ and $\lambda_{min} = 0.1$ in this case.

The results for both cases are plotted in Figs. 9 and 10 respectively. It can be observed that Inertial CVP method captures the load distribution successfully in both cases, where the motion of Voronoi particles is constrained on the disk plane. The partitioning results feature convex, well-shaped sub-domains identical to the original CVP method. The partitioning error converges rapidly. The energy descends monotonically as well, which essentially optimizes the communication volume.

5.5. Case 5

Finally, we consider the 2D dambreak case following the setup from [7]. Initially, a liquid column is placed at the corner of the sink, and collapses due to the existence of gravity. The evolution of flow consists of violent wave-breaking and splashing event, which is crucial in free-surface flow modeling. Meanwhile, the dynamic characteristic of this case deteriorates the performance of repartitioner as well, as the computational load varies dramatically in space and rebalancing becomes expansive. Three means are employed here for a comprehensive assessment of the performance: (1) original CVP method with background velocity; (2) proposed LICVP method with adaptive filter switched off, i.e. only one constraint is applied throughout the simulation, and here only partitioning along the largest principle inertial axis is allowed; (3) proposed LICVP method with adaptive filter, and we set $\lambda_{max} = 0.81$ and $\lambda_{min} = 0.19$. The background velocity is calculated by Eq. (14). The number of particles simulated is 8208, and 12 MPI tasks are launched.

Fig. 11(a)–(c) manifests the simulation result at four instants regarding tests 1, 2 and 3 respectively. Tests 1 and 3 exhibit similar partitioning results at $t = 0.1$, when the water column is not strongly affected by gravity. Whereas test 2 features sub-domains of slim-shaped rectangular, since only one dof (degree of freedom) is unconstrained. Following the propagation of wave front, the load distribution of the system varies accordingly and the load along horizontal axis becomes dominant. At $t = 2$, the partitioning strategy in test 3 is altered, and constraint is applied along **N**, i.e. identical to test 2. The partitioning strategy remains the same afterwards. It is observed that, the incremental property is best preserved in test 2, where the topology of sub-domains is identical during entire simulation. The incremental property for test 1 is the worst, as the topological layout of sub-domains varies constantly. Test 3 generally achieves an intermediate performance, since the partitioning strategy is dynamically shifted at $t = 2$.

The communication load is compared in Fig. 12(a). S_c for test 2 is the largest before $t = 2$, while tests 1 and 3 have similar communication volumes during this period. The slim-shaped sub-domains cannot guarantee optimization of communication reduction compared to the compact sub-domains presented in tests 1 and 3. After $t = 2$, tests 2 and 3 exhibit generally the same level of communication load, whereas test 1 achieves lower S_c during $t = 4$ to $t = 6$ and surpasses tests 2 and 3 afterwards. Although test 1 achieves roughly equivalent performance in optimizing communication load after $t = 2$, the number of communication sub-steps, i.e. edge color, is higher and the system is rebalanced more frequently (see Fig. 12(c)). Regarding incremental property, the same conclusion can be drawn as mentioned in the last paragraph by comparing the averaged data migration (see Fig. 12(b)). The benefit from switching partitioning strategy in test 3 is remarkable, which avoids the violent stage encountered with original CVP method. The runtime comparison is illustrated in Fig. 12(d)). Test 1 is the most time consuming, and with the proposed Inertial CVP method, a speedup of about a factor 6 is achieved.

6. Conclusions

In this paper, a Lagrangian Inertial CVP method is developed by merging the concepts of a background velocity and the Inertial CVP method. The proposed LICVP method is employed as the rebalancer of a multi-resolution parallel framework for the SPH method. The main accomplishment can be summarized as follows:

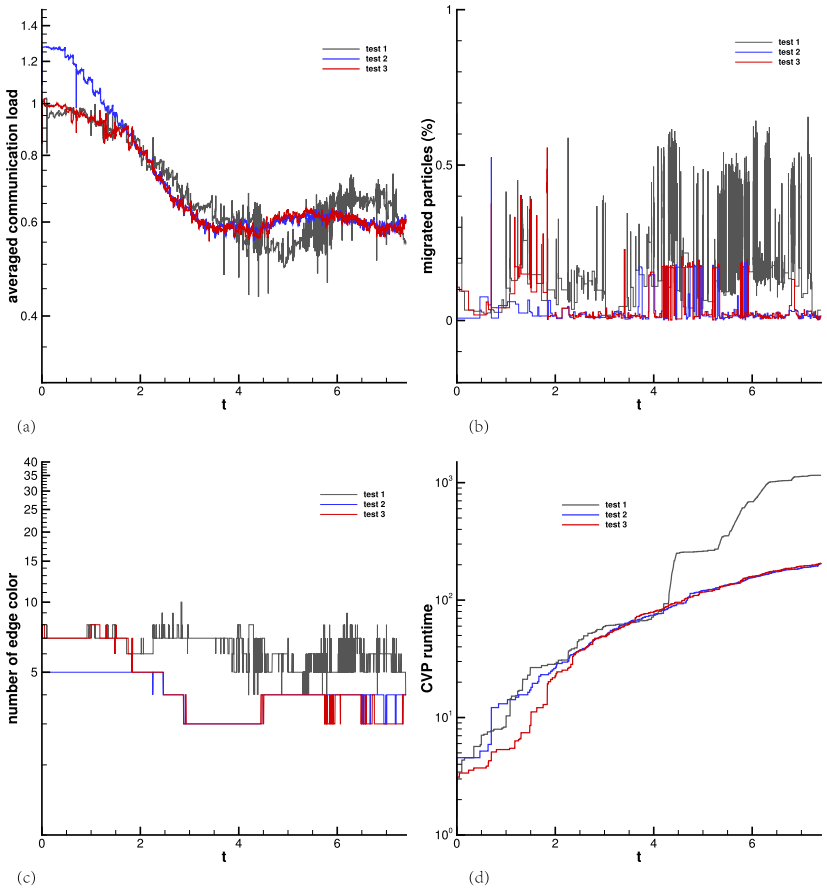


Fig. 12. 2D dam-break: comparison of original CVP method with background velocity, Inertial CVP with adaptive filter switched off and on. (a) Time history of averaged communication load, i.e. proportion of ghost buffer particles versus local registered particles. (b) Time history of averaged migrated particles, i.e. proportion of migrated particles versus local registered particles. (c) Time history of edge color number. (d) Runtime regarding domain decomposition subroutine, i.e. CVP method and data migration.

1. By defining a background velocity, Voronoi particles are able to track the motion of local sub-domains and characterize the topological variation of the system more precisely. Rebalancing upon the updated Voronoi-particle positions improves the incremental property remarkably. Moreover, since the equilibrium is calculated globally, the inter-processor communication is reduced implicitly.

2. The performance of simulations with extremely anisotropic computation-load distribution is improved utilizing the proposed Inertial CVP method. Due to the splitting operator, the Voronoi-particle motion is insensitive of the load variation along the directions of minimum interest, which enhances the incremental property, and improves the convergence as well. Additionally, the adaptive filter allows a dynamic selection of partitioning strategy according to the evolution of load distribution. The selection procedure guarantees a relative balance between data-redistribution and inter-processor communication cost in extreme situations.

Acknowledgments

The first author is partially supported by China Scholarship Council (No. 201506290038). The second author is partially supported by China Scholarship Council (No. 201206290022). The third author acknowledges funding Deutsche Forschungsgemeinschaft (HU 1527/6-1 and HU1527/10-1). The computational resources are provided by Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, Munchen (LRZ).

References

[1] K.D. Devine, E.G. Boman, R.T. Heaphy, B.A. Hendrickson, J.D. Teresco, J. Faik, J.E. Flaherty, L.G. Gervasio, Appl. Numer. Math. 52 (2–3) (2005) 133–152.
[2] K. Schloegel, G. Karypis, V. Kumar, Graph Partitioning for High Performance Scientific Simulations, Army High Performance Computing Research Center, 2000.
[3] B. Hendrickson, K. Devine, Comput. Methods Appl. Mech. Engrg. 184 (2) (2000) 485–500.
[4] L. Fu, S. Litvinov, X.Y. Hu, N.A. Adams, J. Comput. Phys. 341 (2017) 447–473.
[5] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, C. Schulz, Algorithm Eng., Springer, 2016, pp. 117–158.
[6] C. Walshaw, M. Cross, M.G. Everett, J. Parallel Distrib. Comput. 47 (2) (1997) 102–108.
[7] S. Adami, X. Hu, N. Adams, J. Comput. Phys. 231 (21) (2012) 7057–7075.
[8] K. Murotani, S. Koshizuka, T. Tamai, K. Shibata, N. Mitsume, S. Yoshimura, S. Tanaka, K. Hasegawa, E. Nagai, T. Fujisawa, J. Adv. Simul. Sci. Eng. 1 (1) (2014) 16–35.
[9] P.F. Hopkins, Mon. Not. R. Astron. Soc. 450 (1) (2015) 53–110.
[10] H. Meyerhenke, B. Monien, S. Schamberger, Parallel Comput. 35 (10) (2009) 544–569.
[11] L. Fu, X.Y. Hu, N.A. Adams, J. Comput. Phys. 335 (2017) 718–735.
[12] Z. Ji, L. Fu, X.Y. Hu, N.A. Adams, Comput. Methods Appl. Mech. Engrg. 346 (2018) 1156–1178.
[13] H.D. Simon, Comput. Syst. Eng. 2 (2–3) (1991) 135–148.
[14] A. Okabe, B. Boots, K. Sugihara, S.N. Chiu, Spatial tessellations: concepts and applications of voronoi diagrams, vol. 501, John Wiley & Sons, 2009.
[15] S. Lloyd, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[16] Q. Du, M. Emelianenko, L. Ju, SIAM J. Numer. Anal. 44 (1) (2006) 102–119.
[17] L. Fu, Z. Ji, X.Y. Hu, N.A. Adams, Eng. Comput. (2018) in press.
[18] L. Cullen, W. Dehnen, Mon. Not. R. Astron. Soc. 408 (2) (2010) 669–683.
[19] C. Raskin, J. Owen, Astrophys. J. 831 (1) (2016) 26.
[20] D.J. Price, J. Comput. Phys. 231 (3) (2012) 759–794.
[21] S.-i. Inutsuka, J. Comput. Phys. 179 (1) (2002) 238–267.

A.3 Paper III

Zhe Ji, Lin Fu, Xiangyu Y. Hu, Nikolaus A. Adams

A consistent parallel isotropic unstructured mesh generation method based on multi-phase SPH

In *Computer Methods in Applied Mechanics and Engineering*, Volume 363, 2020, pp. 112881, DOI: `https://doi.org/10.1016/j.cma.2020.112881`.

Contribution: My contribution to this work was the development of the method and the corresponding computer code for its implementation. I performed simulations and analyzed the results, and wrote the manuscript for the publication.

Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 363 (2020) 112881
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A consistent parallel isotropic unstructured mesh generation method based on multi-phase SPH

Zhe Ji[a,1], Lin Fu[b,1], Xiangyu Hu[a,*], Nikolaus Adams[a]

[a] *Chair of Aerodynamics and Fluid Mechanics, Department of Mechanical Engineering, Technical University of Munich, 85748 Garching, Germany*
[b] *Center for Turbulence Research, Stanford University, Stanford, CA 94305, USA*

Received 19 June 2019; received in revised form 22 October 2019; accepted 25 January 2020
Available online 26 February 2020

Abstract

In this paper, we propose a consistent parallel unstructured mesh generator based on a multi-phase SPH method. A set of physics-motivated modeling equations are developed to achieve the targets of domain decomposition, communication volume optimization and high-quality unstructured mesh generation simultaneously. A unified density field is defined as the target function for both partitioning the geometry and distributing the mesh-vertexes. A multi-phase Smoothing Particle Hydrodynamics (SPH) method is employed to solve the governing equations. All the optimization targets are achieved implicitly and consistently by the particle relaxation procedure without constructing triangulation/tetrahedralization explicitly. The target of communication reduction is achieved by introducing a surface tension model between distinct partitioning sub-domains, which are characterized by colored SPH particles. The resulting partitioning diagram features physically localized sub-domains and optimized interface communication. The target of optimizing the mesh quality is achieved by introducing a tailored equation-of-state (EOS) and a smooth isotropic kernel function. The mesh quality near the interface of neighboring sub-domains is improved by gradually removing the surface-tension force once a steady state is achieved. The proposed method is developed basing on a new parallel environment for multi-resolution SPH to exploit both coarse- and fine-grained parallelism. A set of benchmarks are conducted to verify that all the optimization targets are achieved consistently within the current framework.

Keywords: Parallel mesh generator; High performance computing; Smoothing particle hydrodynamics; Domain decomposition

1. Introduction

The topic of parallel mesh generation is critical for capturing complex physical phenomena in various areas, e.g. Finite Element Analysis (FEA) [1], Computational Fluid Dynamics (CFD) [2] and image discretization in bioinformatics [3]. Developing scalable, stable and high-quality parallel mesh generation methods is important in reducing simulation cost and achieving high-accuracy for the underlying numerical methods. Recently, new challenges have raised for parallel mesh generation methods due to the rapidly growing capabilities and capacities

* Corresponding author.
E-mail addresses: zhe.ji@tum.de (Z. Ji), linfu@stanford.edu (L. Fu), xiangyu.hu@tum.de (X. Hu), nikolaus.adams@tum.de (N. Adams).
[1] Zhe Ji and Lin Fu contributed equally to this work.

https://doi.org/10.1016/j.cma.2020.112881
0045-7825/

of modern supercomputers. To fully exploit the potential of distributed memory system, the parallel mesh generator needs to resolve various difficulties, e.g. the data dependency, the load balancing and the irregular behavior of the mesh refinement [4]. According to the NASA CFD vision 2030 study [5], mesh generation is still a significant bottleneck in CFD and more research is needed.

Sequential unstructured mesh generation methods can be roughly categorized as advancing front methods (AFT) [6,7], Delaunay refinement-based methods [8,9], Delaunay variational-based methods [10,11], Particle-based methods [12–14], etc. The advancing front method generates the mesh from the geometry boundary and inserts layers of vertices representing the front iteratively towards the interior of the domain [6]. The Delaunay refinement-based method starts from a coarse representation of the geometry and improves the mesh quality by gradually inserting new Steiner points into the domain until a prescribed criterion is achieved [8]. As for Delaunay variational-based method and particle-based method, either an energy function [10,15] or a target mesh-size function [12,16] is defined. Different numerical approaches are then applied to minimize the energy or interpolation error in order to optimize the mesh quality. Despite the similarities shared by both methods, one fundamental difference is whether the connectivity information is required during the optimization procedure. For particle-based mesh generation methods [12,13,17], pair-wise forces are defined between interacting particles to relax the system towards the target distribution. Therefore, the mesh quality is improved implicitly without constructing a Delaunay tessellation for each optimization iteration. Moreover, since the interaction is constrained locally within a short cutoff radius, only local information is required for each particle. Benefiting from aforementioned unique features, the particle-based method can be easily extended to parallel systems and achieve scalable performance.

Comparing to sequential mesh generator, additional targets arise when the mesh is generated in a distributed memory system. Ideally, a parallel mesh generation method should retain the mesh quality generated by the state-of-the-art sequential code and achieve fully code reuse without significantly deteriorating the scalability of the code [18]. Therefore, in order to accomplish the additional goals in a parallel environment, a consistent formulation is required to maintain the quality of the mesh in a parallel environment.

In the past decades, tremendous efforts have been done to develop parallel unstructured mesh generation methods [19]. Initially, most of the developed schemes follow a coarse-grained parallel strategy [20]. A Domain Decomposition (DD) step is first used to partition the geometry into either continuous sub-domains or discrete simply-connected sub-meshes [21,22]. Different sequential mesh-generation kernels are applied to mesh each sub-problem and optimize the interface between sub-domains/sub-meshes respectively. The main effort to increase the parallel efficiency relies on the amount of communication required on the sub-problem interfaces [23]. Depending on the data synchronization strategy, the parallel mesh generations can be categorized into tightly-coupled, partially-coupled and decoupled approaches [19]. Tightly-coupled methods, e.g. Parallel Optimistic Delaunay Mesh (PODM) [24] and Parallel Advancing Front Technique for shared memory computers (PAFT_{SM}) [21], optimize the mesh in the interior and on the interface of each individual sub-problem simultaneously. While stability and quality of the resulting mesh are guaranteed, the parallel implementation induces a significant amount of communication overhead and features low code reuse. For decoupled approaches, e.g. Parallel Projective Delaunay Meshing (P^2DM) method [25] and Parallel Delaunay Domain Decoupling (PD3) method [22], the geometry first is decomposed into continuous sub-domains and each sub-domain is meshed separately. This approach achieves high code reuse, but the mesh quality depends on a proper domain decomposition method. The partially coupled strategy, e.g. Parallel Octree AFT (POAFT) method [21] and Parallel Constrained Delaunay Meshing method (PCDM) [26], find a balance between the aforementioned two approaches. The meshing procedure is separated into two phases by defining an interior region and interface region. The amount of communication is significantly reduced comparing to tightly coupled ones and the codes are more stable in terms of achieving high-quality meshes.

The coarse-grained mesh generators generally feature irregular communication patterns and lack data locality due to the excessive remote data access [27]. Recently with the fast development of manycore processors, e.g. Graphic Processing Unit (GPU), the coarse-grained schemes are no longer suitable for the newly-emerged architectures [28]. Several fine-grained parallel models are exploited to achieve higher concurrency and data locality in shared-memory systems [20,29]. In [20], the data dependency and concurrency are ensured by constructing a graph and utilizing a fine-grained edge-coloring algorithm respectively. Apart from fine-grained parallel models, a hybrid two-level Locality-Aware Parallel Delaunay imaging-to-mesh conversion algorithm (LAPD) is developed in [27]. A partially coupled scheme is employed operating at the coarse level, and a tightly coupled method PODM is utilized to optimize mesh quality within each sub-domain. The inter-node communication only happens at the coarse level

and high-concurrency is maintained by the tightly coupled approach. More recently, a nested master–worker communication model is proposed in [30] to overlap the communication and computation and to further exploit the two-level parallelism on manycore distributed memory system.

To conclude, in order to utilize the full potential of the state-of-the-art clusters, the parallel mesh generation method should be able to achieve the following characteristics: (1) well-balanced load, optimized communication volume, high scalability in the node level; (2) high concurrency and data locality property in the thread level within each node; (3) easy to be extended in a parallel environment, i.e. high code reusability.

According to the above discussions, the particle-based mesh generation method is suitable as a candidate of large-scale parallel-mesh generator. Since the mesh generation procedure is accomplished implicitly without operating on a mesh and the pair-wise interaction only relies on its local information, it fulfills the fine-grained parallelism naturally. With a proper domain decomposition method and dynamic load balancing strategy, e.g. the Centroidal Voronoi Particle (CVP) method [31], scalable performance can be achieved with a large number of computational nodes. Moreover, due to the Lagrangian nature of the particle-based method it is particularly suitable and easy to program for modern parallel environment consisting of shared-memory or distributed-memory systems utilizing various parallel techniques, e.g. Message Passing Interface (MPI) [32], OpenMP [33] and CUDA [34]. A number of well-established codes has been developed for different particle methods on different architectures, e.g DualSPHysics [35] for free-surface weakly-compressible flows using Smoothed Particle Hydrodynamics (SPH), LAMMPS [36] for Molecular Dynamics (MD) and Dissipative Particle Dynamics (DPD) simulations, OpenFPM [37] for hybrid particle-mesh simulations, etc. To the best of our knowledge, the topic of parallel particle-based mesh generator in a distributed memory system has not yet been explored.

In this paper, a consistent particle-based parallel unstructured mesh generation method is developed. Unlike other parallel approaches, which rely on a domain decomposition step first before generating the mesh, the proposed method merges both steps into a single phase. The targets of improving mesh quality, optimizing communication volume and domain decomposition are achieved consistently within one set of physics-motivated modeling equations. By defining a unified target function and introducing a surface-tension model in the governing equation, the newly-developed SPH-based isotropic unstructured mesh generation method [12] is extended to a parallel multi-resolution environment [38,39]. The parallel framework employs both MPI and Thread Building Blocks (TBB) [40] techniques, therefore the mesh-generation procedure is able to exploit the parallelism with both coarse- and fine-grained abstractions.

The rest of the paper is arranged as follows: In Section 2, we first introduce the mathematical description of our targets. The main idea and the physics-motivated modeling equations are then elaborated. The detailed numerical methods, e.g. the geometry definition, the discretization of the modeling equations, the parallel environment, and etc., are presented in Section 3. In Section 4, various validation tests are carried out to demonstrate the performance of the proposed method.

2. Physics motivated models

2.1. Target definition

We first introduce the mathematical definition of the targets in both the domain decomposition and the mesh generation. Given a target function $\Phi(\mathbf{x})$ defined in domain Ω, a point set V is initialized to partition the domain into elements. Each element in the resulting tessellation can be treated as a computational unit. We can characterize the partition as a graph $G = (V, E)$, where E denotes the communication relationship between computational units.

In parallel simulation, V is divided into n disjoint subsets denoted as $V_1, V_2, \ldots, V_n$ respectively, and each subset is associated with one MPI task. An optimal partitioning should have the following properties [41]:

- $V_1 \cup V_2 \ldots \cup V_n = V$ and $V_i \cap V_j = \emptyset$ with $i \neq j$;
- $|V_i| \approx \lceil \frac{|V|}{n} \rceil$, $i = 1, 2, \ldots, n$;
- $\sum_{i<j} E_{ij}$ is minimum, where $E_{ij} = \{\{u, v\} \in E | u \in V_i \wedge v \in V_j\}$.

Note that in this paper only equal mass partitioning is considered.

Regarding to mesh generation, there exists several approaches to characterize the approximation error between the discretized mesh element and the given target function. According to [10], the L^p norm between the gradient of $\Phi(\mathbf{x})$, $\nabla\tilde{\Phi}(\mathbf{x})$, and its interpolation is defined to characterize the error since the mesh quality can strongly affect the gradient error. In this paper, we define an optimal mesh quality by

- $L(\mathbf{x}) = \int_\Omega \| \nabla \tilde{\Phi}(\mathbf{x}) - \nabla\Phi(\mathbf{x})\|_{L^p} d\mathbf{x}$ is minimum.

2.2. Main concept

According to the definition in Section 2.1, we propose that the target function $\Phi(\mathbf{x})$ can be used for both the domain decomposition and mesh generation. The total number of mesh vertices can be calculated basing on $\Phi(\mathbf{x})$, considering that the total mass is known and that each particle has unit mass. The target mass of each sub-domain for load balancing can be determined straightforwardly. A color function can be defined for each particle, where within the same sub-domain particles share the same color.

To achieve the objective of domain decomposition and communication optimization, a surface tension force can be applied between particles with distinct colors to preserve the sharp interface condition between neighboring sub-domains [41]. Consequently, particles of the same color tend to concentrate. Sub-domains are optimized towards convex and compact shape due to the existence of surface tension force. According to [41], the steady-state multi-phase fluid has high similarity to the balanced partitioning diagram, and the objectives of domain decomposition and communication can be achieved implicitly.

During the partitioning stage, the mesh quality can be optimized simultaneously. By introducing a tailored equation-of-state (EOS), the relative error of particle density and target density is characterized as pseudo pressure. The error gradient results in pair-wise particle interaction force and drives particles towards target density distribution while maintaining a regularized and isotropic distribution [12]. Once a steady state has been achieved, the particles in the inner region of each sub-domain are optimized, and the target of minimizing $L(\mathbf{x})$ is achieved implicitly.

Last, the mesh quality near the interface region of neighboring sub-domains can be optimized by gradually alleviating the surface tension force. Since an equilibrium state already has been achieved, the optimization of mesh quality near interface region will only result in local redistribution of mesh vertices, i.e. limited overhead of the communication volume.

2.3. Target function

We first refer the target function as "target density" function or "density" function to relate with fluid dynamics

$$\rho_t = \Phi(\mathbf{x}) \tag{1}$$

Since the target density function determines the size of mesh elements, we can further define the target feature-size function (h_t) based on ρ_t through a mapping function,

$$h_t = Q(\rho_t). \tag{2}$$

The target density function can be defined considering different characteristic fields. In general we can write

$$\begin{cases} \rho_t = \Phi_1(a_1, a_2, \ldots, a_n), \\ h_t = \Phi_2(a_1, a_2, \ldots, a_n), \end{cases} \tag{3}$$

where $a_1, a_2, \ldots, a_n$ are the contributing factors that characterize the mesh-vertex distribution. In [12], the authors suggest to calculate the target function considering the level-set function ϕ, the diffused curvature κ and the minimum distance function ψ taking into account the effect of geometry singularities. Moreover, the target function can also be an arbitrary user-defined function to facilitate capturing details with various objectives.

Based on the target density function, the total mass for generating a volume mesh can be calculated by [14]

$$M_v = \int_\Omega \rho_t \triangle dv, \tag{4}$$

where

$$\triangle = \begin{cases} 1, & \text{if inside the geometry,} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The total mass for generating a surface mesh can be calculated similarly by integrating the target density function over the geometry surface

$$M_s = \int_{\partial\Omega} \rho_t ds. \tag{6}$$

In order to characterize the target information for domain decomposition, we define the total computational load (M_t), i.e. total mass, as

$$M_t = M_v + M_s. \tag{7}$$

Then the target mass for each sub-domain can be derived by

$$M_{proc_i} = \omega_{i,t} M_t, \tag{8}$$

where $i = \{1, 2, \ldots, N_{proc}\}$. $\omega_{i,t}$ is the fraction of the target mass for sub-domain i compared with the total mass, and $\sum_i^{N_{proc}} \omega_{i,t} = 1$. For equal mass partitioning, $\omega_{i,t} = \dfrac{M_t}{N_{proc}}$.

2.4. Model equations

The Lagrangian form of governing equations for isothermal multi-phase compressible flows is [41]

$$\frac{d\rho}{dt} = -\rho \bigtriangledown \cdot \mathbf{v}, \tag{9}$$

$$\frac{d\mathbf{v}}{dt} = -\mathbf{F}_p + \mathbf{F}_v + \mathbf{F}_s, \tag{10}$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}, \tag{11}$$

where ρ is the density, $\mathbf{v}$ the velocity vector, $\mathbf{x}$ the position. $\mathbf{F}_p$, $\mathbf{F}_v$ and $\mathbf{F}_s$ denote the pressure force, the viscous force and the surface tension force respectively.

To close the system, an equation of state (EOS) is required:

$$p = f(\rho), \tag{12}$$

where p denotes the fluid pressure. In the current paper, since the particles are treated as pseudo fluid and the objective is to minimize the interpolation error, the equation of state can be set as

$$p = P_0 \left(\frac{\rho}{\rho_t} \right)^{\gamma}, \tag{13}$$

where P_0 is a constant pressure and γ is a user-defined parameter. This EOS drives particles to relax to the target distribution. Once an equilibrium state has been reached, pressure becomes constant, and consequently the interpolation error is minimized, i.e. $\dfrac{\rho}{\rho_t}$ is approximately constant.

The model equations can be discretized and solved by the Smoothed Particle Hydrodynamics method. The discretized form is presented in Section 3.3.

3. Numerical algorithms

The main numerical algorithms and implementation details are elaborated in this section.

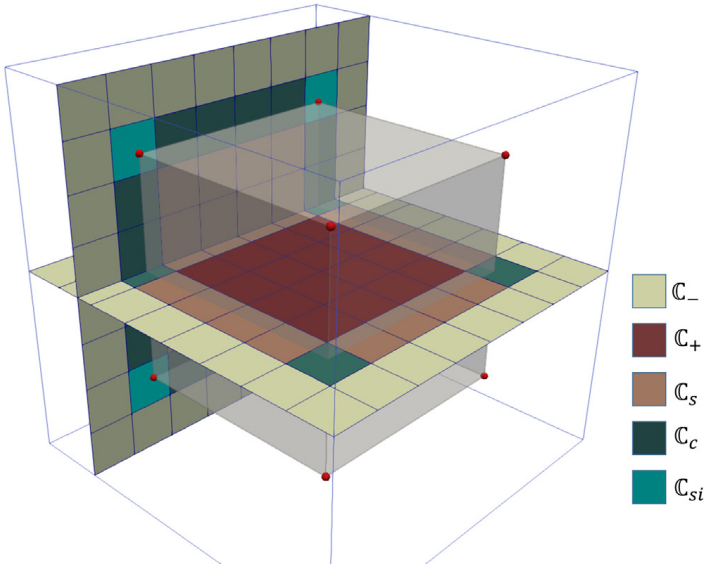


Fig. 1. Tag system developed for characterizing the geometry and facilitating the mesh generating process.

3.1. Geometry definition

First, we use the level-set method [42] to represent the geometry surface using a zero level-set following [12].

$$\Gamma = \{(x, y)|\phi(x, y, t) = 0\}. \tag{14}$$

The level-set field is discretized on a Cartesian background mesh. The mesh-generation region is defined as the positive phase, i.e. $\Gamma_+ = \{(x, y)|\phi(x, y, t) > 0\}$. For detailed description of defining the level-set function on the background mesh, one can refer to [12] and [14].

For complex geometries, there may exist multiple sharp edges and singularity points, which cannot be resolved by a single level-set function. This information should be considered too in order to preserve the accuracy and to recover the geometry. Following [12], the singularity points are directly converted to singularity particles and are fixed during the simulation. Moreover, in this paper, we discretize sharp edges, i.e. feature-curves, with piecewise-linear B-splines. Each segment of the curves is then mapped onto the background grid and all the cells containing feature-curves are characterized as cut-cells. It is also possible to extend to higher order curves, e.g. NURBS curves [43], to improve the accuracy, and will be included in our future work for industrial applications. In addition, the resolution of background mesh used in the current paper is always 1.5 to 2 times higher than the minimum target-feature size. Therefore, the geometry surfaces are always well-resolved compared to the particle size and can be better recovered.

3.2. Target information calculation

The target information defined in Section 2.1 is calculated utilizing the same Cartesian background mesh. A tag system is defined to characterize the positive/negative phase, feature curve, feature surface and singularity respectively. Each cell $\mathbb{C}_i$ is assigned with a unique integer and five categories are defined accordingly, i.e. positive cell ($\mathbb{C}_+$), negative cell ($\mathbb{C}_-$), feature-surface cell ($\mathbb{C}_s$), feature-curve cell ($\mathbb{C}_c$) and singularity cell ($\mathbb{C}_{si}$) (see Fig. 1). Moreover, according to the level-set function, the volume fraction of the positive phase inside each cell can be determined explicitly.

To calculate the target information, the integration can be performed efficiently by looking for corresponding cells based on the tag system. In this paper, the total mass for volume mesh M_v is calculated using the divide-and-conquer method [44] for $\mathbb{S}_v = \{\mathbb{C}_i|\mathbb{C}_i \in (\mathbb{C}_+ \cup \mathbb{C}_s \cup \mathbb{C}_c \cup \mathbb{C}_{si})\}$. The total mass for surface mesh M_s is calculated separately considering the surface integral on the feature surface and the line integration on the feature curve. The surface integration is calculated for $\mathbb{S}_s = \{\mathbb{C}_i|\mathbb{C}_i \in (\mathbb{C}_s \cup \mathbb{C}_c \cup \mathbb{C}_{si})\}$ and the line integration $\mathbb{S}_c = \{\mathbb{C}_i|\mathbb{C}_i \in (\mathbb{C}_c \cup \mathbb{C}_{si})\}$. The same divide-and-conquer method can be applied for the integration. To describe the feature surface explicitly,

the Marching Cube method [45] is used for surface reconstruction. Once the total mass for mesh generation M_t is determined, the target mass for partitioning can be calculated straightforwardly applying Eq. (8).

In order to determine the total number of SPH particles, we assume that each particle possesses unit mass

$$m_i = 1. \tag{15}$$

Since the SPH method inherently conserves mass and there is no mass transfer between particles, the density of each particle will evolve during the relaxation procedure and eventually conforms to the target density field. The target density $\rho_{t,i}$ for particle i is interpolated from the Cartesian background mesh at particle position $\mathbf{r}_i$. The target feature-size is calculated following

$$h_{t,i} = \tau\left(\frac{m_i}{\rho_{t,i}}\right)^{1/d}, \tag{16}$$

where τ is a scaling factor depending on the kernel function and d denotes the spacial dimensionality.

3.3. SPH discretization

With standard SPH method, the density of a particle can be calculated using direct summation over all the neighboring j particles

$$\rho_i = \sum_j m_j W(r_{ij}, h_i), \tag{17}$$

where $W(r_{ij}, h_i)$ is the kernel function, h_i the smoothing length of particle i, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ the connecting vector between particle i and j.

Following [12], the pressure force can be discretized in a symmetric form as

$$\mathbf{F}_{p,i} = \sum_j m_j\left(\frac{p_0}{\rho_{t,i}^2} + \frac{p_0}{\rho_{t,j}^2}\right)\frac{\partial W(r_{ij}, h_{ij})}{\partial r_{ij}}\mathbf{e}_{ij}, \tag{18}$$

where $h_{ij} = \dfrac{h_i + h_j}{2}$, $\dfrac{\partial W(r_{ij}, h_{ij})}{\partial r_{ij}}$ is the derivative of kernel function, and $\mathbf{e}_{ij} = \dfrac{\mathbf{r}_{ij}}{r_{ij}}$ is the unit vector pointing from particle i to j.

Note that the density term in the original discretized form disappears by assuming $\gamma = 2$ in Eq. (13) (see [12]), i.e.

$$p = p_0\frac{\rho^2}{\rho_t^2}. \tag{19}$$

Therefore the density summation term defined in Eq. (17) is no longer necessary.

The viscous force is calculated following

$$\mathbf{F}_{v,i} = \sum_j m_j \frac{2\eta_i\eta_j}{\eta_i + \eta_j}\left(\frac{1}{\rho_i^2} + \frac{1}{\rho_j^2}\right)\frac{\partial W(r_{ij}, h_{ij})}{\partial r_{ij}}\frac{\mathbf{v}_{ij}}{r_{ij}}, \tag{20}$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, and $\eta = \rho\nu$ is the dynamic viscosity. In this paper, we use

$$\nu \sim 0.1 r_c|\mathbf{v}|, \tag{21}$$

where r_c is the cut-off radius of particle interaction range, assuming that the local Reynolds number is on the order of $O(10)$. Moreover, by setting

$$\rho = \rho_t, \tag{22}$$

the viscous force model can be further simplified. Meanwhile, as suggested in [12], a simplified friction model is utilized to set an effectively infinite friction coefficient and to damp particle kinetic energy to zero after each time-step.

As discussed in Sections 1 and 2, the targets of maintaining compact and physically-connected sub-domains are handled by introducing a surface tension model between particles belonging to distinct sub-domains, i.e. particles carrying different colors. Ideally, the surface tension force can be modeled by the continuum surface force (CSF)

method [46] or the continuum surface stress (CSS) method [47], where a finite transitional band is used to characterize the interface. Within the transitional band the surface-tension force is approximated as a continuous force. However, to avoid the direct calculation of curvature a simplified surface tension model is utilized in this paper. Similarly with [41], the acceleration contributed by the interface force can be approximated by an inter-particle repulsive pressure force,

$$\mathbf{F}_{s,i} = -\sum_j \beta(t) m_j \Big(\frac{p_0}{\rho_{t,i}^2} + \frac{p_0}{\rho_{t,j}^2} \Big) \frac{\partial W(r_{ij}, h_{ij})}{\partial r_{ij}} \mathbf{e}_{ij}, \tag{23}$$

where $\beta(t)$ is a time-dependent coefficient to characterize the strength of surface-tension effect

$$\beta(t) = \begin{cases} 0, & \text{if } C_i = C_j, \\ \sigma(t), & \text{overwise}, \end{cases} \tag{24}$$

where C_k is the color function for particle k. Note that the simplified surface-tension model has high similarity with the discretized pressure force formulation, Eq. (18). With the coefficient $\sigma(t) > 0$, particles of different colors near the interface region of neighboring sub-domains are separated by a repulsive force. Moreover, in high-curvature regions particles are expected to concentrate more, consequently resulting in a larger repulsive force to regularize the interface shape.

To incorporate with the mesh generation procedure, the surface tension force is removed once the target of domain-decomposition is achieved. In order to maintain numerical stability, $\beta(t)$ is reduced gradually. In this paper, we use a linear function to ramp down the strength of surface tension effect between the time interval $[t_0, t_1]$

$$\beta(t) = \begin{cases} 3 & , \text{ if } t \le t_0, \\ 3(1 - \frac{t_0 - t}{t_0 - t_1}) & , \text{ if } t_0 < t \le t_1, \\ 0 & , \text{ if } t_1 < t, \end{cases} \tag{25}$$

where t_0 is the time when the initial partitioning is converged, and an initial constant of 3 is set for β following the suggestion in [41]. t_1 can be obtained by adding a fraction of t_0, i.e. $t_1 = (1 + \vartheta)t_0$. In this paper, we set $\vartheta \in [0.1, 0.2]$.

The convergence of partitioning is achieved when the particle system is fully relaxed and an equilibrium state is maintained. We measure the topology variation of the system for a certain amount of iterations, e.g. 50. If the topology remains static for the predefined interval, we assume the partitioning procedure terminates and set $t = t_0$. In order to measure the topology variation, the sampling procedure introduced in [41] can be carried out.

3.4. Time integration

Following [41] and [12], a simplified time-integration scheme is employed as

$$\widetilde{\mathbf{v}}_{n+\frac{1}{2}} = \mathbf{v}_n + \frac{1}{2}\mathbf{a}_n \Delta t, \tag{26}$$

$$\mathbf{v}_{n+\frac{1}{2}} = \widetilde{\mathbf{v}}_{n+\frac{1}{2}} + \frac{1}{2}\widetilde{\mathbf{a}}_{n+\frac{1}{2}} \Delta t, \tag{27}$$

$$\mathbf{r}_{n+1} = \mathbf{r}_n + \mathbf{v}_{n+\frac{1}{2}} \Delta t. \tag{28}$$

The acceleration $\mathbf{a}_n$ is first calculated from the pressure force $\mathbf{F}_p$ and the surface tension force $\mathbf{F}_s$ and is used to update the mid-point velocity $\widetilde{\mathbf{v}}_{n+\frac{1}{2}}$. Then the viscous force $\mathbf{F}_v$ is calculated to obtain the mid-point acceleration $\widetilde{\mathbf{a}}_{n+\frac{1}{2}}$ utilizing the predicted velocity $\widetilde{\mathbf{v}}_{n+\frac{1}{2}}$. At last the particle position is updated by a full timestep according to the modified velocity $\mathbf{v}_{n+\frac{1}{2}}$.

The time-step size of the simulation is calculated with respect to the physical model to maintain numerical stability. In this paper, the time-step size is determined by the CFL criterion, the viscous criterion, and the body force criterion respectively [12],

$$\Delta t = min\Big(0.25\sqrt{\frac{r_c}{|\mathbf{a}|}}, \frac{1}{40}\frac{r_c}{|\mathbf{v}|}, 0.125\frac{r_c^2}{\nu} \Big), \tag{29}$$

where the artificial speed of sound is assumed as $c_s \sim 40|\mathbf{v}|_{max}$.

3.5. Singularity, feature curve and feature surface

Four types of particles, i.e. singularity particle, feature-curve particle, feature-surface particle and normal particle, are defined to characterize the features of underlying geometry specifically. Singularity particles are employed to represent geometrical singularities such as sharp corners. The position is not updated once a particle is marked as singularity particle. Feature-curve particles and feature-surface particles are used to represent sharp edges and surface of the geometry respectively. Particles that are inside the mesh-generation region, i.e. the positive phase, are referred as normal particles.

The particle type is identified using the background mesh. As mentioned in Section 3.2, all the geometry surfaces, singularity points and sharp edges are explicitly marked with cut cells. When a particle crosses feature boundaries, it is characterized according to the type of the cell it lies in and then mapped to the feature according the geometrical information. Therefore, the issue of crossing boundaries can be avoided.

The feature-curve particles are used to provide repulsive force for surface and normal particles to prevent penetration. During the triangulation/tetrahedralization process, they are also utilized to generate 1D line mesh. For the force calculation, only the same type of particles or singularity particles are considered within the cut-off radius. The pair-wise interaction force is projected to the tangential direction of the curve $\mathbf{T}(\mathbf{r}_i)$ at position $\mathbf{r}_i$

$$\mathbf{F}_{*,i} = (\mathbf{F}_{*,i} \cdot \mathbf{T}(\mathbf{r}_i))\mathbf{T}(\mathbf{r}_i). \tag{30}$$

After updating position, the particles are projected back to the feature curve at the closest point to preserve the geometry.

Similarly, the feature-surface particles are used as the boundary conditions of normal particles and are also used to generate surface mesh. The contribution from normal particles are excluded in the force calculation of the feature-surface particles. The normal component of the interaction force is ignored

$$\mathbf{F}_{*,i} = \mathbf{F}_{*,i} - (\mathbf{N}(\mathbf{r}_i) \cdot \mathbf{F}_{*,i})\mathbf{N}(\mathbf{r}_i), \tag{31}$$

where $\mathbf{N}(\mathbf{r}_i)$ is the unit normal vector on the surface. To constrain the particle motion on the surface, the updated position is projected back onto the surface by

$$\mathbf{r}_i = \mathbf{r}_i - \phi_i \mathbf{N}(\mathbf{r}_i). \tag{32}$$

The detailed equations regarding the calculation of $\mathbf{N}(\mathbf{r}_i)$ can be found in [12].

3.6. Triangulation and tetrahedralization

In the proposed method, since the mesh quality is optimized implicitly without connectivity information, the 2D triangulation or 3D tetrahedralization is only performed for post-processing to visualize mesh elements and calculate mesh quality. For 2D triangulation, the mesh is constructed similarly with [12] utilizing a local Voronoi tessellation. At the sub-domain boundaries, the ghost buffer particles from neighboring processors are utilized to generate the Voronoi diagram. A pair-wise synchronization is then performed to remove duplicated elements. Since the resulting particle distribution is sufficiently isotropic and the number of neighbors is always bounded, a unique mesh can be obtained using the information from ghost particles. For 3D tetrahedralization, the open-source code TetGen [48] is used. The flip operations included in TetGen (2–3 flip, 3–2 flip and 4–4 flip) are performed to improve the connectivity.

The runtime of the triangulation/tetrahedralization procedure is not considered in this paper, since it is only a post-processing step. The runtime can be neglected compared to the total runtime for all the cases tested in Section 4. However, when the scale increases significantly, the serial code TetGen may become a bottleneck. In this scenario, the post-processing step can be accelerated utilizing modern parallelization techniques. There are several existing methods that can be employed directly in this regard [49,50] and will be considered in the future.

3.7. Mesh quality criterion

For isotropic triangular meshes, the mesh quality is quantified by $G = 2\sqrt{3}\frac{S}{PH}$ and the angle θ, where S is the triangle area, P the half-perimeter and H the length of the longest edge. θ_{max}, G_{avg} and G_{min}/θ_{min} are the

maximum, average and minimum values respectively. $\theta^{\#}_{min}$ is the averaged value of the minimum angle in each triangle. $\theta_{<30}$ is the number of triangle that contains angle smaller than 30°. The distribution of angles is provided too to check the regularity.

For isotropic tetrahedral mesh, the mesh quality is quantified by the dihedral angle θ and radius ratio $\gamma = 3\frac{r_{in}}{r_{circ}}$ respectively, where r_{in} is the inradius and r_{circ} the circumradius of a tetrahedron. θ_{max}, γ_{avg} and $\gamma_{min}/\theta_{min}$ are the maximum, average and minimum values respectively. $\theta^{\#}_{min}$ is the averaged value of the minimum dihedral angle in each tetrahedron. To evaluate the distribution, diagrams of dihedral angle and radius ratio are provided. The number of slivers are measured by counting the number of tetrahedra with different smallest dihedral angles, i.e. 10°, 20°, 30° and 40°.

3.8. Parallel environment for multi-resolution SPH

The proposed method is implemented in a newly developed parallel environment for multi-resolution SPH [38], which is designed for large-scale simulations with arbitrarily adaptive smoothing-length.

The code utilizes a localized nested hierarchical data structure and a parallel fast-neighbor-search algorithm for an efficient construction of ghost buffer particles in remote processors. With a tailored multi-level cell-linked-list, neighboring particles on remote processors can be identified by simply comparing the tag system constructed on the data structure between two neighboring subdomains [39]. A localized hierarchical data structure is further developed in [38] to eliminate the memory bottleneck and simplify the data management. The cost of parallel fast-neighbor-searching is reduced as well.

An undirected graph is used to characterize the sparse data communication relationship between neighboring subdomains. The communication frequency is optimized by the edge-coloring algorithm [39]. For each communication substep, only efficient nonblocking point-to-point communication is involved. The graph-based communication strategy is further improved in [38] by introducing a concept of "defused graph" to anticipate potential communicating neighbors within a certain time period in the future. Consequently, the reconstruction of the communication graph every-timestep is avoided.

The framework is parallelized with both MPI and TBB. By testing a standard compressible gas dynamics solver, the framework exhibits scalable performance on current state-of-the-art computer clusters for both uniform and adaptive particle distributions. The weak scaling reveals that the code scales well up to at least 3584 cores. Good efficiency is achieved for strong scaling tests (scales up to 7168 cores) at various scales. For more detailed description of the framework, the readers are referred to [38,39,51].

3.9. Initial particle sampling

The initial particle sampling can be proceeded in two steps. First, the domain is roughly divided into a desired number of subdomains. Then SPH particles are sampled randomly into each subdomain. In [41], the geometry based close packing technique is utilized for the first step. For regular geometries, this method can generate satisfying results. However, for complex geometries and large variations of the density field, this method fails to partition the geometry into balanced subdomains. In this paper, the Centroidal Voronoi Particle (CVP) method [31] is employed. The CVP has been originally developed as a mesh-type-independent domain decomposition method. The main contribution is to merge the concepts of Centroidal Voronoi Tessellation (CVT) and Voronoi Particle dynamics (VP). The CVT is introduced to achieve a high-level compactness of the partitioning subdomains, while the Voronoi Particle dynamics is incorporated by solving physics-motivated governing equations to relax the particle system towards the target partition with good load balance. Later, the CVP has been extended to generate optimal initial conditions for particle-based methods [52]. The CVP is able to handle complex geometries and large jump of the density function.

The CVP is easy to implement and to be parallelized [31,38]. First, a set of Voronoi generators are initialized according to the number of MPI tasks desired. Then the generators are randomly placed in the positive phase, i.e. $\Gamma_+ = \{(x, y)|\phi(x, y, t) > 0\}$. Since the target mass of each subdomain is already known according to Eq. (8), then the information can be used to solve the governing equations of CVP on the background mesh until an equilibrium state is achieved. The boundary conditions of CVP is enforced following [52]. Since the number of Voronoi generators is negligible comparing to total number of SPH particles and the CVP only need to be performed

once, the cost for generating a good initial seeding is trivial. After the final positions of the Voronoi generators are obtained, the SPH particles are initialized randomly within the Voronoi cell. For complex geometries, it is recommended to place SPH particles randomly inside a sphere, whose diameter is the minimum distance to its nearest Voronoi generator, to achieve better compactness of the resulting subdomains.

3.10. Overview

A detailed flowchart of the proposed mesh generation method is summarized in Algorithm 1. All the equations are organized in the same order as implemented in the code. It is worth noticing that the equations share highly similarities with standard multi-phase SPH method [53], and the solver can be easily modified from an existing SPH code. Therefore, the infrastructures of the original code developed in [38] can be used directly as an independent module. The processes of constructing localized data structure (Line 5 and 8), parallel fast neighbor searching (Line 9), establishing communication topology and data communication (Line 5 and 8) remain intact. One major difference comparing to a SPH solver is that, in the current method, particle migrations between neighboring domains are not necessary due to the surface tension model introduced. The aforementioned modules are not expanded and elaborated in detail in Algorithm 1, and more information can be found in [38,39] respectively. For preprocessing, an additional background mesh needs to be constructed and the level-set function has to be solved in order to obtain the target information (Line 1–3).

Algorithm 1 Flowchart of the parallel mesh generation method

1: Initialize the background Cartesian mesh;
2: Initialize the level-set function (Eq. (14)) and target density function (Eq. (3)) basing on the background mesh;
3: Calculate the target information for mesh generation (Eq. (7)) and domain decomposition (Eq. (8));
4: Initial particle sampling; ▷ *See Section 3.9 for detailed description*
5: Initialize the parallel environment; ▷ *e.g. construct local data structure, build communication topology, allocate resources, and etc.*
6: **while** $t < t_{end}$ **do**
7: Define particle target density (ρ_t), scale (h_t, see Eq. (16)) as well as other information at $\mathbf{r}_i^n$;
8: Refresh data structure and communication topology; ▷ *See [38] for detailed description*
9: Construct ghost buffer particles, and find neighboring particles to construct neighbor list;
10: Reset particle velocity and forces;
11: Calculate pressure force $\mathbf{F}_p$ (Eq. (18));
12: Calculate surface-tension coefficient (Eq. (24) and accumulate surface-tension force $\mathbf{F}_s$ (Eq. (23));
13: Map $\mathbf{F}_p$ and $\mathbf{F}_s$ for feature-curve and feature-surface particles (Eqs. (30) and (31));
14: Set physical time-step size (Eq. (29)) and update the mid-point velocity $\widetilde{\mathbf{v}}_{n+\frac{1}{2}}$ (Eq. (26));
15: Accumulate viscous force (Eq. (22));
16: Set physical time-step size (Eq. (29)) and update predicted velocity $\mathbf{v}_{n+\frac{1}{2}}$ (Eq. (27));
17: Update particle position (Eq. (28));
18: Find particles that are close to the geometry features utilizing the tag system (defined in Section 3.2). Map the new singularity/feature-curve/feature-surface particles into corresponding singularity/feature curve/feature surface (Section 3.5);
19: **if** Do post-processing **then**
20: Generate the corresponding mesh and calculate mesh quality;
21: **end if**
22: **end while**

4. Numerical validation

In this section, a set of two- and three-dimensional test cases are presented to validate the performance of our method. All cases in this section are simulated on the facilities provided by Leibniz-Rechenzentrum (LRZ). For all the test cases below, we define the communication volume as N_{eg}/N_{et}, where N_{eg} is the number of elements that contain vertices of different colors and N_{et} is the total number of elements generated.

4.1. Circle

We first consider a 2D circle with adaptive resolution. The domain is $[0, 100] \times [0, 100]$, and the circle is defined as

$$\Gamma = \{(x, y) | 43 - \sqrt{(x-50)^2 + (y-50)^2} = 0\}. \tag{33}$$

Table 1
Mesh quality of the circle case.

	G_{avg}	G_{min}	θ_{max}	θ_{min}	$\theta^{\#}_{min}$	$\theta_{<30}$	N_p	N_{tri}[a]
circle_6mpi	0.91	0.53	110.7	28.2	52.0	11	4,289	7,977
circle_12mpi	0.93	0.55	109.1	29.9	53.9	2	13,129	25,111

[a] N_p denotes the total number of particles and N_{tri} the total number of triangles.

The target feature-size function is given as $h_t = h_{min} + \frac{tanh(\frac{2.5\phi}{43})}{tanh(2.5)}(h_{max} - h_{min})$, where h_{max} and h_{min} are the maximum and minimum mesh size. Two resolutions are simulated with different number of MPI tasks. For the first case (referred as *circle_6mpi*), we set $h_{max} = 3.125$ and $h_{min} = 0.391$, and 6 MPI tasks are launched. For the second case (referred as *circle_12mpi*), we set $h_{max} = 1.95$ and $h_{min} = 0.195$, and 12 MPI tasks are launched. The simulation results for *circle_6mpi* are illustrated in Figs. 2 and 3. Figs. 4 and 5 are results of case *circle_12mpi* respectively. The measurement of mesh quality for both cases are shown in Table 1.

From the simulation results, it can be observed that before the ramping-down of surface tension force, all sub-domains feature convex and connected shape, and a sharp interface is maintained between neighboring sub-domains (see Figs. 2(c) and 4(c)). After removing the surface-tension force, the sharp-interface condition is gradually relaxed and the mesh vertices near the interface regions are regularized to an isotropic distribution (see Figs. 2(d) and 4(d)). The final meshes still feature convex shape of sub-domains (see Figs. 2(b) and 4(b)), and the increase of communication volume after removing the surface tension force is negligible (see Figs. 2(f) and 4(f)).

High quality meshes are generated for both cases and for both in the regions near geometry boundaries and in the far field (see Figs. 2(a)(e) and 4(a)(e)). The convergence history of mesh quality and runtime information are also provided in Figs. 3 and 5. Both cases feature proper convergence. Also it can be observed that the overall mesh quality has a rapid increase during the ramping-down of the surface-tension force stage, which starts at approximately 4000 iterations for *circle_12mpi* and 40,000 for *circle_12mpi*. This phenomenon is consistent with the expectation since the repulsive force between sub-domains introduces irregularities at the interface regions.

4.2. Square

Second, we consider a 2D square case with constant resolution using larger scale and larger amount of MPI tasks (referred as *square_56mpi*). The size of the square is 95×95. The total number of mesh vertices is 1,772,894 and 56 MPI tasks are launched. Within each MPI tasks 14 TBB threads are initialized for higher concurrency. The target feature-size is set with a constant value $h_t = 0.0714$. For the initial condition, we first initialize 56 Voronoi generators and sample the generators uniformly into the background mesh in positive region. The SPH particles are then randomly sampled inside each Voronoi cell. The simulation results are presented in Figs. 6 and 7 and mesh quality statistics are given in Table 2.

The resulting partitioning diagram (see Fig. 6(a)(b)) features compact and convex shape of sub-domains, which has high similarities with the partitioning results using the CVP method [31] and a graph-based partitioning method in [54]. Similarly with the first case, the sharp-interface condition is preserved before removing the surface tension force (see Fig. 6(d)). After optimizing the mesh quality near the interface, small disturbance is introduced at the inter-domain boundaries (see Fig. 6(d)), and the amount of communication overhead is limited too (see Fig. 6(e)). It is worth mentioning that the optimization of communication with the presence of surface tension force does not always lead to the monotonic decrease of communication volume, which depends on the initial condition particle seeding and the distribution of the target function.

High-quality meshes are obtained from the calculation. Over 80% of angles range from 55° to 65° (see Fig. 7(a)). The convergence history of mesh quality measurements presented in Fig. 7(b) and (c) demonstrate the good convergence of our method.

4.3. Bunny

The *Stanford Bunny* model [55] is considered. A constant target feature-size of $h_t = 1.5$ is defined and the total number of particles is 240,370. 20 MPI tasks are allocated and each contains 14 TBB threads. Similarly

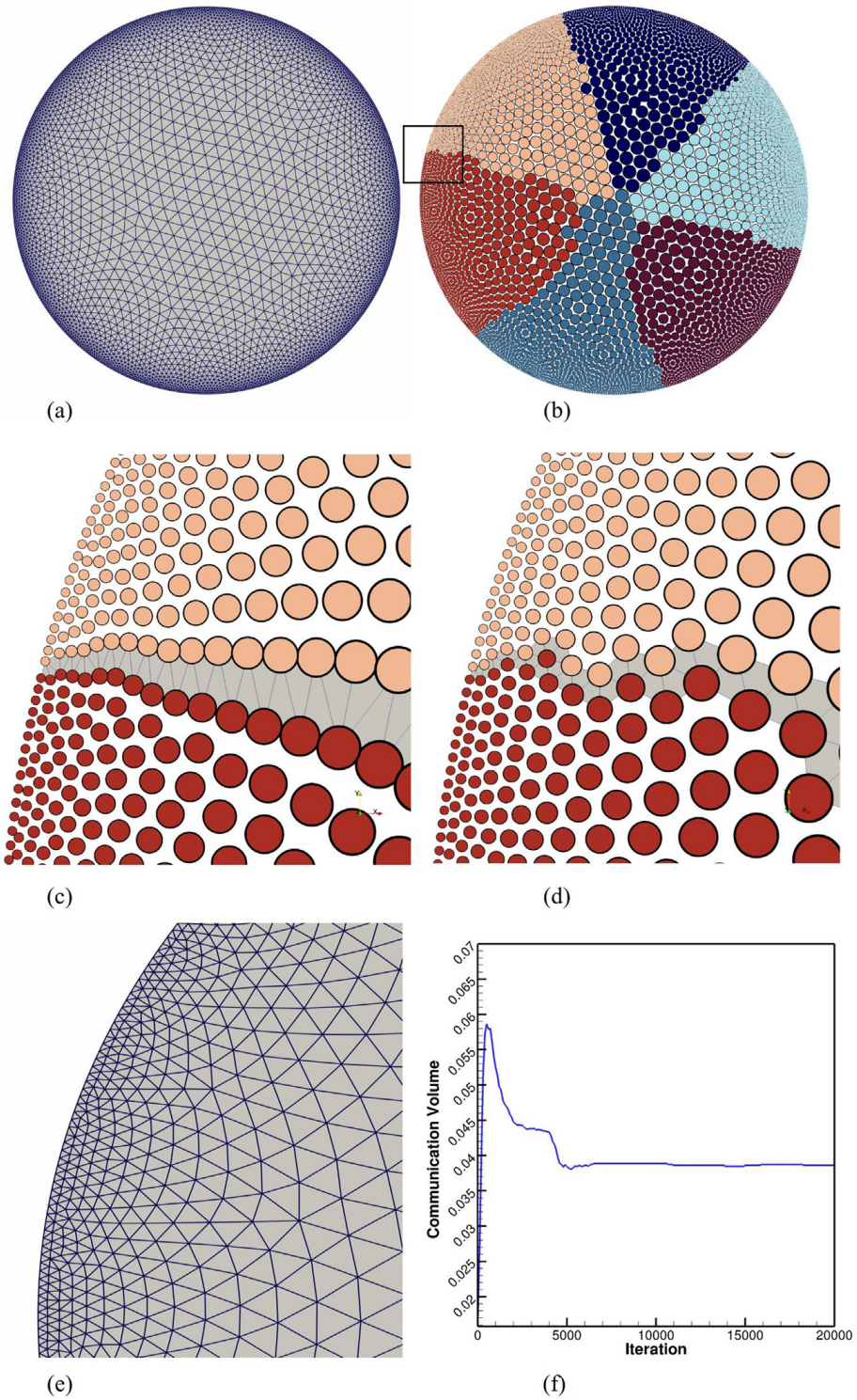


Fig. 2. *circle_6mpi*: (a) Generated mesh after 20,000 iterations. (b) Particle distribution after 20,000 iterations. Particles are plotted with colors of each sub-domain and radius identical to the target feature-size. Particle distribution (c) before removing surface tension force and (d) after relaxation (zoom-in view of the box region in (b)). (e) Zoomed-in view of the final mesh after 20,000 iterations. (f) History of communication volume. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

with previous case, 20 Voronoi generators are sampled uniformly inside the geometry. Particles are then initialized randomly inside a sphere, whose diameter is the minimum distance to its nearest neighbor. The initial seeding of SPH particles are illustrated in Fig. 8(a), where particles are rendered with sub-domain colors.

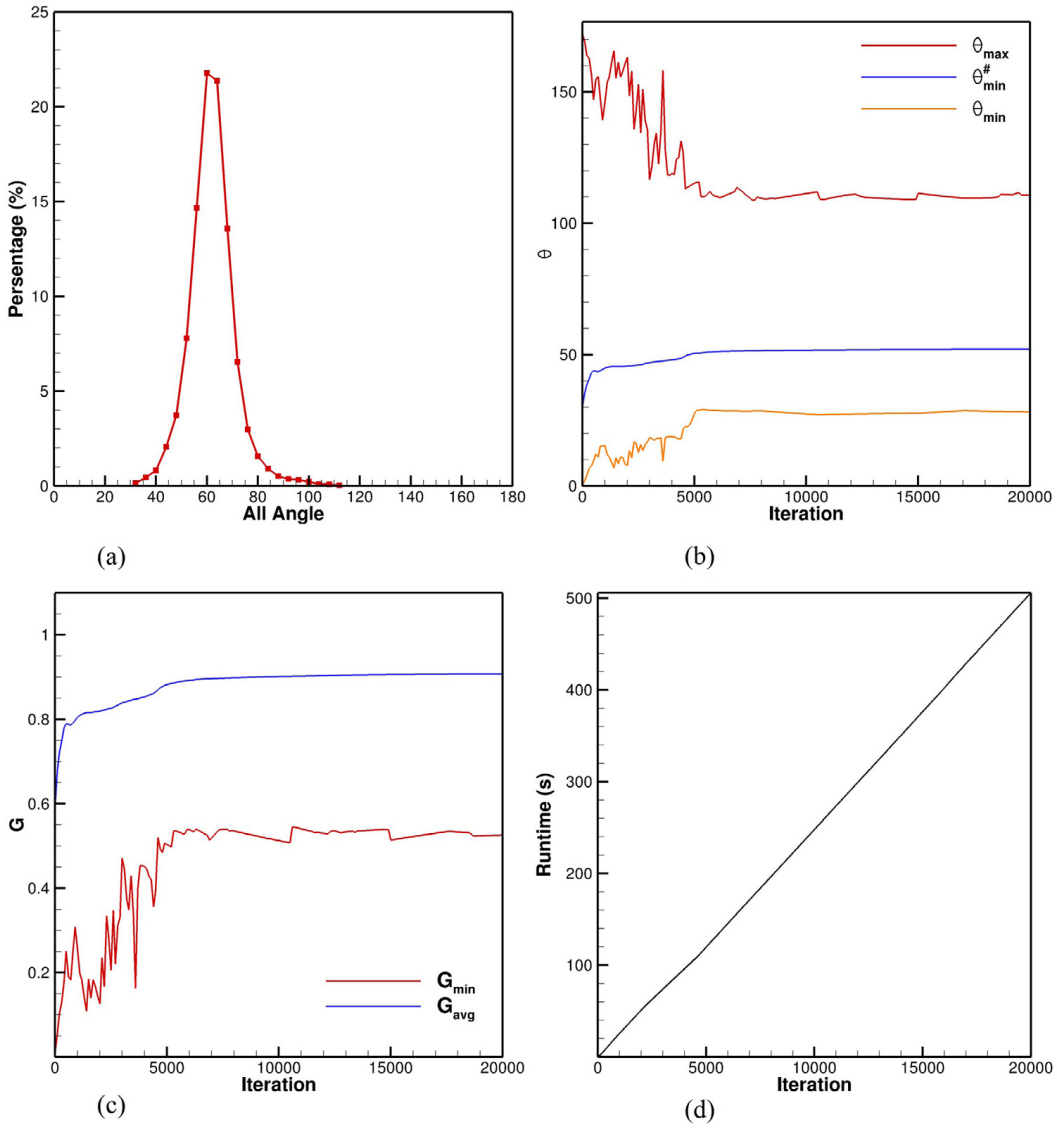


Fig. 3. *circle_6mpi*: (a) Histogram of the angle distribution. (b) Convergence history of θ_{max}, θ_{min} and $\theta^{\#}_{min}$. (c) Convergence history of G_{avg} and G_{min}. (d) History of runtime.

Table 2
Mesh quality of the square case.

	G_{avg}	G_{min}	θ_{max}	θ_{min}	$\theta^{\#}_{min}$	$\theta_{<30}$	N_p	N_{tri}
square_56mpi	0.96	0.58	105.3	35.5	56.9	0	1,772,894	3,541,674

The particle distribution and the generated mesh are illustrated in Fig. 8(c)–(e) at two different camera positions. A cross-section view is presented as Fig. 8(b) to show the inter-domain boundaries. All the results demonstrate that the compact and connected sub-domains are maintained, even at the connecting region of the ears. The features and details of the model are well-captured and mesh vertices are distributed homogeneously inside the geometry. The history of communication volume as shown in Fig. 9(f) exhibits a slight increase after the ramping-down of the surface-tension force (after 75,000 iterations), however the overall overhead is approximately 0.4%.

Good mesh quality is obtained and the statistics are presented in Fig. 9 and Table 3 respectively. The histories of mesh quality demonstrate that the simulation converges properly.

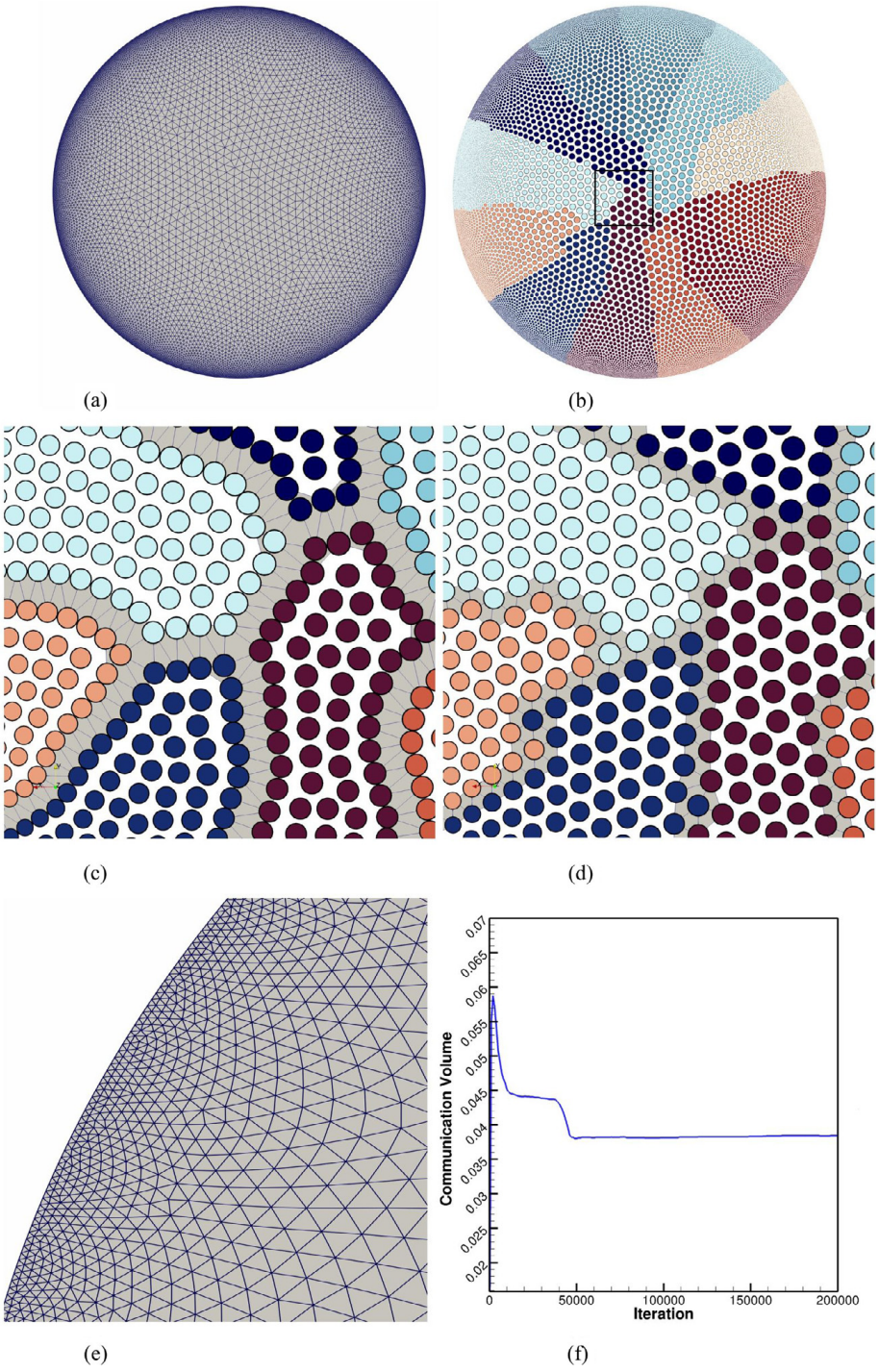


Fig. 4. *circle_12mpi*: (a) Generated mesh after 200,000 iterations. (b) Particle distribution after 200,000 iterations. Particles are plotted with colors of each sub-domain and radius identical to the target feature-size. Particle distribution (c) before removing surface tension force and (d) after relaxation (zoom-in view of the box region in (b)). (e) Zoomed-in view of the final mesh after 200,000 iterations. (f) History of communication volume. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.4. Cube

We consider a cube of size [95 × 95]. This is a simple geometry but contains both singularities and feature curves. A constant target feature-size of $h_t = 0.78$ is defined and the total number of particles is 1,888,113. 24

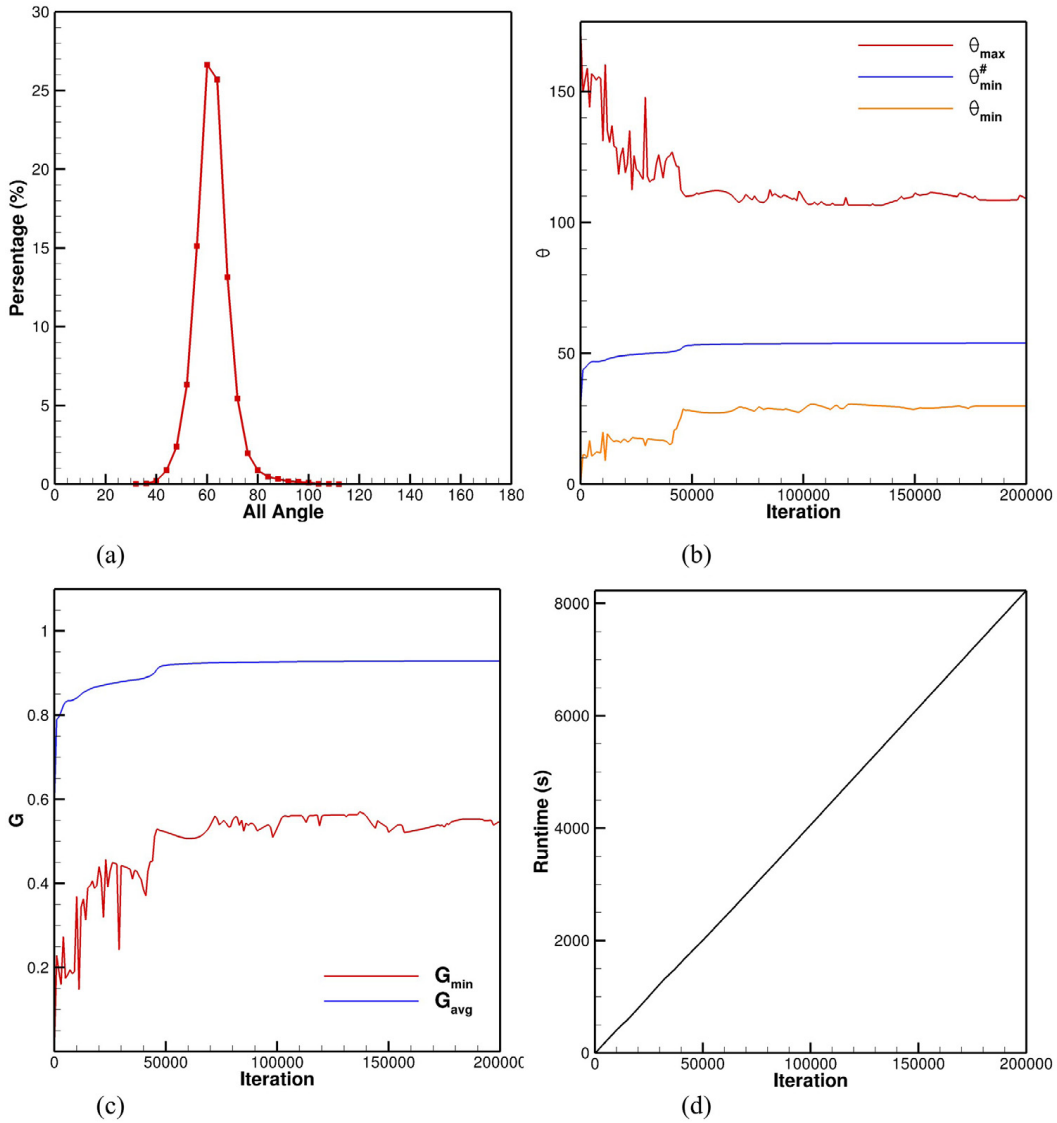


Fig. 5. *circle_12mpi*: (a) Histogram of the angle distribution. (b) Convergence history of θ_{max}, θ_{min} and $\theta^{\#}_{min}$. (c) Convergence history of G_{avg} and G_{min}. (d) History of runtime.

Table 3
Mesh quality of the Stanford bunny case.

	$\theta_{min}/\theta_{max}$	$\gamma_{min}/\gamma_{avg}$	$\theta^{\#}_{min}$	$\theta_{<10}$	$\theta_{<20}$	$\theta_{<30}$	$\theta_{<40}$	N_p	N_{tet}
bunny _20mpi	15.3/154.8	0.24/0.92	56.6	0	2	127	8636	240,370	1,366,196

MPI tasks are allocated and each contains 14 TBB threads. Similarly with the bunny case, 24 Voronoi generators are sampled uniformly inside the geometry. The initial particle seeding is shown in Fig. 11(a).

The simulation result after 217,500 iterations is illustrated in Fig. 10 presented by particles rendered with sub-domain colors (see Fig. 10(a)(c)(e)) and tetrahedra (see Fig. 10(b)(d)(f)). Again overall compact and convex sub-domains are maintained with small disturbance at the interface (see Fig. 10(c) for a zoom-in view and Fig. 10(e) for a clipped view). The overhead of communication volume after relaxation of surface tension force is approximately 1%.

Regarding to mesh quality, statistics and history curves are presented in Table 4 and Fig. 11. According to the Dihedral angle histogram in Fig. 11(c), most of the angles concentrate at approximately 60° (∼60%) and 90° (∼25.5%), which is highly close to the dihedral angles of a Body-Centered-Cubic (BCC) tetrahedron and features

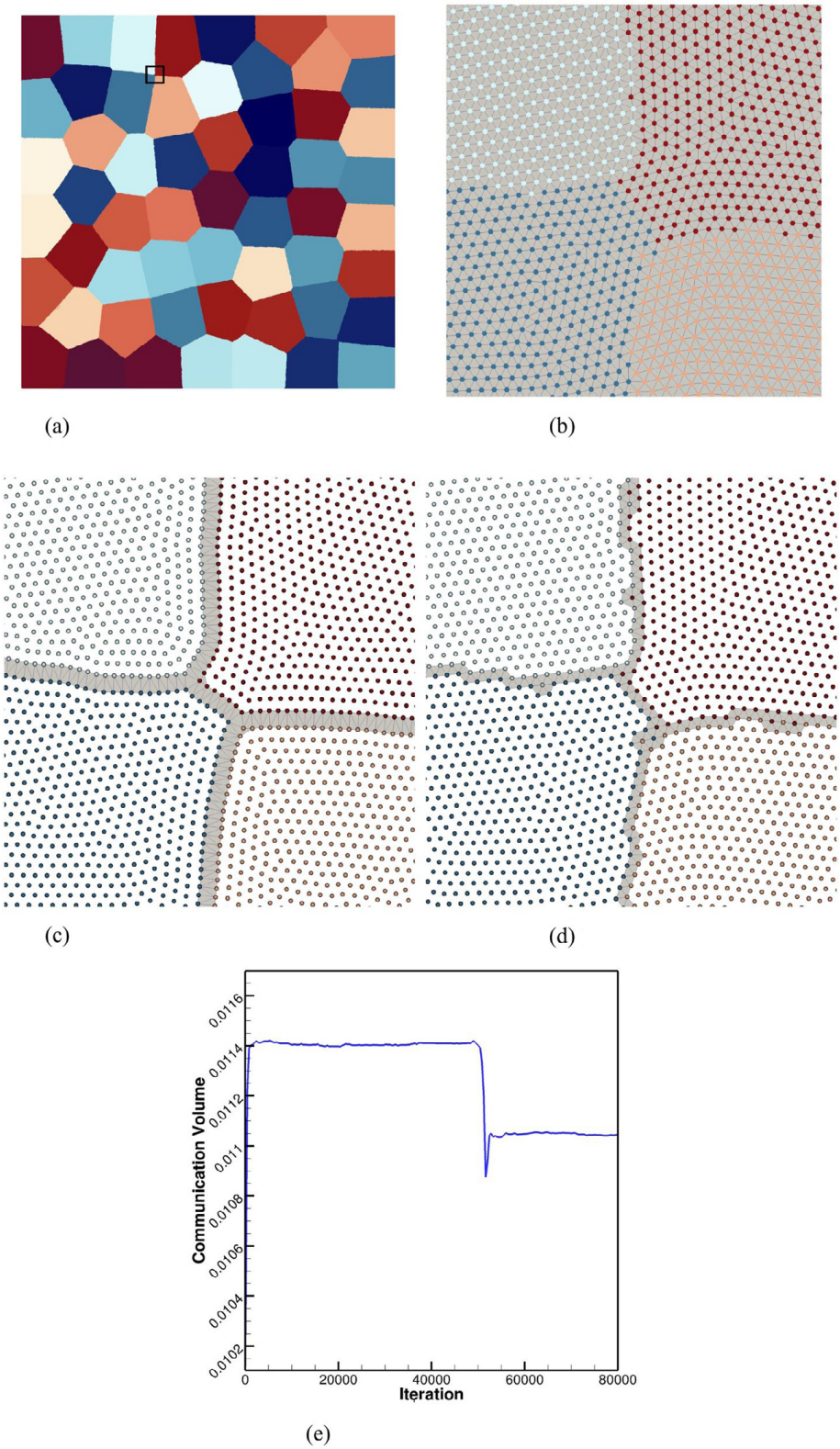


Fig. 6. *square_56mpi*: (a) Generated mesh after 80,000 iterations. (b) Zoomed-in view of the box region in (a) after 80,000 iterations. Particles are plotted with colors of each sub-domain. Particle distribution (c) before removing surface tension force and (d) after relaxation (zoom-in view of the box region in (a)). (e) History of communication volume. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

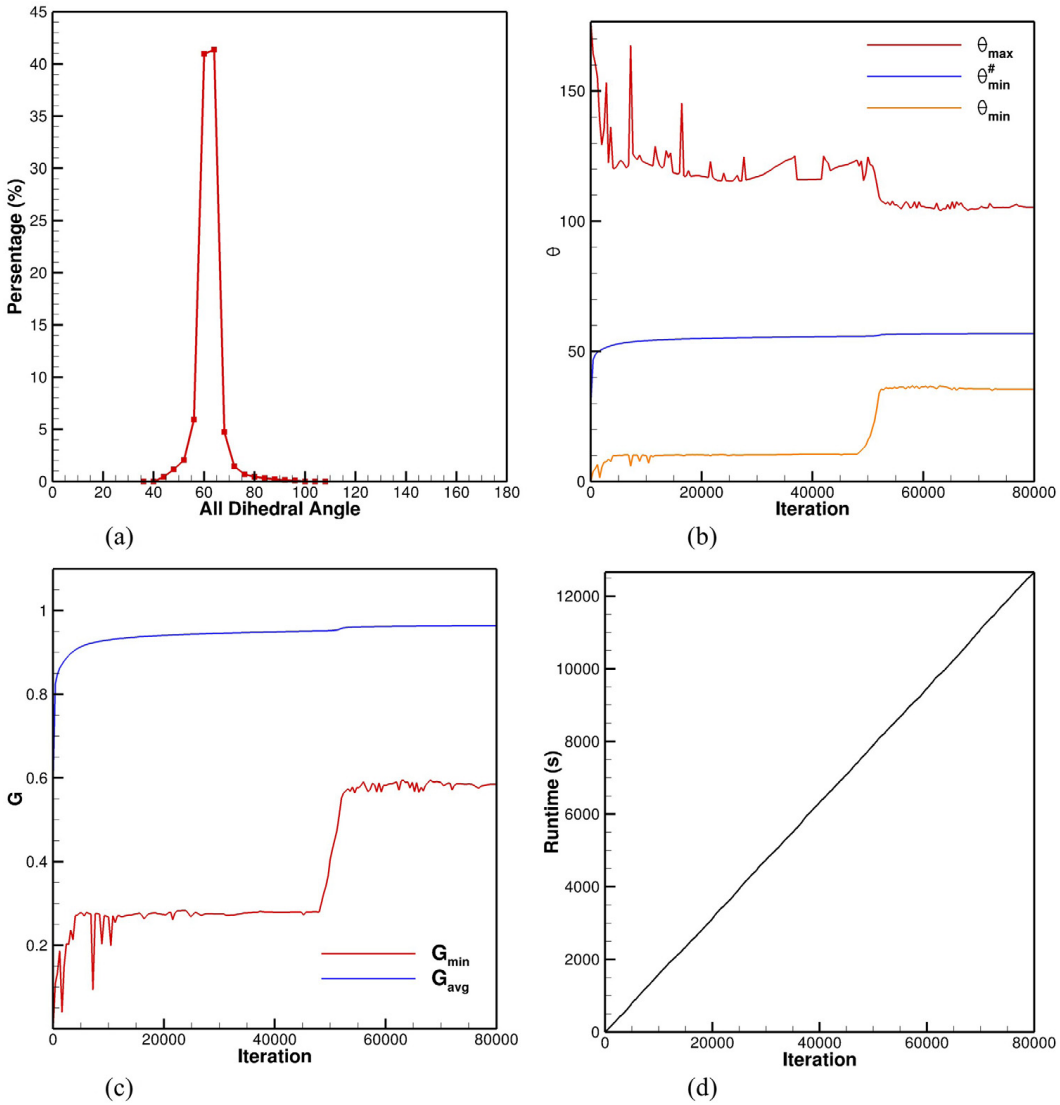


Fig. 7. *square_56mpi*: (a) Histogram of the angle distribution. (b) Convergence history of θ_{max}, θ_{min} and $\theta^{\#}_{min}$. (c) Convergence history of G_{avg} and G_{min}. (d) History of runtime.

Table 4
Mesh quality of the cube case.

	$\theta_{min}/\theta_{max}$	$\gamma_{min}/\gamma_{avg}$	$\theta^{\#}_{min}$	$\theta_{<10}$	$\theta_{<20}$	$\theta_{<30}$	$\theta_{<40}$	N_p	N_{tet}
cube _24mpi	21.5/147.3	0.27/0.94	58.8	0	0	187	7796	1,888,113	11,127,061

minimum mean square error [56]. From the zoom-in view (see Fig. 10(d)) and clipped view of the generated mesh (see Fig. 10(f)), the BCC distribution of mesh vertices can be observed. For the radius ratio diagram (see Fig. 11(d)), about 83% of all tetrahedra falls into the range between 0.94 to 1.13.

4.5. Spur gear

Lastly we consider a realistic geometry of spur gear developed for gear lubrication tests [57]. Multiple singularities and sharp edges are presented in the model. The size of computational domain is $[86.4 \times 17.6 \times 86.4]$. The minimum and maximum target feature-size is 0.4 and 1.6 respectively. The target density field is calculated considering singularities, feature curve, curvature and distance to the geometry surface similar with [12]. The total number of particles calculated is 358,836. 6 MPI tasks are allocated and each contains 14 TBB threads. 6

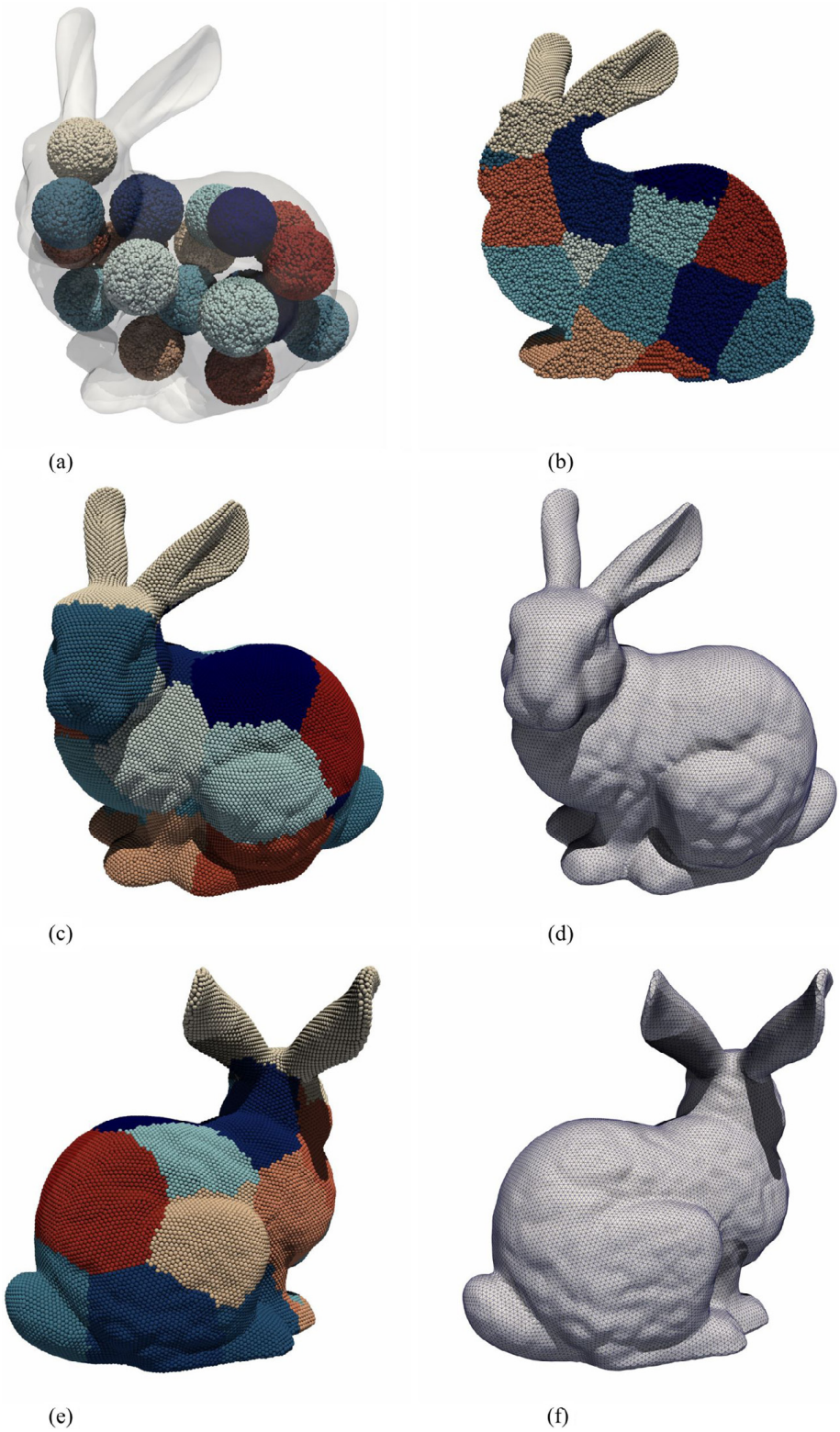


Fig. 8. *bunny_20mpi*: (a) Initial particle distribution. Particles are plotted with sub-domain colors. (b) Cross-section view of the particle distribution after 100,000 iteration. Front view of resulting mesh after 100,000 iteration plotted with (c) particles and (d) surfaces with edges. Back view of resulting mesh after 100,000 iteration plotted with (e) particles and (f) surfaces with edges. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

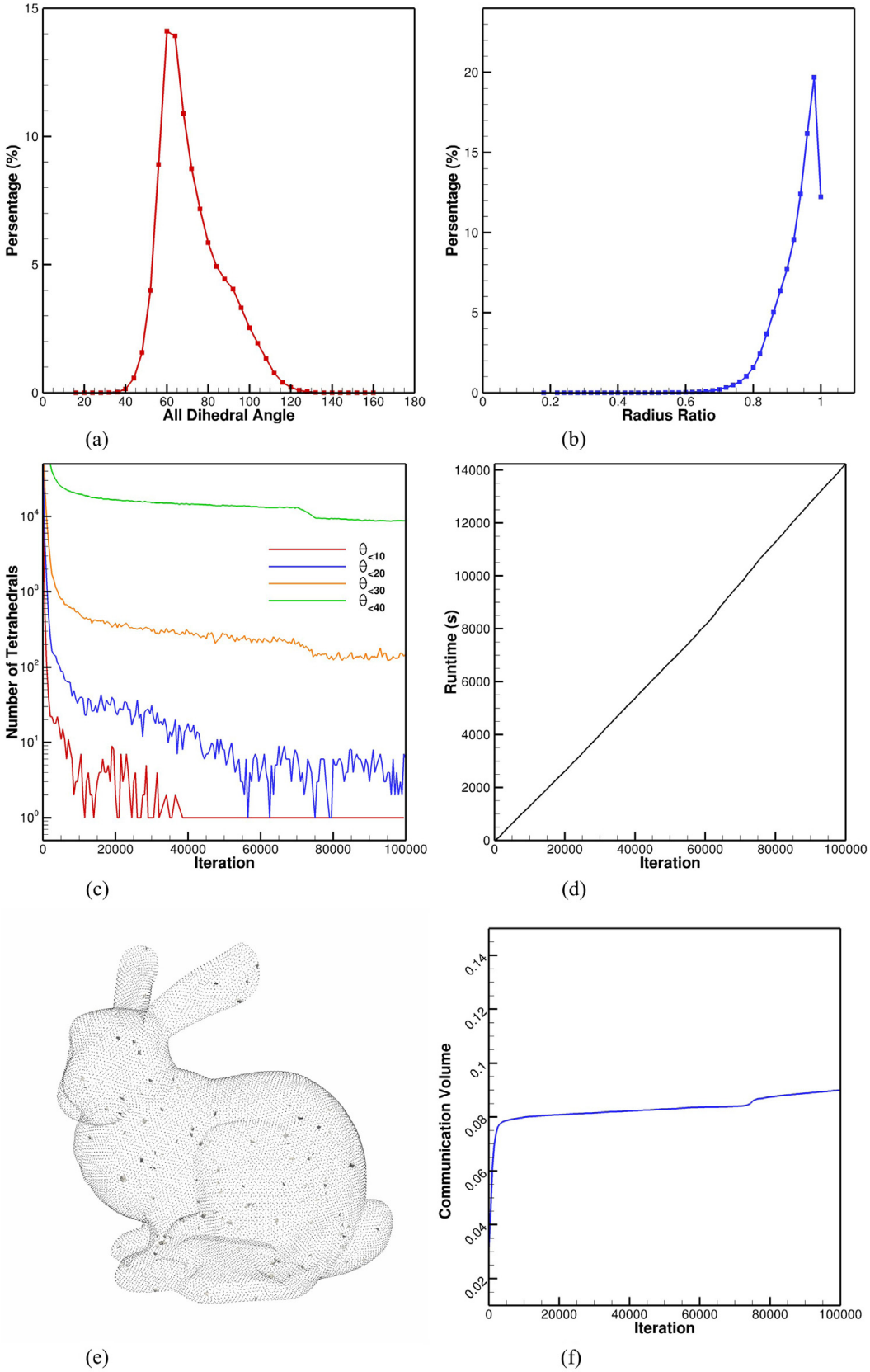


Fig. 9. (a) Histogram of the dihedral angle distribution. (b) Histogram of the radius ratio distribution. (c) Convergence history of number of tetrahedra with minimum dihedral angle smaller than 10°, 20°, 30° and 40°. (d) History of runtime. (e) Tetrahedra with minimum dihedral angle smaller than 30°. (f) History of communication volume.

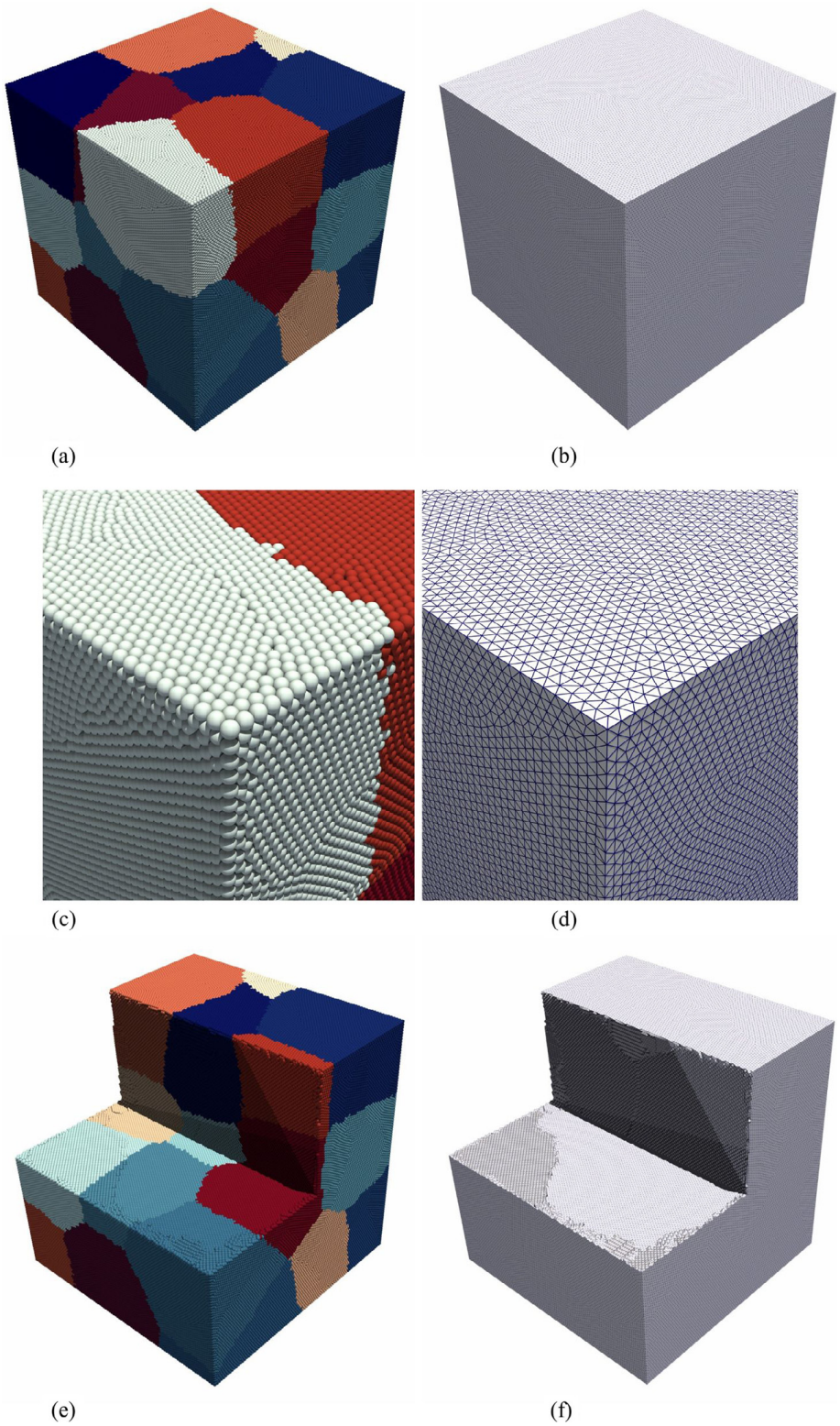


Fig. 10. Simulation result after 217,500 iterations plotted with (a) particles and (b) surfaces with edges. (c) Zoomed-in view of (a). (d) Zoomed-in view of (b). Simulation result with clipping after 217,500 iterations plotted with (e) particles and (f) surfaces with edges. Particles are rendered by sub-domain colors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

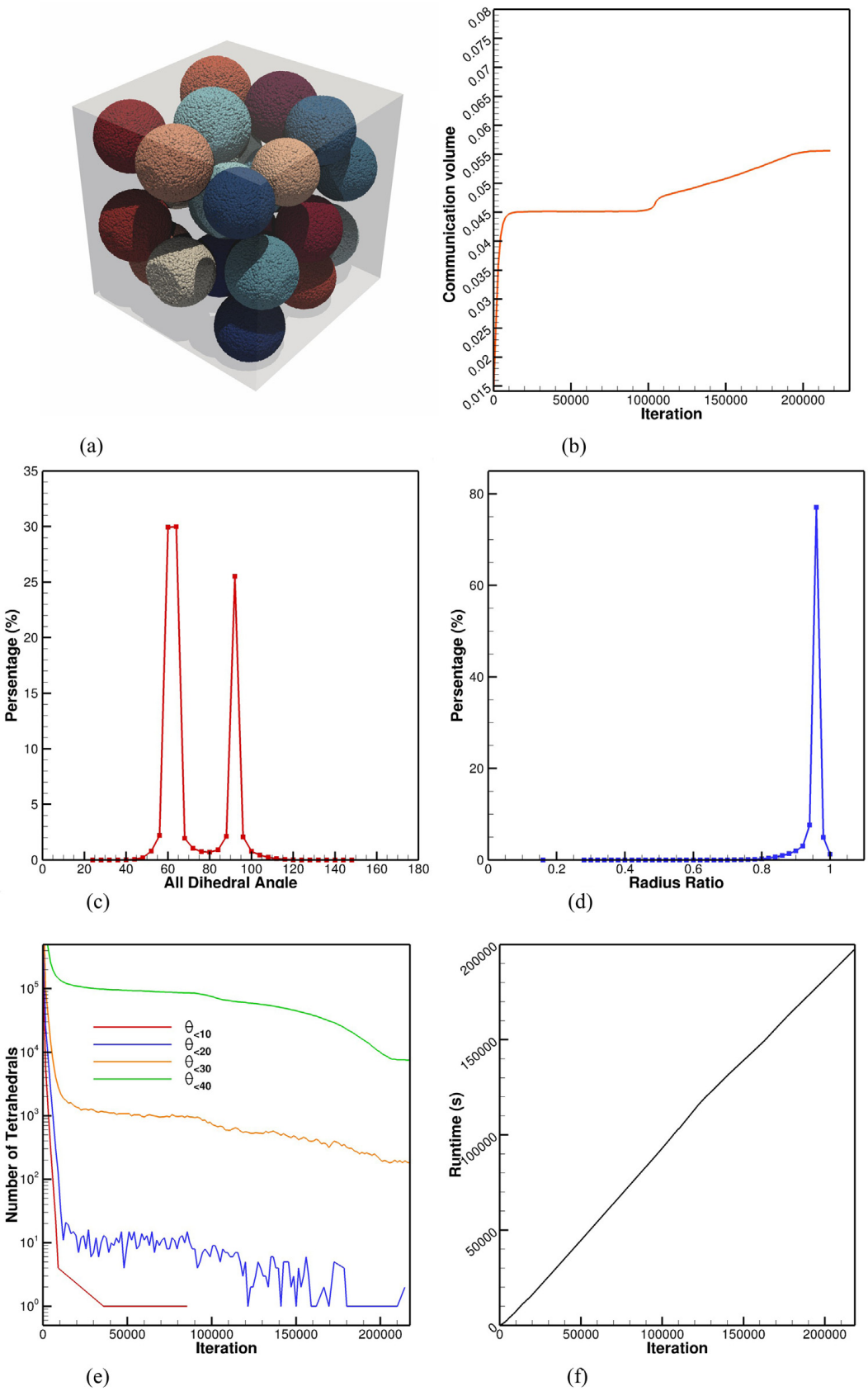


Fig. 11. (a) Initial seeding of particles. (b) History of communication volume. (c) Histogram of the dihedral angle distribution. (d) Histogram of the radius ratio distribution. (e) Convergence history of number of tetrahedra with minimum dihedral angle smaller than 10°, 20°, 30° and 40°. (f) History of runtime.

Table 5
Mesh quality of the spur gear case.

	$\theta_{min}/\theta_{max}$	$\gamma_{min}/\gamma_{avg}$	$\theta^{\#}_{min}$	$\theta_{<10}$	$\theta_{<20}$	$\theta_{<30}$	$\theta_{<40}$	N_p	N_{tet}
Spur_gear _6mpi	6.84/168.7	0.14/0.90	54.8	1	162	4244	73411	358,836	1,847,416

Voronoi generators are uniformly distributed in the geometry and particles are initially randomly sampled within each Voronoi cell. The initial condition is shown in Fig. 13(a).

The generated results after 250,000 iterations are illustrated in Fig. 12(a) and (b) with particle and mesh representation, and clipped views (see Fig. 12(c) and (d)) are presented too. The resulting partitioning sub-domains feature convex and compact shape. The sharp-interface condition is maintained before the relaxation of surface-tension force (see Fig. 12(e)). While some disturbances are observed at the geometry corners after the system is fully relaxed (see Fig. 12(a) and (f)), the communication overhead is around 0.15% (see 13(b)).

The histories of $\theta_{<10}$, $\theta_{<20}$, $\theta_{<30}$ and $\theta_{<40}$ (Fig. 13(e)) show good convergence of the simulation. The number of slivers is ignorable, i.e. 1 tetrahedron has dihedral angle smaller than 10°, comparing to total number of elements generated (1,847,416). The histograms of dihedral angle (13(c)) and radius ratio (13(d)) exhibit good mesh quality too (see Table 5).

5. Conclusions

In this paper, we have developed a consistent parallel mesh generation method with a multi-phase SPH formulation and particle relaxation strategy. The objectives of partitioning the domain, optimizing communication volume and improving mesh quality are achieved consistently by solving the same set of physics-motivated governing equations. The main contributions of the paper are:

(1) A unified target density function is defined to characterize the targets of both the domain decomposition and the mesh generation. The target density function can be any smooth scalar field considering various geometrical features and user-defined inputs. By utilizing a background Cartesian mesh and level-set function, the total number of mesh vertices and target mass for each sub-domain can be determined a priori;

(2) A parallelization strategy is developed and a set of physics motivated governing equations are proposed to achieve all underlying targets consistently. A surface tension model is introduced to the previous particle-based mesh generator [12] to handle the additional target of optimizing the communication volume in a parallel environment. During the domain decomposition stage, the mesh quality is improved simultaneously in the interior region of each sub-domain. Once a steady state is achieved, the mesh quality near the interface region is optimized by gradually alleviating the surface tension force.

(3) A multi-phase SPH formulation is utilized to solve the governing equations. The previously-developed mesh generator [12] is extended to higher dimensions and parallelized with both MPI and TBB technique. Numerical results demonstrate that the resulting sub-domains feature compact and regularized shape, and high-quality mesh is generated simultaneously. The communication overhead caused by the optimization of mesh quality near the interface is limited even in cases with complex geometry and large spacial adaptivity;

(4) With the proposed parallel mesh generation method, high quality triangle/tetrahedron mesh can be generated without the need of constructing Delaunay Triangulation/Tetrahedralization explicitly. Since only local information are required during the simulation and the same set of governing equations are solved for all the particles, the proposed method features high consistency and code reusability. Benefiting from the scalable parallel environment designed previously in [38], the mesh generation procedure is able to exploit both fine-grained and coarse-grained parallelization.

Given all the above-mentioned advantages, the current particle-based method is still considerably more expensive than the state-of-the-art Delaunay-based methods. In the future, we are looking forward to extend the proposed algorithm to GPU-based architectures to achieve higher concurrency and performance. Although SPH is a CPU intensive method, it is particularly suitable for the parallelization on manycore processors and can achieve very

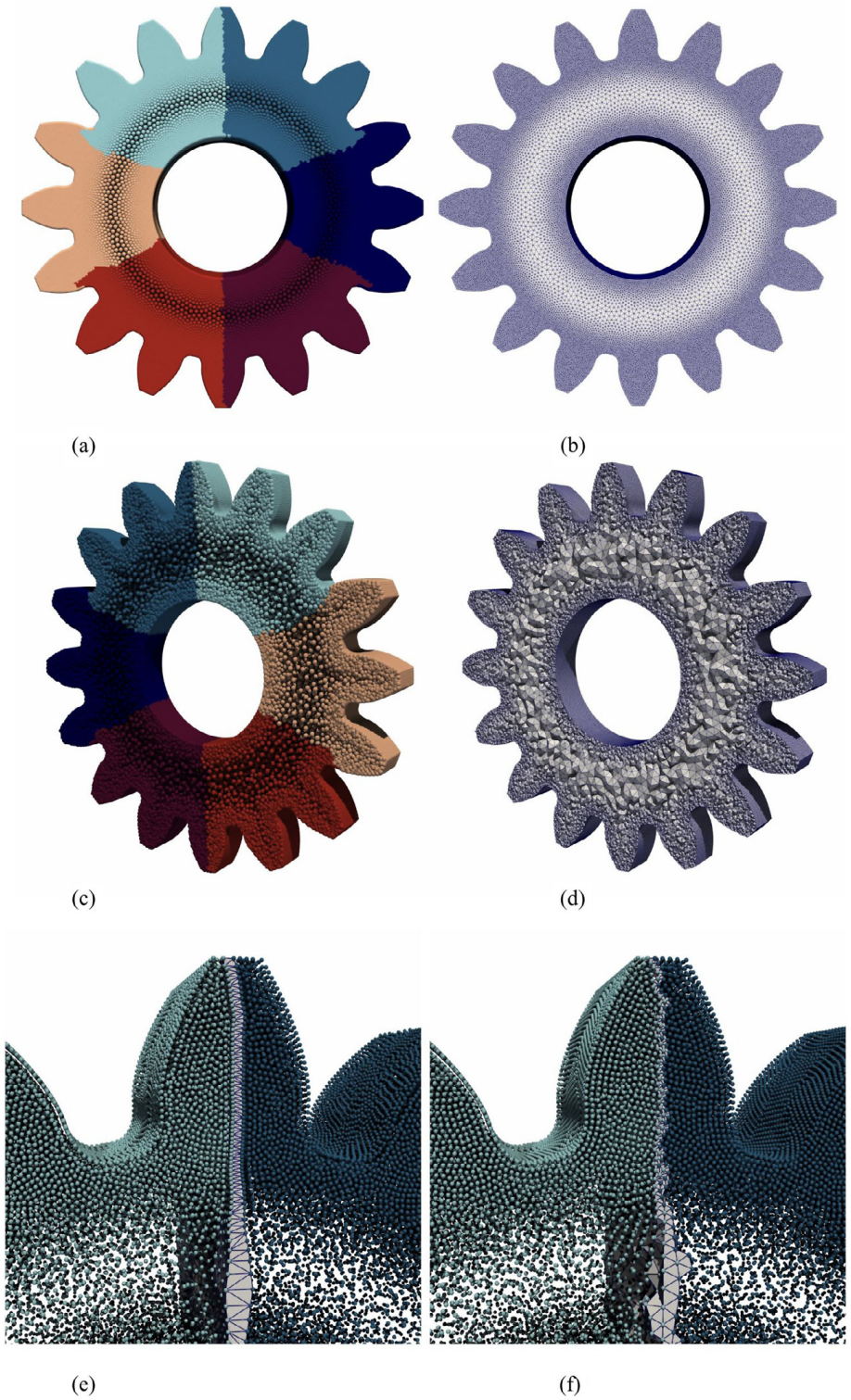


Fig. 12. Simulation result after 250,000 iterations plotted with (a) particles and (b) surfaces with edges. (c) Clipped view of (a). (d) Clipped view of (b). Particle distribution (c) before removing surface tension force and (d) after fully relaxed. Particles are rendered by sub-domain colors. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

high performance. After extended to GPU architecture, a speedup of one to two orders of magnitude normally can be achieved.

Moreover, more studies on initial particle seeding strategies will be carried out. It would be interesting to see that the proposed method is coupled with other existing meshers or point cloud generators which feature high

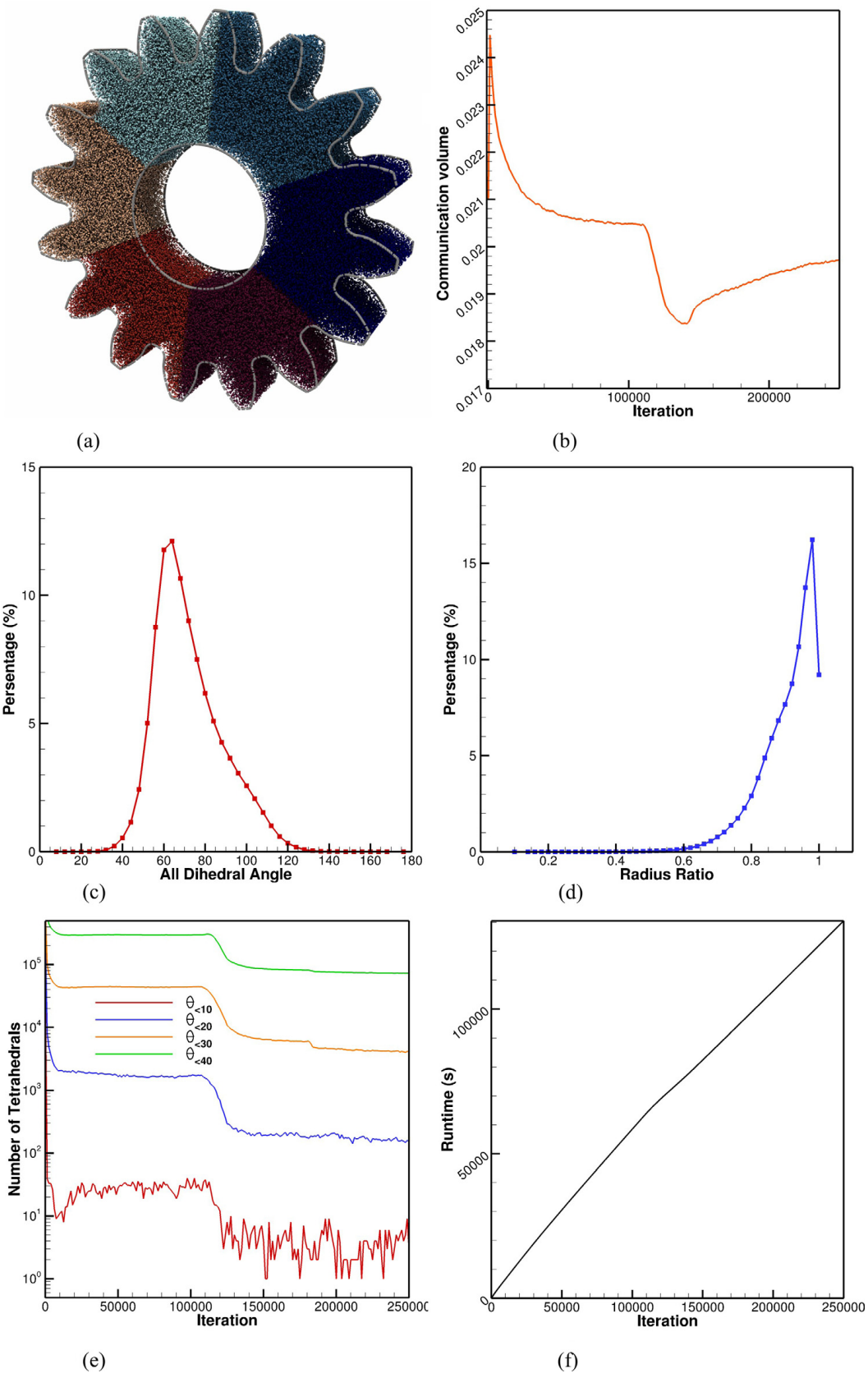


Fig. 13. (a) Initial seeding of particles. (b) History of communication volume. (c) Histogram of the dihedral angle distribution. (d) Histogram of the radius ratio distribution. (e) Convergence history of number of tetrahedra with minimum dihedral angle smaller than 10°, 20°, 30° and 40°. (f) History of runtime.

performance in terms of generating an initial particle distribution, e.g. the Advancing Front Point Generation Method [58]. Namely, the existing mesher is executed first to generate the initial particle position following the target density function and then the proposed method is employed to optimize the particle distribution for better mesh quality. Consequently, the runtime required for our method to achieve convergence will be reduced significantly.

Acknowledgments

Zhe Ji is partially supported by China Scholarship Council (No. 201506290038). Xiangyu Hu acknowledges funding Deutsche Forschungsgemeinschaft (HU1527/10-1 and HU1527/12-1). The computational resources are provided by Leibniz-Rechenzentrum der Bayerische Akademie der Wissenschaften, Munchen (LRZ).

References

[1] P.J. Frey, P.-L. George, Mesh Generation: Application to Finite Elements, ISTE, 2007.
[2] M.A. Park, J.A. Krakos, T. Michal, A. Loseille, J.J. Alonso, Unstructured grid adaptation: Status, potential impacts, and recommended investments toward CFD Vision 2030, 2016.
[3] D. Feng, C. Tsolakis, A.N. Chernikov, N.P. Chrisochoides, Scalable 3D hybrid parallel Delaunay image-to-mesh conversion algorithm for distributed shared memory architectures, Procedia Eng. 124 (2015) 18–30.
[4] D. Feng, A.N. Chernikov, N.P. Chrisochoides, A hybrid parallel Delaunay image-to-mesh conversion algorithm scalable on distributed-memory clusters, Proc. Eng. 163 (2016) 59–71.
[5] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, D. Mavriplis, CFD vision 2030 study: a path to revolutionary computational aerosciences, 2014.
[6] J. Schöberl, NETGEN An advancing front 2D/3D-mesh generator based on abstract rules, Comput. Vis. Sci. 1 (1) (1997) 41–52.
[7] R. Löhner, Recent advances in parallel advancing front grid generation, Arch. Comput. Methods Eng. 21 (2) (2014) 127–140.
[8] J.R. Shewchuk, Delaunay refinement algorithms for triangular mesh generation, Comput. Geom. 22 (1–3) (2002) 21–74.
[9] L.P. Chew, Guaranteed-quality delaunay meshing in 3D (short version), in: Proceedings of the Thirteenth Annual Symposium on Computational Geometry, ACM, 1997, pp. 391–393.
[10] S. Ni, Z. Zhong, Y. Liu, W. Wang, Z. Chen, X. Guo, Sliver-suppressing tetrahedral mesh optimization with gradient-based shape matching energy, Comput. Aided Geom. Design 52 (2017) 247–261.
[11] Q. Du, V. Faber, M. Gunzburger, Centroidal voronoi tessellations: Applications and algorithms, SIAM Rev. 41 (4) (1999) 637–676.
[12] L. Fu, L. Han, X.Y. Hu, N.A. Adams, An isotropic unstructured mesh generation method based on a fluid relaxation analogy, Comput. Methods Appl. Mech. Engrg. 350 (2019) 396–431.
[13] Z. Zhong, X. Guo, W. Wang, B. Lévy, F. Sun, Y. Liu, W. Mao, Particle-based anisotropic surface meshing, ACM Trans. Graph. 32 (4) (2013) 99.
[14] L. Fu, X.Y. Hu, N.A. Adams, Adaptive anisotropic unstructured mesh generation method based on fluid relaxation analogy, Commun. Comput. Phys. (2019) http://dx.doi.org/10.4208/cicp.OA-2019-0049, in press.
[15] Q. Du, D. Wang, Tetrahedral mesh generation and optimization based on Centroidal Voronoi Tessellations, Internat. J. Numer. Methods Engrg. 56 (9) (2003) 1355–1373.
[16] P.-O. Persson, Mesh size functions for implicit geometries and PDE-based gradient limiting, Eng. Comput. 22 (2) (2006) 95–109.
[17] M.D. Meyer, P. Georgel, R.T. Whitaker, Robust particle systems for curvature dependent sampling of implicit surfaces, in: International Conference on Shape Modeling and Applications 2005, SMI'05, IEEE, 2005, pp. 124–133.
[18] C. Tsolakis, N. Chrisochoides, M.A. Park, A. Loseille, T.R. Michal, Parallel anisotropic unstructured grid adaptation, in: AIAA Scitech 2019 Forum, 2019, p. 1995.
[19] N. Chrisochoides, Parallel mesh generation, in: Numerical Solution of Partial Differential Equations on Parallel Computers, Springer, 2006, pp. 237–264.
[20] H. Rakotoarivelo, F. Ledoux, F. Pommereau, Fine-grained locality-aware parallel scheme for anisotropic mesh adaptation, Proc. Eng. 163 (2016) 123–135.
[21] R. Lohner, J.R. Cebral, Parallel advancing front grid generation, in: International Meshing Roundtable, Sandia National Labs, Citeseer, 1999.
[22] L. Linardakis, N. Chrisochoides, Delaunay decoupling method for parallel guaranteed quality planar mesh refinement, SIAM J. Sci. Comput. 27 (4) (2006) 1394–1423.
[23] A. Loseille, V. Menier, F. Alauzet, Parallel generation of large-size adapted meshes, Procedia Eng. 124 (2015) 57–69.
[24] D. Nave, N. Chrisochoides, L.P. Chew, Guaranteed-quality parallel Delaunay refinement for restricted polyhedral domains, Comput. Geom. 28 (2–3) (2004) 191–215.
[25] J. Galtier, P.L. George, Prepartitioning as a way to mesh subdomains in parallel, in: 5th International Meshing Roundtable, Citeseer, 1996.
[26] L.P. Chew, N. Chrisochoides, F. Sukup, Parallel constrained Delaunay meshing, ASME Appl. Mech. Div. Publ. 220 (1997) 89–96.
[27] D. Feng, A.N. Chernikov, N.P. Chrisochoides, Two-level locality-aware parallel Delaunay image-to-mesh conversion, Parallel Comput. 59 (2016) 60–70.
[28] J.-F. Remacle, V. Bertrand, C. Geuzaine, A two-level multithreaded Delaunay kernel, Procedia Eng. 124 (2015) 6–17.

[29] G. Rokos, G.J. Gorman, K.E. Jensen, P.H. Kelly, Thread parallelism for highly irregular computation in anisotropic mesh adaptation, in: Proceedings of the 3rd International Conference on Exascale Applications and Software, University of Edinburgh, 2015, pp. 103–108.
[30] D. Feng, A.N. Chernikov, N.P. Chrisochoides, A hybrid parallel Delaunay image-to-mesh conversion algorithm scalable on distributed-memory clusters, Comput. Aided Des. 103 (2018) 34–46.
[31] L. Fu, X.Y. Hu, N.A. Adams, A physics-motivated Centroidal Voronoi Particle domain decomposition method, J. Comput. Phys. 335 (2017) 718–735.
[32] M.P. Forum, MPI: A Message-Passing Interface Standard, Tech. Rep., University of Tennessee, Knoxville, TN, USA, 1994.
[33] O.A.R. Board, OpenMP application program interface vsersion 3.0, 2008, URL http://www.openmp.org/mp-documents/spec30.pdf.
[34] J. Nickolls, I. Buck, M. Garland, K. Skadron, Scalable parallel programming with CUDA, Queue 6 (2) (2008) 40–53, http://dx.doi.org/10.1145/1365490.1365500, URL http://doi.acm.org/10.1145/1365490.1365500.
[35] A.J. Crespo, J.M. Domínguez, B.D. Rogers, M. Gómez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, O. García-Feal, DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH), Comput. Phys. Comm. 187 (2015) 204–216.
[36] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1) (1995) 1–19.
[37] P. Incardona, A. Leo, Y. Zaluzhnyi, R. Ramaswamy, I.F. Sbalzarini, OpenFPM: A scalable open framework for particle and particle-mesh codes on parallel computers, Comput. Phys. Comm. (2019).
[38] Z. Ji, L. Fu, X.Y. Hu, N.A. Adams, A new multi-resolution parallel framework for SPH, Comput. Methods Appl. Mech. Engrg. 346 (2019) 1156–1178.
[39] L. Fu, Z. Ji, X.Y. Hu, N.A. Adams, Parallel fast-neighbor-searching and communication strategy for particle-based methods, Eng. Comput. 36 (3) (2019) 899–929.
[40] G. Contreras, M. Martonosi, Characterizing and improving the performance of intel threading building blocks, in: Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on, IEEE, 2008, pp. 57–66.
[41] L. Fu, S. Litvinov, X.Y. Hu, N.A. Adams, A novel partitioning method for block-structured adaptive meshes, J. Comput. Phys. 341 (2017) 447–473.
[42] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1) (1988) 12–49.
[43] L. Piegl, On NURBS: a survey, IEEE Comput. Graph. Appl. 11 (1) (1991) 55–71.
[44] L. Han, X. 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.
[45] W.E. Lorensen, H.E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, in: ACM Siggraph Computer Graphics, Vol. 21, No. 4, ACM, 1987, pp. 163–169.
[46] J. Brackbill, D.B. Kothe, C. Zemach, A continuum method for modeling surface tension, J. Comput. Phys. 100 (2) (1992) 335–354.
[47] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, G. Zanetti, Modelling merging and fragmentation in multiphase flows with SURFER, J. Comput. Phys. 113 (1) (1994) 134–147.
[48] H. Si, TetGen, a Delaunay-based quality tetrahedral mesh generator, ACM Trans. Math. Softw. 41 (2) (2015) 11.
[49] D.P. Starinshak, J. Owen, J. Johnson, A new parallel algorithm for constructing Voronoi tessellations from distributed input data, Comput. Phys. Commun. 185 (12) (2014) 3204–3214.
[50] R. Chen, C. Gotsman, Localizing the delaunay triangulation and its parallel implementation, in: Transactions on Computational Science XX, Springer, 2013, pp. 39–55.
[51] Z. Ji, L. Fu, X.Y. Hu, N.A. Adams, A Lagrangian Inertial Centroidal Voronoi Particle method for dynamic load balancing in particle-based simulations, Comput. Phys. Comm. 239 (2019) 53–63.
[52] L. Fu, Z. Ji, An optimal particle setup method with Centroidal Voronoi Particle dynamics, Comput. Phys. Comm. 234 (2019) 72–92.
[53] S. Adami, X. Hu, N.A. Adams, A new surface-tension formulation for multi-phase SPH using a reproducing divergence approximation, J. Comput. Phys. 229 (13) (2010) 5011–5021.
[54] H. Meyerhenke, B. Monien, S. Schamberger, Graph partitioning and disturbed diffusion, Parallel Comput. 35 (10–11) (2009) 544–569.
[55] G. Turk, M. Levoy, Zippered polygon meshes from range images, in: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, ACM, 1994, pp. 311–318.
[56] E. Barnes, N. Sloane, The optimal lattice quantizer in three dimensions, SIAM J. Algebr. Discrete Methods 4 (1) (1983) 30–41.
[57] B.R. Hoehn, P. Oster, T. Tobie, K. Michaelis, Test methods for gear lubricants, Goriva i maziva 47 (2) (2008) 141–152.
[58] R. Löhner, E. Onate, An advancing front point generation technique, Commun. Numer. Methods. Eng. 14 (12) (1998) 1097–1108.