Front-Tracking mesh adaptation for the simulation of two-phase flows with coalescence and breakup

Paul Regnault

LABORATOIRE MODÉLISATION ET SIMULATION MULTI ÉCHELLE
Sous la co-tutelle de :
CNRS
UNIVERSITÉ GUSTAVE EIFFEL
UPEC • UNIVERSITÉ PARIS-EST CRÉTEIL

UNIVERSITÉ PARIS-EST

ÉCOLE DOCTORALE SCIENCES, INGÉNIERIE ET ENVIRONNEMENT (SIE) - ED531

LABORATOIRE MODÉLISATION ET SIMULATION MULTI-ECHELLE (MSME, UMR 8208 CNRS)
ÉQUIPE TRANSFERTS DE CHALEUR ET DE MATIÈRE

THÈSE DE DOCTORAT
Discipline : Mécanique des fluides

Présentée par

Paul REGNAULT

Dirigée par

Stéphane VINCENT et **Eric CHÉNIER**

Pour obtenir le grade universitaire de
DOCTEUR de l'UNIVERSITÉ PARIS-EST

Front-Tracking mesh adaptation for the simulation of two-phase flows with coalescence and breakup

---

Présentée et soutenue publiquement le 19 décembre 2023 devant le jury composé de :

Jean-Luc ESTIVALÈZES	Professeur associé	ONERA	Rapporteur
Adrien TOUTANT	Professeur des universités	PROMES	Rapporteur
Taraneh SAYADI	Chargée de recherche	∂'Alembert - Sorbonne Université	Examinatrice
Tavares MATHILDE	Docteure	LadHyX	Examinatrice
Stéphane ZALESKI	Professeur des universités	∂'Alembert - Sorbonne Université	Examinateur
Stéphane VINCENT	Professeur des universités	MSME	Directeur de thèse
Eric CHÉNIER	Maître de conférences	MSME	Co-directeur de thèse

Adaptation de maillage pour la simulation d'écoulements diphasiques avec coalescence et rupture dans le cadre de la méthode Front-Tracking

Résumé

Dans le cadre des écoulements diphasiques à phases séparées, ce travail porte sur la gestion dynamique du maillage de l'interface (constitué de triangles en 3D) et son impact sur l'approximation des propriétés géométriques que sont la position et la courbure. Les équations de conservation de la mécanique des fluides sont résolues sur des grille fixes, décalées et structurées. L'interface est suivie de façon lagrangienne au cours du temps avec un maillage mobile et déformable : on parle de méthode de type « Front-Tracking ». En plus des opérations de remaillage classiques (suppression et échange d'arêtes, insertion de sommets notamment), on étudiera l'adaptation du maillage à la courbure de l'interface et l'utilisation d'une approximation polynomiale pour améliorer l'insertion de sommets ou la suppression d'arêtes. Ces méthodes sont évaluées sur des surfaces analytiques mobiles et déformables, sans résolution des équations de Navier-Stokes ni changement topologique. Dans les écoulements diphasiques, des changements topologiques peuvent avoir lieu : la coalescence et la rupture. Nous proposons une méthode de coalescence et une méthode de rupture d'interface. Ces deux méthodes sont activées selon des critères de distance et sont basées uniquement sur le maillage de l'interface, sans recourir au maillage eulérien. Ces méthodes sont utilisées sur des configurations numériques et expérimentales de la littérature pour apprécier leur robustesse et leurs performances.

Mots-clés : Front-tracking, diphasique, maillage d'interface, cas test analytique instationnaire, courbure, coalescence, rupture.

Front-Tracking mesh adaptation for the simulation of two-phase flows with coalescence and breakup

Abstract

In the context of two-phase flows with separated phases, this work focuses on dynamic management of the interface mesh (made up of connected triangles in 3D) and its impact on the approximation of geometrical properties that are position and curvature. The conservation equations of fluid mechanics are solved on fixed, staggered and structured grids. The interface is tracked in a Lagrangian fashion with a moving and deformable mesh: this method is known as the "Front-Tracking" method. In addition to classical remeshing operations (edge splitting, collapsing and swapping for instance), we will study the adaptation of the mesh to the curvature of the interface and the use of polynomial approximation to improve edge splitting and collapsing. These methods are evaluated on analytical, mobile and deformable surfaces, with neither the resolution of the Navier-Stokes equations nor topological changes. In two-phase flows, topological changes may happen: coalescence and breakup. We propose a method for coalescence and a method for breakup. These two methods are activated by distance criteria and rely only on the interface mesh, without resorting to the Eulerian mesh. These methods are employed on numerical and experimental configurations from the literature to appreciate their robustness and performances.

Keywords: Front-Tracking, two-phase flows, interface mesh, unsteady analytical test case, curvature, coalescence, breakup.

Remerciements

Je tiens à remercier mes rapporteurs, M. Estivalèzes et M. Toutant pour l'examen de mon manuscrit de thèse. Mes remerciements vont ensuite à Mme Sayadi, Mme Tavares et M. Zaleski d'avoir accepté de faire partie de mon jury. Je tiens à remercier M. Eymard, Mme Sayadi et Mme Potdevin d'avoir accepté de faire partie de mes comités de thèse et pour leurs conseils. Je tiens à remercier mes directeurs de thèse M. Chénier et M. Vincent pour le sujet de thèse et le temps qu'ils m'ont consacré. Les échanges et les suggestions m'ont permis de progresser, d'améliorer le travail et de prendre du recul.

Je tiens à remercier M. Nicolas pour les relectures et les échanges. Je tiens à remercier l'IDRIS, le TGCC et le CINES pour l'accès aux ressources de calcul et les formations proposées. Je tiens à remercier les membres du laboratoire MSMSE, en particulier Mme Boston pour les démarches administratives et Guillaume pour son aide sur les stations de calcul de MSME. Mes remerciements vont également aux permanents de l'équipe TCM, pour les échanges notamment sur le code FUGU : M. Trouette, Can, Vincent, Amine et Mme Ould-Rouiss. Je tiens à remercier les docteurs et doctorants de l'équipe TCM : Dahia, Syphax, Mohamed, Abdelhamid, Georges, Ivan, Nour, Faina, Andrew, Clément, Tanguy, Nadjma, Abderaouf, Désir-André et Samy. J'ai apprécié les échanges avec les membres des autres équipes, en particulier avec Matthieu, Robin, Amritesh, Souhail, Rosa et Houda. J'ai également apprécié les échanges avec les membres du CFD et de l'EDSIE. Je tiens à remercier ma famille pour tout, en particulier mes grands-parents, mes parents, Mathilde et Anne.

Contents

Part I

Introduction and related work

1 Introduction

A quick tour of two-phase flows and their dedicated numerical methods is given.

Two-phase flows involve two phases, such as air and water, separated by an interface. A wide variety of such flows can be encountered in the environment and the industry. It is often important to understand the physics of those flows and to describe them quantitatively for security or efficiency reasons. Experiments and numerical simulations are complementary in understanding these flows. Numerical simulation may give access to information otherwise inaccessible with experiments, since visibility issues may prevent precise measurements: for dense droplet-laden flows for instance, or even for only one bubble, the difference in refractive indices makes it difficult to see its shape (Hua and Lou, 2007). Experiments are also necessary to validate the numerical methods.

Atomization is a process where a liquid jet disintegrates into drops and ligaments, thus forming a spray. Sprays are rife with coalescence and breakup, induced by drop collisions for instance. Sprays have many applications, such as production of powdered milk (Finotello et al., 2017), agriculture, painting, boilers, and internal combustion. The atomization of fuel is an important step in most engines, as the drops thus formed strongly influence the efficiency of combustion and emission of pollutants. Many works aim at describing these processes, such as in Reitz and Bracco (1982); Desjardins et al. (2008); Fuster et al. (2009); Shinjo and Umemura (2010); Herrmann (2011); Bo et al. (2011); Le Chenadec and Pitsch (2013); Ling et al. (2015); Janodet et al. (2022).

Two-phase flows have many other applications: for printing, the breakup of asymmetric ligaments plays an important role, as it generates satellite droplets which may blur the result (Planchette et al., 2019). In heat exchangers, boiling crisis happens when the local heat flux reaches a maximum, the Critical Heat Flux (CHF), because of the formation of a film and the temperature of the wall rises, which may damage the exchanger. In all of these cases, the aim is to get a better understanding of these flows to control the outcome of a process or predict an event.

Binary drop collision was originally studied in order to better understand rain, when droplets in clouds generate raindrops (Jayaratne et al., 1997). Then, drop collision was studied for sprays, as the different collision outcomes influence the drop size distribution in a spray (Ashgriz and Givi, 1987; Brenn and Frohn, 1989). The Weber number (We) is the ratio of inertia over surface tension, in some collision studies it is defined as $\mathrm{We} = \rho U D/\sigma$, where ρ is the density of the liquid, U the relative velocity, D the initial drop diameter and σ the surface tension coefficient. Experiments have revealed five regimes of binary droplet collisions: coalescence with minor deformation, bouncing, coalescence with major deformation, reflexive separation and stretching separation (Ashgriz and Poo, 1990; Qian and Law, 1997). Pan et al. (2009) went up to a Weber of 5100 for water, and identified new outcomes: "fingering lamella", "separation after fingering", "breakup of outer fingers", and "prompt splattering into multiple secondary droplets". Knowing the different collision regimes may help model the spray on a larger scale with Sub-Grid Scale (SGS) models for instance. Two-phase flows are indeed mostly multi-scale: for instance, a film drainage process intervenes in drop collision. The difference in scales is such that the simulation of many two-phase flows proves to be difficult because of the computational power needed. Some numerical works rely on sub-grid models to include small-scale physics (Kwakkel et al., 2013). Many works rely on Adaptive Mesh Refinement (AMR) to save up computational resources. In this work, we are concerned with numerical methods to simulate two-phase flows, and especially interface tracking methods which plays an important role. There are several approaches to simulate two-phase flows, and they will be shortly described afterwards. We present briefly the Navier-Stokes equations, the One-Fluid Model (OFM) and the associated interface tracking methods.

1.1 Hypotheses

We deal with two incompressible and isothermal fluids with different physical properties (density and viscosity). The two phases are separated. A surface tension coefficient σ is used to express the surface tension force between the two fluids. The two fluids are considered Newtonian: we suppose that there is a linear relation between the stress tensor and the strain rate tensor. For instance, we will try to simulate air and water but not polymers. The two fluids are isothermal in our computations, so their density and viscosity are considered constant in each phase. The two fluids are considered non-miscible: we suppose that there exist an interface separating them, and we neglect its thickness.

1.2 Fixed and deformable grids, meshless methods

The methods used to solve multiphase flow problems can be distinguished in two main classes. In the first class, the interface is used as a boundary between two domains with distinct phases. This approach requires automatic mesh generators to adapt the mesh to the interface as it deforms, this is a body-fitted mesh. It is also possible to use a mesh without conforming to the interface, with either structured or unstructured meshes, and fixed meshes or AMR. We may also cite the Smoothed particle hydrodynamics (SPH) method, a meshless Lagrangian approach. Eirís et al. (2023) reviews the main approaches and classifies the existing approaches in three categories: mesh-based, meshless-Finite Volume (FV) and SPH methods. The Lattice Boltzmann (LB) method has also been applied to two-phase flows (Gunstensen et al., 1991). The eXtreme Mesh deformation approach (X-MESH) relays the interface between nodes and allows elements with zero measure with a Finite Element Method (FEM) (Moës et al., 2023).

1.3 Navier-Stokes equations

In this work, we consider separated two-phase flows. In a computational domain, the two phases are separated by an interface. There are two main approaches for the simulation of two-phase flows. The first, called the two-fluid model, consists in solving two sets of Navier-Stokes equations, one in each phase. The two solutions are coupled with jump conditions across the interface. The second approach is called the One-Fluid Model. The Eulerian mesh does not conform to the topology of the interface, this kind of method belongs to Fictitious Domain (FD) methods (Khadra et al., 2000).

1.3.1 Two-Fluid model

In each phase $k = 1, 2$, the Navier-Stokes equations are thus written:

$$\nabla \,.\, \mathbf{u}_k = 0 \tag{1.1a}$$

$$\frac{\partial \rho_k \mathbf{u}_k}{\partial t} + \nabla \,.\, (\rho_k \mathbf{u}_k \otimes \mathbf{u}_k) = \rho_k \mathbf{g} + \nabla \,.\, \mathbf{T}_k \tag{1.1b}$$

with $\mathbf{T}_k \equiv -p_k \mathbf{I} + 2\mu_k \mathbf{D}_k$ the stress tensor and $\mathbf{D}_k = \frac{1}{2}(\bar{\nabla}\mathbf{u}_k + \bar{\nabla}\mathbf{u}_k{}^T)$ the deformation tensor associated with phase k.

The jump conditions across the interface are thus defined:

$$\mathbf{u}_1 = \mathbf{u}_2 \tag{1.2a}$$

$$\mathbf{n} \cdot (\mathbf{T}_1 - \mathbf{T}_2) \cdot \mathbf{n} = \sigma\kappa \tag{1.2b}$$

$$\mathbf{t} \cdot (\mathbf{T}_1 - \mathbf{T}_2) \cdot \mathbf{n} = 0 \tag{1.2c}$$

with σ the surface tension coefficient, κ twice the mean curvature, $\mathbf{n}$ and $\mathbf{t}$ the unit normal and tangent to the interface.

For instance, the Arbitrary Lagrangian-Eulerian (ALE) adapts at each iteration an unstructured mesh to the deforming interface, and the mesh does not move at the same velocity as the flow.

1.3.2 One-Fluid model

In this work, the two-phase flow is described by the OFM developed by Kataoka (1986). The Navier-Stokes equations are solved for a single equivalent fluid, with an additional source term in the momentum equation: the surface tension. A method is also required to follow the location of the interface between the two fluids. The most commonly used interface tracking methods are the Volume Of Fluid (VOF) method and the Level-Set (LS) method. The scalar function C is the color function, the volume fraction locating the interface crossing a given control volume Ω_{ijk}. It equals 1 if the mixture contains only fluid 1 and 0 if it is only comprised of fluid 2. The properties in the domain such as the density and the viscosity may be determined with an arithmetic average:

$$\rho = C\rho_1 + (1-C)\rho_2 \tag{1.3}$$

$$\mu = C\mu_1 + (1-C)\mu_2 \tag{1.4}$$

For the viscosity, it may be relevant to use an harmonic average:

$$\mu = \frac{\mu_1\mu_2}{C\mu_2 + (1-C)\mu_1} \tag{1.5}$$

We may also cite the discontinuous average:

$$\mu = \begin{cases} \mu_1 & C > 0.5 \\ \mu_2 & C \leq 0.5 \end{cases} \tag{1.6}$$

Benkenida and Magnaudet (2000) proposed a scheme for viscosity combining arithmetic and harmonic averages. By splitting the viscous stress tensor, another approach of mixed viscosity average is possible. Caltagirone and Vincent (2001) proposed a new formulation of the viscous stress tensor, which may be thus written for incompressible flows, with three pseudo-tensors for elongation, shearing, and rotation:

$$2\mu\mathbf{D} = \kappa \begin{pmatrix} \frac{\partial u}{\partial x} & 0 & 0 \\ 0 & \frac{\partial v}{\partial y} & 0 \\ 0 & 0 & \frac{\partial w}{\partial z} \end{pmatrix} + \zeta \begin{pmatrix} 0 & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & 0 & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & 0 \end{pmatrix} - \eta \begin{pmatrix} 0 & \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} & \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \\ \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} & 0 & \frac{\partial v}{\partial z} - \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial x} - \frac{\partial u}{\partial z} & \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} & 0 \end{pmatrix} \tag{1.7}$$

where new viscosity coefficients are defined: $\kappa = 2\mu$ the elongation viscosity, located at the pressure node, $\zeta = 2\mu$ the shearing viscosity and $\eta = \mu$ the rotation viscosity. Vincent et al. (2014) used this formulation of the viscous stress tensor (eq. 1.7) to propose a mixed viscosity average. The elongation viscosity is obtained with an arithmetic average based on the volume fraction at the pressure nodes. The shearing and rotation viscosities are computed with an harmonic average, based on a linear interpolation of the volume fraction.

Now, the Navier-Stokes equations can be thus written in the OFM:

$$\nabla \,.\, \mathbf{U} = 0 \tag{1.8}$$

$$\frac{\partial \rho\mathbf{U}}{\partial t} + \nabla \,.\, (\rho\mathbf{U} \otimes \mathbf{U}) = -\nabla p + \nabla \cdot (2\mu\mathbf{D}) + \rho\mathbf{g} + \mathbf{S}_\sigma \tag{1.9}$$

with $\mathbf{D} = \frac{1}{2}\left(\nabla\mathbf{U} + \nabla^t\mathbf{U}\right)$ and $\mathbf{S}_\sigma$ is the surface tension force.

In the VOF and LS methods, a transport equation is solved for the volume fraction as well (eqs. 1.10 and 1.12, presented subsequently). It is equivalent to the Lagrangian equation used in the Front-Tracking method to advect the markers (eq. 1.13, presented below).

Summary. We suppose that the density ρ and the dynamic viscosity μ are constant in each phase and that the flow is incompressible. A single set of equations is written for both phases (One-Fluid Model).

In the OFM, interface modelling methods may in turn be divided into two classes: implicit and explicit representations, respectively called Front-Capturing and Front-Tracking methods. The location of the interface is known respectively through a scalar field or markers.

1.3.2.1 Front-Capturing

Front-Capturing methods are also referred to as implicit methods because the position of the interface is known implicitly by the isosurface of a function. A supplementary advection scheme is used to transport the interface. One issue is to avoid smearing the interface. The VOF and LS methods are Front-Capturing methods, information about the interface are possessed by the mesh cell.

Volume Of Fluid. The Volume Of Fluid (VOF) method was invented by Hirt and Nichols (1981). It uses an additional transport equation for the phase function in order to determine the location of the interface:

$$\frac{\partial C}{\partial t} + \mathbf{U} \,.\, \nabla C = 0 \tag{1.10}$$

The scalar function is the volume fraction, also known as color function, it equals 1 if the mixture is only filled with fluid 1 and 0 if it is only composed of fluid 2. This method follows the ratio of the volume of both fluids in each cell, and an additional step is required to reconstruct the interface, but it is not determined precisely in each cell in the classical VOF method, contrary to the Front-Tracking method. It conserves generally well the volume, especially in Weymouth and Yue (2010), but the shape of the interface is generally more difficult to retain.

Level-Set method. The classical Level-Set method represents the interface with the signed distance function Φ:

$$\Phi = \pm \min_{y \in \Gamma} \|x - y\| \tag{1.11}$$

The signed distance is positive on one side of the interface and negative on the other side, and satisfies $|\nabla \Phi| = 1$. An additional transport equation is required:

$$\frac{\partial \Phi}{\partial t} + \mathbf{U} \,.\, \nabla \Phi = 0 \tag{1.12}$$

It is simpler to implement than the VOF method but the volume of each phase is not conserved in the classical LS method. In many implementations, the distance function has to be re-initialized. Sussman and Puckett (2000) proposed to couple the VOF and LS to benefit from their respective advantage: volume conservation and curvature evaluation. The Conservative Level-Set (CLS) has been developed to improve the volume conservation with a smeared out Heaviside function and a modification of the reinitialization step (Olsson and Kreiss, 2005; Olsson et al., 2007). Desjardins et al. (2008) further improved it by developing the Accurate Conservative Level-Set (ACLS). Chiodi and Desjardins (2017) improved the accuracy of the reinitialization procedure. A variant of the LS method is the Diffuse Interface Model (DIM) (Yue et al., 2004).

1.3.2.2 Front-Tracking

The Front-Tracking method is a an explicit method. Explicit methods describe the interface between the two fluids with a representation of lower dimension than the problem. Each vertex of coordinate $\mathbf{x}_m$ represents the surface and it is advected by the following transport equation:

$$\frac{d\mathbf{x}_m}{dt} = \mathbf{V}_m \tag{1.13}$$

Contrary to the VOF and LS methods, information are possessed by vertices in the Front-Tracking method. The size of the triangles is a priori independent from the size of the Eulerian mesh, but for numerical reasons (interpolations, memory cost to cite but a few) it is often limited. An Eulerian grid is used to compute the velocity field and the velocity is then interpolated on the front.

The main difficulty in 3D is the implementation of a robust algorithm for the merging and breakup of interfaces. The markers may or not be connected to form elements (segments in 2D and triangles in 3D). The difficulty is increased with the use of connected markers. Contrary to Shin and Juric (2002), connected markers were used in this work to simplify the computation of the surface curvature and normal. Coalescence and breakup are not treated automatically in our implementation.

1.3.3 Ghost-Fluid method

Fedkiw et al. (1999); Kang et al. (2000) developed the Ghost-Fluid Method (GFM) to avoid spreading the interface. Jump conditions are imposed at the interface instead on the different variables: pressure, density, viscosity. Ghost cells are used to extrapolate variables on the other of the interface before computing their derivatives. Instead of including the surface tension as a force in the momentum equation, it is included as a pressure jump in the discretization of the pressure gradient operator. Terashima and Tryggvason (2009); Bo et al. (2011) simulate compressible two-phase flow with a Front-Tracking and a GFM.

1.4 Summary

Front-Capturing methods use the Eulerian grid to describe the interface while the Front-Tracking method uses a deformable mesh to describe the interface explicitly. As a result, Front-Capturing methods in their simpler implementations treat coalescence automatically. For instance two bubbles at a distance smaller than the grid size coalesce. We elected to work with the Front-Tracking method, which is an explicit method. Coalescence and breakup are not treated automatically.

1.5 Numerical framework

We use a Finite Volume Method (FVM) with staggered grids and a Lagrangian mesh as as depicted in fig. 2.1.

1.5.1 Pressure-velocity coupling

The pressure is unknown, it is indirectly specified in the continuity equation. The pressure influences the velocity field in the momentum equation, and it is correct when the resulting velocity field satisfies the continuity equation. Thus one may either solve a coupled system (mass and momentum conserving equations) or decouple velocity and pressure (Chorin, 1968). In this work, we use the fully-coupled solver presented in El Ouafa et al. (2023). It relies on a simultaneous resolution of the linearized momentum and continuity equations with a BiCGStab(2) iterative solver (Dongarra et al., 1998). Penalization is employed for the boundary conditions (Angot et al., 1999).

1.5.2 Momentum conserving

We use a Momentum Conserving method derived from Nangia et al. (2019), yet where both not only the volume fraction but also the momentum is transported in a more consistent way to improve the stability of the code (Elouafa, 2022). The Third-order Strong Stability-Preserving Runge-Kutta method (SSP-RK3) (Gottlieb et al., 2001) and Convergent and Universally Bounded Interpolation Scheme for the Treatment of Advection (CUBISTA) (Alves et al., 2003) schemes are used.

1.5.3 Time step criteria

There are many time-step conditions in the literature aside from the Courant-Friedrichs-Lewy condition (CFL), based on surface tension, viscosity and gravity. According to Brackbill et al. (1992), there is a stability condition for the explicit treatment of surface tension:

$$\Delta t < \sqrt{\frac{(\rho_1 + \rho_2)\Delta x^3}{4\pi\sigma}} \tag{1.14}$$

Kang et al. (2000) adapts the timestep Δt to:

$$\Delta t\left(C_{conv} + C_{visc} + \sqrt{(C_{conv} + C_{visc})^2 + 4\left({C_{grav}}^2 + {C_{capl}}^2\right)}\right) \leq 2CFL \tag{1.15}$$

with:

$$C_{conv} = \max\left(\frac{\|u\|_\infty}{\Delta x}, \frac{\|v\|_\infty}{\Delta y}, \frac{\|w\|_\infty}{\Delta z}\right) \tag{1.16}$$

$$C_{visc} = \max\left(\frac{\mu_1}{\rho_1}, \frac{\mu_2}{\rho_2}\right)\left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} + \frac{2}{\Delta z^2}\right) \tag{1.17}$$

$$C_{grav} = \sqrt{\max\left(\frac{|g_x|}{\Delta_x}, \frac{|g_y|}{\Delta_y}, \frac{|g_z|}{\Delta_z}\right)} \tag{1.18}$$

$$C_{capl} = \sqrt{\frac{\sigma\|\kappa\|_\infty}{\min(\rho_1, \rho_2)\Delta x^2}} \tag{1.19}$$

1.5.3.1 Summary

We use a Finite Volume Method on a fixed structured grid. A Front-Tracking method is used to track the interface between the two fluids. A coupled solver is used with a momentum preserving method.

1.5.4 Results

When not mentioned otherwise, we work in SI units.

1.6 Outline

In the context of two-phase flows with separated phases, this work focuses on dynamic management of the interface mesh (made up of connected triangles in 3D) and its impact on the approximation of geometrical properties that are position and curvature. The conservation equations of fluid mechanics are solved on fixed, staggered and structured grids. The interface is tracked in a Lagrangian fashion with a moving and deformable mesh: this method is known as the "Front-Tracking" method. In addition to classical remeshing operations (edge splitting, collapsing and swapping for instance), we will study the adaptation of the mesh to the curvature of the interface and the use of polynomial approximation to improve edge splitting and collapsing. These methods are evaluated on analytical, mobile and deformable surfaces, with neither the resolution of the Navier-Stokes equations nor topological changes. In two-phase flows, topological changes may happen: coalescence and breakup. We propose a method for coalescence and a method for breakup. These two methods are activated by distance criteria and rely only on the interface mesh, without resorting to the Eulerian mesh. These methods are employed on numerical and experimental configurations from the literature to appreciate their robustness and performances. The influence of the coupling of the Lagrangian mesh with the Navier-Stokes solver is studied on simulations of rising bubbles. This document is organized in several parts, each part is divided in turn in multiple chapters.

Part I. comprises this introduction and presents a non-exhaustive list of methods to simulate two-phase flows. The main features of the code are presented. Afterwards, the different steps of the Front-Tracking method are described.

Part II. presents the remeshing procedure. The remeshing steps and criteria are evaluated on unsteady analytical test cases.

Part III. presents a comparison of the developed Front-Tracking method with a VOF method to appreciate the influence of the coupling of the Lagrangian mesh with the Eulerian grid. Numerical methods for coalescence and breakup are then reviewed. We then present a method for coalescence and breakup and use them in test cases.

Part IV. draws a conclusion from the results and outlines perspectives of of this work.

Part V. contains the appendices, the list of acronyms, the nomenclature and the bibliography.

2 Front-Tracking method

Previously we elected to work on the Front-Tracking method to simulate two-phase flows. Now, we will give a quick overview of the Front-Tracking method and whence it came. We then introduce our notations for a triangle mesh and present the main features of the Front-Tracking method.

2.1 Front-Tracking introduction

This section provides an overview of the Front-Tracking method and introduces the first choices that were made for its implementation in the home-made code Fugu, developed by the MSME laboratory at Gustave Eiffel University. The code uses staggered grids as depicted in fig. 2.1. The Front-Tracking method is an interface tracking method which gives several alternatives to provide information about the interface, namely its position to compute the density and viscosity in the domain and the curvatures and normals for the surface tension.

The Front-Tracking method is derived from the Immersed Boundary Method (IBM), which was developed by Peskin (1977) to simulate elastic boundaries for cardiac applications. Two meshes, an Eulerian mesh and a Lagrangian mesh, are coupled by a "smoothed approximation to the Dirac delta function" δ_h, thus enabling interpolation and spreading of quantities (Peskin, 2002). In our implementation, we use two meshes:

- a structured fixed mesh to solve the Navier-Stokes equations, hereinafter referred to as a "grid", or "Eulerian mesh", with a mesh spacing h or h_1, h_2, h_3 if the spacing is not the same in the x, y, z directions.
- a moving and deformable mesh to track the front, hereinafter referred to as a "mesh", or "Lagrangian mesh", with a reference edge length d.

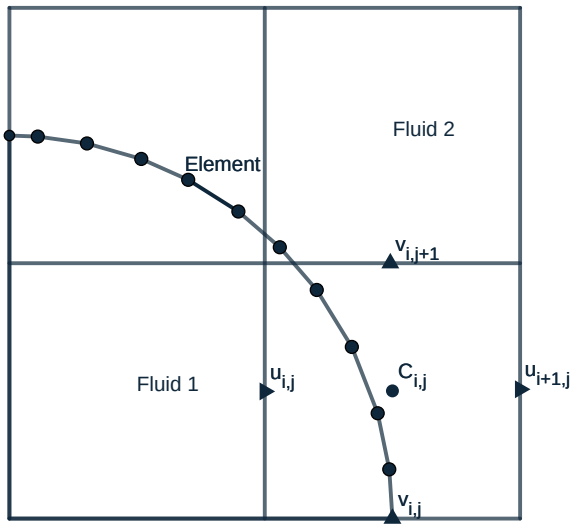


Figure 2.1: Staggered grids and Lagrangian mesh depicted in 2D for simplicity's sake

The location of the interface is known explicitly with the Front-tracking method, contrary to implicit methods such as the VOF or LS methods. The interface is represented by massless markers which are advected in a Lagrangian fashion. The velocity of the markers is not known, so the Eulerian fixed grid is used to compute the velocity field and the velocity is then interpolated on the front.

The main difficulty in 3D is the implementation of a robust algorithm for the merging and breakup of interfaces. The markers may or not be connected to form elements (segments in 2D and triangles in 3D). For instance the Level Contour Reconstruction Method (LCRM) uses a triangle mesh reconstructed from an isosurface of a variable I (volume fraction or a distance function), defined by a value I_{ref}, without storing any connectivities (Shin and Juric (2002), depicted in fig. 2.2). For only one iteration, with the volume fraction, one may use $I_{ref} = 0.5$. The topological changes are automatic, not unlike the VOF method, because the topological changes are made implicitly on the grid with the indicator function and the Lagrangian mesh is reconstructed from the isosurface. The surface tension is computed without connectivities, with spreading functions (Peskin, 2002) to distribute the contribution of each triangle of the Lagrangian mesh onto the Eulerian mesh. Another example of Front-Tracking without connectivities can be found in Torres and Brackbill (2000). The difficulty is increased with the use of connected markers.

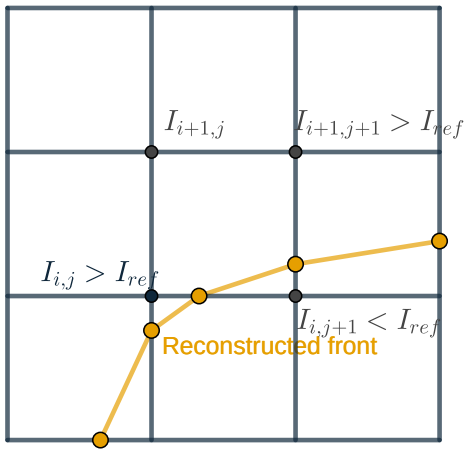


Figure 2.2: Illustration of the Level Contour Reconstruction Method (LCRM), the interface represented in yellow is reconstructed from the variable I

Chosen implementation. In our implementation, connectivities are stored to have access to a larger array of methods to compute the surface tension and volume fraction. For each triangle, the indices of its three neighbors and its three vertices are stored.

2.2 Notations for a triangle mesh

Let us reintroduce the notations from Koffi Bi et al. (2022). The discrete approximation of the interface consists of a family $\mathcal{T}$ of triangles T. Let $\mathcal{X}$ be the set of vertices x of the triangle mesh. The set of vertices of any $T \in \mathcal{T}$ are denoted $\partial^2 T \subset \mathcal{X}$. $\mathcal{T}$ is conforming: each intersection $T_i \cap T_j, i \neq j$, is reduced to either a vertex or an edge of the mesh (Balarac et al., 2022), which means there are no "hanging nodes". Each triangle has three neighbors or less if it is on the boundary in the case of an open mesh. $\mathcal{T}$ is valid: the open triangles T_i are mutually disjoint: $T_i \cap T_j = \varnothing, i \neq j$. From this point on, an intersection-free mesh will refer to such a mesh, where triangle intersections are limited to edges for adjacent triangles or vertices. We refer to the coordinate of any vertex $x \in \mathcal{X}$ by $\mathbf{x}$ and the surface of any triangle $T \in \mathcal{T}$ by S_T. We also introduce:

- the set whose elements are couples of vertices $\mathcal{X}_2 = \left\{(x_\alpha, x_\beta) \in \mathcal{X} \times \mathcal{X} \mid \exists\, T \in \mathcal{T},\ \{x_\alpha, x_\beta\} \subset \partial^2 T\right\}$.
- the first and second neighborhood $\mathcal{N}_1(x)$ and $\mathcal{N}_2(x)$ of any vertex $x \in \mathcal{X}$ (depicted in Figure 2.3) with:

$$\mathcal{N}_1(x) = \left\{x' \in \mathcal{X} \mid (x, x') \in \mathcal{X}_2\right\} \tag{2.1}$$

$$\mathcal{N}_2(x) = \left\{x' \in \mathcal{X} \setminus \mathcal{N}_1(x) \mid \exists\, x'' \in \mathcal{N}_1(x),\ (x', x'') \in \mathcal{X}_2\right\} \cup \mathcal{N}_1(x) \tag{2.2}$$

The valence (or degree) of a vertex is the number of its incident edges. When dealing with closed Lagrangian grids the valence also equals the number of incident triangles. The valence equals 6 in Figure 2.3.

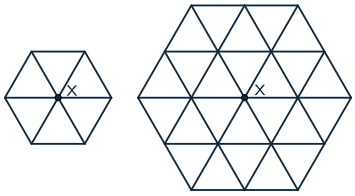


Figure 2.3: First neighborhood $\mathcal{N}_1$ (left), second neighborhood $\mathcal{N}_2$ (right)

The markers are ordered in each triangle in order to compute the curvature, normal and surface tension. For a closed surface, we choose the normal vector to the interface pointing outside denoted $\mathbf{n}$ to define the orientation. We then define a subset $\mathcal{X}_2(T)$ of $\mathcal{X}_2$. The orientation of any triangle $T \in \mathcal{T}$ by $\mathcal{X}_2(T) \subset \mathcal{X}$ is defined in eq. 2.3 with $\times$ the cross product (right-hand rule).

$$\mathcal{X}_2(T) = \{(x_\alpha, x_\beta), (x_\beta, x_\gamma), (x_\gamma, x_\alpha) \mid \partial^2 T = \{x_\alpha, x_\beta, x_\gamma\},\ ((\mathbf{x}_\beta - \mathbf{x}_\alpha) \times (\mathbf{x}_\gamma - \mathbf{x}_\alpha)) \cdot \mathbf{n} > 0\} \tag{2.3}$$

The unit normal vector $\mathbf{n}_T$ to any triangle $T \in \mathcal{T}$ is defined in eq. 2.4 with $\partial^2 T = \{x_\alpha, x_\beta, x_\gamma\}$ and $\mathcal{X}_2(T) = \{(x_\alpha, x_\beta), (x_\beta, x_\gamma), (x_\gamma, x_\alpha)\}$.

$$\mathbf{n}_T = \frac{(\mathbf{x}_\beta - \mathbf{x}_\alpha) \times (\mathbf{x}_\gamma - \mathbf{x}_\alpha)}{\|(\mathbf{x}_\beta - \mathbf{x}_\alpha) \times (\mathbf{x}_\gamma - \mathbf{x}_\alpha)\|} \tag{2.4}$$

eq. 2.5 is an approximation of the unit normal vector to the surface at any vertex $x \in \mathcal{X}$ denoted by $\mathbf{n}_{\mathcal{N}_1}(x)$. This expression is proportional to the linear weighted combination by the triangle surface S_T of the normal vectors $\mathbf{n}_T$ of all the triangles T sharing the vertex $x \in \mathcal{X}$.

$$\mathbf{n}_{\mathcal{N}_1}(x) = \frac{\sum_{T \in \mathcal{T} \mid x \in \partial^2 T} S_T \mathbf{n}_T}{\left\|\sum_{T \in \mathcal{T} \mid x \in \partial^2 T} S_T \mathbf{n}_T\right\|} \tag{2.5}$$

2.3 Algorithm

The Front-Tracking algorithm is described in algorithm 1. We will now present the methods used in this algorithm. The Lagrangian mesh is transported to follow the interface. After this transport, a topological method examines

the new mesh and may perform coalescence or breakup according to predefined criteria. A remeshing operation is then applied to maintain the mesh at a sufficient quality for the two subsequent steps : surface tension and volume fraction computation. After that, the discretized Navier-Stokes equations are solved.

Algorithm 1: Front-Tracking algorithm

1 Transport of the front
2 Topology (coalescence/breakup)
3 Remeshing
4 Computation of the surface tension
5 Computation of the volume fraction
6 Solving the Navier-Stokes equations

2.4 Lagrangian-Eulerian coupling

In the Front-Tracking method, the main variables of interest such as the velocity are solved on the Eulerian grid. The coordinates of the vertices are used to compute the volume fraction for the density and viscosity and sometimes the surface tension as well on the Eulerian grid. Information need to be transferred from the Lagrangian grid to the Eulerian grid and vice versa with distribution functions. The stencil depends on the distribution function. The weights have a finite support to limit the computational cost.

2.4.1 Hypotheses

We take the "smoothed interface approach" and thus suppose that the variables vary smoothly across the interface and that we can use distribution functions for the coupling (Tryggvason et al., 2011). No modification is made in the schemes used in the solver: the Front-tracking method is solely used to compute the material properties or some of the source terms in the conservation equations, mainly the surface tension.

2.4.2 Interpolation

To interpolate a variable Φ on the front at a location l, we identify the grid node closest to l and interpolate on a stencil as in eq. 2.6, where Φ_f^l is the quantity on the front at location l, $\Phi_{i,j,k}$ is its counterpart on the grid point (i,j,k), $w_{i,j,k}^l$ is the weight of each grid node associated to location l. The weights must verify eq. 2.7, since the interpolation of a constant velocity field should give the exact velocity.

$$\Phi_f^l = \sum_{i,j,k} w_{i,j,k}^l \Phi_{i,j,k} \tag{2.6}$$

$$\sum w_{i,j,k}^l = 1 \tag{2.7}$$

2.4.3 Spreading

The "spreading" operation is also referred to as a "smoothing" operation in the literature. The same weighting functions may be used to interpolate quantities from the Eulerian grid to the Lagrangian grid and to spread quantities from the Lagrangian grid to the Eulerian grid. We transfer values from a portion of the Lagrangian grid of surface ΔS to a volume ΔV of Eulerian grid. A necessary condition on the spreading is the conservation of the quantity Φ (eq. 2.8). The interfacial quantity Φ_f is expressed in units per area, and its counterpart on the grid $\Phi_{i,j,k}$ in units per volume.

$$\int_{\Delta S} \Phi_f(s) ds = \int_{\Delta V} \Phi_{i,j,k}(\mathbf{x}) dV \tag{2.8}$$

The grid value $\Phi_{i,j,k}$ is obtained by summing the weighted contributions of the surrounding triangles of area S_T as in eq. 2.9, with h_1, h_2, h_3 the grid spacings.

$$\Phi_{i,j,k} = \sum_l \Phi_f^l w_l^{i,j,k} \frac{S_T}{h_1 h_2 h_3} \tag{2.9}$$

Numerically, for each triangle of the Lagrangian mesh, information are distributed to the Eulerian grid by determining a stencil based on the coordinates of the vertices of the triangle. The weighting function $w_{i,j,k}^l(\mathbf{x})$, also known as δ_h is often written as a product of 1D functions with a scaled distance r as its variable. The distances between the vertex of the Lagrangian mesh and the node of the Eulerian grid in the x, y and z directions are scaled by their respective mesh spacings. For the construction of the Peskin distribution functions such as in eq. 2.11, the reader may refer to Peskin (2002). This "smoothed approximation to the Dirac delta function" is nonsingular but converges towards the Dirac function (Peskin, 2002). For a grid node (i, j, k) and a marker x of coordinates $\mathbf{x} = (x_1, x_2, x_3)$:

$$w_{i,j,k}^l(\mathbf{x}) = \delta((x_1 - ih_1)/h_1)\delta((x_2 - jh_2)/h_2)\delta((x_3 - kh_3)/h_3) \tag{2.10}$$

$$\delta(r) = \begin{cases} \delta_1(r) & |r| \leq 1 \\ 1/2 - \delta_1(2 - |r|) & 1 < |r| < 2 \\ 0 & |r| \geq 2 \end{cases} \tag{2.11}$$

$$\delta_1(r) = \frac{3 - 2|r| + \sqrt{1 + 4|r| - 4r^2}}{8} \tag{2.12}$$

2.5 Transport of the front

When transferring information between the Eulerian grid and the Lagrangian grid, there is a loss of precision, so a special attention is required for the numerical methods. At the same time, we need to pay special attention to robustness, where Peskin's distribution functions may be of help. The coordinates of each marker of the front $\mathbf{x}$ is updated at each time step based on a velocity $\mathbf{v}$ reconstructed from the Eulerian grid. The front is advected by solving eq. 2.13, supposing that there is no phase change.

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \tag{2.13}$$

2.5.1 Interpolation of the velocity of the markers

The front is advected by an interpolated velocity of the fluid, which is not necessarily divergence-free even though the velocity field on the fixed grid may be divergence-free (Tryggvason et al., 2001). Peskin's smoothing functions such as eq. 2.11 can be used to interpolate the velocity. The Parabolic Edge Reconstruction Method (PERM) interpolation presented in McDermott and Pope (2008) is more compact than the Peskin interpolation and uses the divergence at the cell vertices to interpolate the velocity. A comparison of interpolation methods used with the Front-Tracking method can be found in Tavares (2019) and Gorges et al. (2022). Some works like du Cluzeau et al. (2019) use only the component of the velocity normal to the interface.

2.5.2 Temporal scheme

A temporal scheme is required to solve eq. 2.13. The influence of the order of the Runge-Kutta scheme on the order of the precision was evaluated for orders in the range 1 to 3 in Tavares (2019). We describe four Runge-Kutta schemes below. A comparison of these schemes is presented in We present in appendix A. For example, with a first-order Runge-Kutta scheme, the new location of the marker $x^{(n+1)}$ is given in eq. 2.14.

First-order Runge-Kutta.

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \mathbf{v}\Delta t \tag{2.14}$$

With Δt the variable timestep at the current iteration.

Second-order Runge-Kutta.

$$\mathbf{x}^{\star} = \mathbf{x}^{(n)} + \mathbf{v}\Delta t \tag{2.15}$$

$$\mathbf{x}^{(n+1)} = \frac{1}{2}\left(\mathbf{x}^{(n)} + \mathbf{x}^{\star}\right) + \mathbf{v}^{(\star,n+1)}\frac{\Delta t}{2} \tag{2.16}$$

However, for a 2nd Runge-Kutta scheme, an extrapolation of the velocity of the marker $\mathbf{v}^{(\star,n+1)}$ is required for the intermediate step denoted $(*)$. The following extrapolation scheme is used for this extrapolation in order to maintain the 2nd accuracy (written here for adaptive timestep):

$$\mathbf{v}^{(\star)} = (1 + \frac{\Delta t}{\Delta t^{(n)}})\mathbf{v}^{(n)}(\mathbf{x}^{\star}) - \frac{\Delta t}{\Delta t^{(n)}}\mathbf{v}^{(n-1)}(\mathbf{x}^{\star}) \tag{2.17}$$

With Δt the variable timestep at the current iteration and $\Delta t^{(n)}$ the previous timestep.

Third-order Runge-Kutta. The Third-order Strong Stability-Preserving Runge-Kutta method (SSP-RK3) is one of the 3rd Runge-Kutta schemes (Gottlieb et al., 2001). It is used in this work with extrapolations of the velocity field, as in Nangia et al. (2019):

$$\begin{aligned}
\mathbf{x}^{\star} &= \mathbf{x}^{(n)} + \Delta t\mathbf{v}^{(n)} && \text{(2.18a)}\\
\mathbf{x}^{\star\star} &= \frac{3}{4}\mathbf{x}^{(n)} + \frac{1}{4}\left[\mathbf{x}^{\star} + \Delta t\mathbf{v}^{(\star,n+1)}\right] && \text{(2.18b)}\\
\bar{x}_m^{(n+1)} &= \frac{1}{3}\mathbf{x}^{(n)} + \frac{2}{3}\left[\mathbf{x}^{\star\star} + \Delta t\mathbf{v}^{(\star\star,n+\frac{1}{2})}\right] && \text{(2.18c)}
\end{aligned}$$

Like in Nangia et al. (2019), we use extrapolated velocities eq. 2.17 and the two-step Adams-Bashforth scheme:

$$\mathbf{v}^{(\star\star)} = (2 + \frac{\Delta t}{\Delta t^{(n)}})\frac{1}{2}\mathbf{v}^{(n)}(\mathbf{x}^{\star}) - \frac{\Delta t}{\Delta t^{(n)}}\frac{1}{2}\mathbf{v}^{(n-1)}(\mathbf{x}^{\star}) \tag{2.19}$$

Fourth-order Runge-Kutta. A classic 4th Runge-Kutta scheme is given in subsection 2.5.2. Similarly, we can use eqs. 2.17 and 2.19 to improve the order when using spatial interpolation.

$$\begin{aligned}
\mathbf{x}^{\star} &= \mathbf{x}^{(n)} + \frac{\Delta t}{2}\mathbf{v}^{(n)} && \text{(2.20a)}\\
\mathbf{x}^{\star\star} &= \mathbf{x}^{(n)} + \frac{\Delta t}{2}\mathbf{v}^{(\star,n+\frac{1}{2})} && \text{(2.20b)}\\
\mathbf{x}^{\star\star\star} &= \mathbf{x}^{(n)} + \Delta t\mathbf{v}^{(\star\star,n+\frac{1}{2})} && \text{(2.20c)}\\
\bar{x}_m^{(n+1)} &= \mathbf{x}^{(n)} + \frac{\Delta t}{6}\left(\mathbf{v}^{(n)} + 2\mathbf{v}^{(\star,n+\frac{1}{2})} + 2\mathbf{v}^{(\star\star,n+\frac{1}{2})} + \mathbf{v}^{(\star\star\star,n+1)}\right) && \text{(2.20d)}
\end{aligned}$$

2.6 Remeshing

The transport of the interface modifies the distribution of the vertices, and thus its quality. We need to control the quality of the mesh to represent the interface and compute the curvatures precisely. Remeshing operations are more complicated in three dimensions. The additions and removals of triangles may be based on the triangle edges, they are then called edge splitting and collapsing. A few general rules of remeshing for the Front-Tracking method are given in Tryggvason et al. (2001). In 3D, a better quality can be achieved by using smoothing and swapping procedures on the surface. While the Front-Tracking method with non-connected markers requires to reconstruct

the entire interface to maintain the resolution, the use of connectivities gives us control on the mesh on a local level. Our remeshing procedure is detailed in chapter 4.

2.7 Computing the volume fraction

Solving the Navier-Stokes equations for two-phase flows requires the knowledge of the physical properties in both phases, such as the density and the viscosity. The volume fraction is necessary to determine these properties. The volume fraction and/or the phase indicator function may also be used to compute the surface tension. For that matter, a method is required to transfer information from the Lagrangian grid to the Eulerian grid. One method to compute a phase indicator function χ uses a Poisson equation:

$$\nabla^2 \chi = \nabla \cdot \int \Gamma(t)\delta()\mathbf{n}dS \tag{2.21}$$

Discretized as:

$$\nabla^2 \chi = \nabla \mathbf{G} \tag{2.22}$$

With :

$$\mathbf{G}_{i,j,k} = \sum_{T \in \mathcal{T}} w_T^{i,j,k} S_T \mathbf{n}_T \tag{2.23}$$

The normal times the surface of each triangle is spread on the Eulerian grid with a distribution function D (such as in eq. 2.10). The interface has a finite thickness because of the distribution functions (Unverdi and Tryggvason, 1992). The numerical method used in Tryggvason et al. (2001) introduces numerical errors such as overshoots and requires additional steps such as filtering. Ceniceros (2010) uses the Closest Point Transform (Mauch, 2000) to compute a signed distance function. Geometrical methods can be used to compute the volume fraction in each cell (Dijkhuizen et al., 2010) or approximate it, such as the Ray-Casting method. The latter consists in computing the number of intersection between a ray taken in a given direction and the interface to determine which cell is in which fluid. We implemented the Ray-Casting method and it is presented in appendix B.

2.8 Breakup and coalescence

Contrary to the methods based on the transport of a scalar, such as the VOF or the Level-Set method, topological changes are not treated automatically in the basic Front-Tracking method. In the LCRM, the topological changes are automatic, because the topological changes are made implicitly on the grid with the indicator function and the Lagrangian mesh is reconstructed from the isosurface (Shin et al., 2013). We present a geometrical method for coalescence and breakup in chapter 7.

2.9 Computation of the curvature and the surface tension

There are several ways to compute curvatures and normals on a triangulated surface, such as the Laplace-Beltrami Operator (LBO) (Meyer et al., 2003) or local polynomial fittings (Jiao and Zha, 2008). We use the latter in this work. The surface tension may be computed with the Integral Formulation (IF) (Popinet, 2018), with contributions at the limits of the control volume, or with a volumetric formulation.

2.9.1 Integral formulation

In 2D, the surface tension is a force tangential to the interface : $\sigma\mathbf{t}$, with $\mathbf{t}$ the unit tangent to the interface and σ the surface tension coefficient (N m^{-1}). For a control volume Ω, intersected by the interface in two points A and B, the surface tension is:

$$\int_\Omega f_\sigma = \oint_A^B \sigma d\mathbf{t} \tag{2.24}$$

where σ_A and σ_B are the surface tension coefficients at A and B.

The second integral is along the interface. In two dimensions, the surface tension is simply the sum of the tensions at the entry and exit points of the interface in Ω. An advantage is that the contributions on adjacent control volumes cancel out. Popinet and Zaleski (1999) proposed a new numerical representation of surface tension and pressure jumps in 2D. They compute intersections I_{int} between the interface and control volume Ω and compute a component of the surface tension contribution $\pm\sigma\mathbf{t}_{I_{int}}$ for the adjacent control volumes along intersected faces. The computation is done once and added to adjacent cells. The intersection is also used to modify the pressure gradient discretization.

In 3D, we obtain a line integral:

$$\int_\Omega f_\sigma = \oint_C \sigma\mathbf{t} \times \mathbf{n} dl \tag{2.25}$$

The most encountered model in the literature with the Front-Tracking method employs the Integral Formulation with Peskin's functions (eq. 2.11): the surface tension is computed on the Lagrangian mesh and then distributed onto the fixed Eulerian grid (Shin et al., 2005). Deen et al. (2004) uses a mass-weighted distribution. The surface tension force can be computed independently for each element.

2.9.2 Volumetric formulation

By using Frenet's formula, the volumetric formulation is obtained (Popinet, 2018):

$$\int_\Omega f_\sigma = \oint_A^B \sigma d\mathbf{t} = \oint_A^B \sigma\kappa\mathbf{n} ds = \int_\Omega \sigma\kappa\mathbf{n}\delta_S \tag{2.26}$$

With s the curvilinear cordinate, δ_S a surface Dirac δ-function which is non-null only on the interface. Here κ equals twice the mean curvature. In the Continuum Surface Force (CSF) formulation, the volume fraction C is used to define this Dirac (Brackbill et al., 1992):

$$\sigma\kappa\mathbf{n}\delta_S = \sigma\kappa\nabla C \tag{2.27}$$

Shin et al. (2005) developed a hybrid formulation to compute a curvature with Peskin's functions to be used in a CSF formulation. Shin et al. (2018) extended this hybrid formulation to variable surface tension. Popinet (2009) presented a height-function method to evaluate the curvature for a CSF approach. In this work, we use either the CSF approach with height functions or the Integral Formulation with Peskin's functions.

2.10 Issues with volume conservation

The Lagrangian mesh is a discrete approximation of the interface, so the discrete volume is not exact. We cannot have the exact volume and the exact positions of the vertices at the same time. One of the drawbacks of the Front-Tracking method is that it does not conserve the volume unlike the VOF method, and the errors cumulate if no counter-measure is taken. The errors may come from the transport (spatial interpolation and temporal integration) and remeshing operations (edge splitting, collapsing, swapping, smoothing).

Volume correction. The coordinates of the vertices can be modified to retrieve the initial discrete volume. This volume correction may be activated every few iterations (Bunner and Tryggvason, 2002; Pivello, 2012). As pointed out by Roghair et al. (2016), this may cause problems when the reference point for the volume correction lies outside the interface, with skirted bubbles for instance. They also mention the risk of smoothing out physical undulations. Generally, the volume error is computed, and if it exceeds a threshold, the coordinates of the markers are corrected. In Takeuchi and Tryggvason (2020), a volume correction method along the velocity vector is presented. The total amount of corrections is minimized in terms of l_2 norm with the volume conservation constraint. Roghair et al. (2016) uses the smoothing method of Kuprat et al. (2001) not only to smooth parts of

the mesh but to correct the volume locally (after an elementary remeshing operation such as element addition or removal or edge swapping) or globally, on the whole mesh after the transport.

Now, we present two methods to correct the volume for incompressible flows from Tavares (2019) and Koffi Bi (2021) which we use in this work. It should be possible to add a source term in the case of phase change.

Velocity-Based Correction. The Velocity-Based Correction (VBC) shifts the vertices by a displacement proportional to the magnitude of their normal velocity along their normals so that the discrete volume equals the initial discrete volume. Let us define card $\mathcal{T}$. The discrete volume delimited by a closed Lagrangian mesh $\mathcal{V}$ may be decomposed in card $\mathcal{T}$ volumes of tetrahedra $\mathcal{V}_T$ made up of the triangles $T \in \mathcal{T}$ and a reference point G, preferably in the surroundings of the interface for numerical reasons. We use the barycenter as a reference point (eq. 2.28).

$$\mathbf{x}_G = \frac{1}{\operatorname{card}\mathcal{T}} \sum_{x \in \mathcal{X}} \mathbf{x}_x \tag{2.28}$$

The vertices of a triangle $T \in \mathcal{T}$ are denoted $x_{T,i}, i = 1, 2, 3$. We note (n) and $(*)$ respectively the state of the interface before and after the volume correction. The volume of the tetrahedron constituted by triangle T and the reference point G is given by eq. 2.29.

$$V = \Big((\mathbf{x}_{T,1}^{(n)} - \mathbf{x}_G) \wedge (\mathbf{x}_{T,2}^{(n)} - \mathbf{x}_G)\Big) \cdot \Big(\mathbf{x}_{T,3}^{(n)} - \mathbf{x}_G\Big)/6 \tag{2.29}$$

For a triangle T, the vertex $x_{T,i}$ of velocity $\mathbf{v}_{T,i}$ and unit normal $\mathbf{n}_{T,i}$, is displaced by $\mathbf{d}_{T,i}$ (Equation).

$$\mathbf{d}_{T,i} = |\mathbf{v}_{T,i} \cdot \mathbf{n}_{T,i}| \mathbf{n}_{T,i} \tag{2.30}$$

The new position of marker $x_{T,i}$ is:

$$\mathbf{x}_{T,i}^{(*)} = \mathbf{x}_{T,i}^{(n)} + \alpha \times \mathbf{d}_{T,i}^{(n)}, \text{ with } \alpha \in \mathbb{R} \tag{2.31}$$

We determine α by expressing the new volume $\mathcal{V}$ with the shifted vertices. For a triangle T, the volume of the tetrahedron after correction $\mathcal{V}_T^*$ is obtained by adding a correction depending on three coefficients A_T, B_T, C_T and the unknown α (Equations eq. 2.32,2.33).

$$\mathcal{V}_T^{(*)} = \mathcal{V}_T^{(n)} + \big(A_T\alpha^3 + B_T\alpha^2 + C_T\alpha\big)/6 \tag{2.32}$$

$$\begin{aligned}
A_T =& \Big(\mathbf{d}_{T,1}^{(n)} \wedge \mathbf{d}_{T,2}^{(n)}\Big) \cdot \mathbf{d}_{T,3}^{(n)} \\
B_T =& \Big(\mathbf{d}_{T,2}^{(n)} \wedge \mathbf{d}_{T,3}^{(n)}\Big) \cdot \Big(\mathbf{x}_{T,1}^{(n)} - \mathbf{x}_G\Big) + \Big(\mathbf{d}_{T,3}^{(n)} \wedge \mathbf{d}_{T,1}^{(n)}\Big) \cdot \Big(\mathbf{x}_{T,2}^{(n)} - \mathbf{x}_G\Big) + \\
& \Big(\mathbf{d}_{T,1}^{(n)} \wedge \mathbf{d}_{T,2}^{(n)}\Big) \cdot \Big(\mathbf{x}_{T,3}^{(n)} - \mathbf{x}_G\Big) \\
C_T =& \Big((\mathbf{x}_{T,2}^{(n)} - \mathbf{x}_G) \wedge (\mathbf{x}_{T,3}^{(n)} - \mathbf{x}_G)\Big) \cdot \mathbf{d}_{T,1}^{(n)} + \Big((\mathbf{x}_{T,3}^{(n)} - \mathbf{x}_G) \wedge (\mathbf{x}_{T,1}^{(n)} - \mathbf{x}_G)\Big) \cdot \mathbf{d}_{T,2}^{(n)} + \\
& \Big((\mathbf{x}_{T,1}^{(n)} - \mathbf{x}_G) \wedge (\mathbf{x}_{T,2}^{(n)} - \mathbf{x}_G)\Big) \cdot \mathbf{d}_{T,3}^{(n)}
\end{aligned} \tag{2.33}$$

The new volume $\mathcal{V}^{(*)}$ delimited by the triangle mesh is:

$$\mathcal{V}^{(*)} = \mathcal{V}^{(n)} + \big(A\alpha^3 + B\alpha^2 + C\alpha\big)/6 \tag{2.34}$$

With

$$A = \sum_{T \in \mathcal{T}^{(n)}} A_T\,, \quad B = \sum_{T \in \mathcal{T}^{(n)}} B_T\,, \quad C = \sum_{T \in \mathcal{T}^{(n)}} C_T \tag{2.35}$$

The conservation of the volume is expressed by eq. 2.36:

$$\mathcal{V}^{(*)} = \mathcal{V}^{(0)} \iff A\alpha^3 + B\alpha^2 + C\alpha + 6\mathcal{V}^{(n)} - 6\mathcal{V}^{(0)} = 0 \tag{2.36}$$

The unknown α is thus solution to a cubic equation, which we solve directly with Cardano's formula. Among the three solutions, the real solution with the smallest absolute value is retained.

Homothetic Rescaling. The Homothetic Rescaling (HR) shifts the vertices by a homothety to retrieve the initial discrete volume. It is more suited to spherical interfaces. For a given center of homothety x_G (such as the barycenter), the homothety factor is computed so that the new discrete volume $\mathcal{V}^*$ equals the initial discrete volume $\mathcal{V}^{(0)}$ (eq. 2.37). Each marker x is then moved according to eq. 2.38.

$$k_{x_G} = \sqrt[3]{\frac{\mathcal{V}^{(0)}}{\mathcal{V}^{(n)}}} \tag{2.37}$$

$$\mathbf{x}^* = \mathbf{x}_G + k_{x_G}(\mathbf{x} - \mathbf{x}_G) \tag{2.38}$$

Volume deterioration during the transport. Concerning the spatial interpolation, the discrete velocity field may be divergence-free for incompressible flows but this is not necessarily the case for the interpolated velocity (Tryggvason et al., 2011). Peskin and Printz (1993) constructed a divergence operator based on the interpolation scheme, which modifies the projection step in the velocity-pressure coupling, in order to limit the volume alteration.

Volume deterioration during the remeshing. Concerning the remeshing operations, they can be designed to limit the volume loss or even conserve exactly the discrete volume. Several authors designed a volume conserving smoothing (Kuprat et al., 2001; Liu et al., 2002; de Sousa et al., 2004; Toutant et al., 2012). Lindstrom and Turk (1998) designed the Memoryless Simplification algorithm (MSA) which conserves the discrete volume locally during edge collapsing.

By using surface reconstruction with remeshing operations (splitting, collapsing, smoothing) or a fine Lagrangian mesh relatively to the Eulerian one, this issue may be alleviated (Tryggvason et al., 2001). Gorges et al. (2022) perform no volume corrections yet rely on higher order interpolation and temporal schemes and more precise remeshing techniques with local surface reconstruction. We investigate the use of local surface reconstruction to mitigate the alteration of the discrete volume in the next chapter as well.

2.11 Summary

We use the Ray-Casting method to compute the volume fraction and the phase indicator. A volume correction step is employed at the end of the remeshing algorithm. We use either height functions or the integral formulation of the surface tension distributed with Peskin's functions.

Part II

Unsteady analytical test cases

3 Remeshing

Now, we will go over remeshing in the literature, after that we will detail our remeshing procedure, and the remeshing criteria behind it.

The Lagrangian mesh is used to compute the phase indicator, the volume fraction, and in some cases the curvature and normal for the surface tension. The quality of the computation of the curvature depends on the quality of the mesh. The objective of remeshing is to maintain the mesh at a sufficient quality throughout the simulation while mitigating the cumulation of subsequent position errors. Remeshing criteria for meshes are numerous in the literature. In this work, we are concerned with the size of the edges with regards to the Eulerian mesh spacing h, the adaptation of the mesh to the curvature, the valence (section 2.2,13) and the shape of the triangles. These criteria are not always compatible, sometimes a compromise has to be reached. With an Eulerian mesh, we define an edge length interval $[\lambda_{min}, \lambda_{max}]$ based on the mesh spacing h, since the velocity of the markers is interpolated from the Eulerian mesh and triangles larger than the Eulerian mesh would not represent the interface correctly if it has a non-null curvature. Because of the transport the triangles may become larger or smaller than the upper and lower bounds of our edge length interval. When the triangles become too large, we may split them to maintain a sufficient resolution. Yet when the triangles become too small compared to our predefined resolution, we may want to delete them to save up memory and computation time. We perform local operations at each iteration instead of global operations on the whole mesh, like global mesh optimizations. We indeed suppose that the displacement of the vertices is small at each time step and that performing remeshing operations locally (at some vertices only) is sufficient.

3.1 A quick overview of the literature

Estimating curvature on a mesh. The methods used to compute curvature can be distinguished in two main classes: surface fitting and discrete methods. Surface fitting methods search for a function (and thus its coefficients) that fits the mesh locally. This function can be obtained by interpolating the coordinates of the vertices of a stencil (the number of vertices is given by the number of unknown coefficients), or by an approximation, the function then minimizes a measure of distance from the vertices (with a weighted least-squares formulation, the number of vertices should be superior or equal to the number of unknown coefficients). The second class, discrete methods, employ approximations based on the definition of curvature, such as Laplace-Beltrami Operator (LBO) or the IF. The reader may refer to Petitjean (2002) and Gatzke and Grimm (2006) for comprehensive reviews. According to (Koffi Bi, 2021), the LBO method does not always converge with the mesh size depending on the vertex distribution.

Remeshing. Frey (2000) developed an anisotropic remeshing method with metrics, based on the principal curvatures. Principal curvatures can be used to represent a shape more efficiently with regards to the number of vertices (Alliez et al., 2002, 2003). In this work, we are interested in adapting the mesh to the curvature, which is one reason why we need a method to compute the curvatures on a triangle mesh.

Preserving features. Remeshing methods can be designed to preserve the features of a shape (Li et al., 2021). For the Front-Tracking method, when the contact angle is modeled, special care has to be taken to respect the contact angle when remeshing near a wall (Shang et al., 2018). Eigenvalue analysis can be used to handle features of the mesh: smooth patches, ridges and corners (Jiao, 2007). This can be used to smooth the mesh while preserving the features, or prevent the vertices from moving off a ridge for instance when collapsing an edge (Brochu, 2012).

3.2 Remeshing algorithm

The remeshing algorithm is described in Algorithm 2. Triangle edges that are too small are deleted, after that a swapping procedure is applied. Triangle edges are split if they are too large with regards to our remeshing criteria, a new swapping step is then performed. The mesh is smoothed and badly shaped triangles are deleted. The mesh data is saved in allocatable arrays, and the addition and removal of triangles and vertices require a step to resize the arrays for the memory imprint. A volume correction step is applied at the end of the remeshing procedure.

Algorithm 2: Remeshing algorithm

1 Triangle deletion
2 Swapping
3 Triangle addition
4 Swapping
5 Smoothing
6 Bad triangle removal
7 Compacting data structure
8 Volume correction

3.3 Surface Reconstruction

The geometric properties of the interface can be calculated with a global or local reconstruction of the discretized surface. In this work, for a vertex x_P the interface is approximated locally and the curvature is computed on this approximate surface and serves as an approximation for the curvature at the vertex x_p. Jiao and Zha (2008) proposed a local polynomial fitting method with weighted least-squares. The locality of this surface reconstruction method is an advantage for the parallelization. Local surface reconstruction methods give good results for the curvature compared to classical methods, namely LBO, and are more robust for irregular vertex distribution (Koffi Bi, 2021). Li et al. (2021) uses normals and tangents as well in their least-squares formulation.

Now, we describe the procedure used to compute a local surface reconstruction at a vertex x_P. First, a stencil (for instance $\mathcal{N}_1(x_P)$ or $\mathcal{N}_2(x_P)$ section 2.2, page 13) is selected around the vertex x_P, the coordinates of these vertices will be used in the weighted least-squares. These coordinates are expressed in a local orthonormal basis originating at x_P where $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ are its axes. A first approximate normal $\mathbf{n}_{x_P}^{(0)}$ to the surface at $\mathbf{x}_P$ is computed and serves to define the local basis. The $\mathbf{Z}$ axis is aligned with the approximate normal, $\mathbf{X}$ and $\mathbf{Y}$ are chosen arbitrarily in the plane passing x_P and are normal to $\mathbf{n}_{x_P}^{(0)}$.

First approximation of the normal $\mathbf{n}_{x_P}^{(0)}$. In Wang et al. (2013), the normal is approximated with an area-weighted average of face normals. In Max (1999), larger weights are assigned to smaller triangles (eq. 3.1). Zinchenko et al. (1997) performs several iterations of the whole reconstruction to get a better correspondence between the normal used in the local basis and the computed normal.

$$\mathbf{n}_m(x_i) = \frac{\displaystyle\sum_{T\in\mathcal{T}_i^\star} \frac{(\mathbf{x}_j-\mathbf{x}_i)\times(\mathbf{x}_k-\mathbf{x}_i)}{\|\mathbf{x}_j-\mathbf{x}_i\|^2\|\mathbf{x}_k-\mathbf{x}_i\|^2}}{\left\|\displaystyle\sum_{T\in\mathcal{T}_i^\star} \frac{(\mathbf{x}_j-\mathbf{x}_i)\times(\mathbf{x}_k-\mathbf{x}_i)}{\|\mathbf{x}_j-\mathbf{x}_i\|^2\|\mathbf{x}_k-\mathbf{x}_i\|^2}\right\|} \tag{3.1}$$

with $\mathcal{T}_i^\star = \left\{T \in \mathcal{T} | \partial^2 T = \{x_i, x_j, x_k\}, (x_j, x_k) \in \mathcal{X}_2(T)\right\}$

Weighting in the least-squares. In Jiao and Zha (2008), larger weights are assigned to vertices close to the origin x_P, and if their normals are similar to the approximate normal of the stencil $\mathbf{n}_{x_P}^{(0)}$. Including the normal in the weight is useful to preserve sharp features or if the mesh is coarse. Li et al. (2021) use the tangents instead of

the normals in their new weighting and Wendland functions instead of the "inverse-distance-weighting" scheme (Jiao and Zha, 2008) noting that if the origin is not a vertex, an asymmetry might be introduced.

System resolution. The resolution of the system involves issues about the rank of the matrix which may be alleviated by using larger stencils which in turn may smooth out the reconstructed surface (Cazals and Pouget, 2005). For instance, some issues can arise with the $\mathcal{N}_1$ stencil, if the number of vertices is not superior or equal to the number of unknown coefficients in the polynomial, the system is underdetermined and the stencil must be completed (Du et al., 2006). Koffi Bi (2021) shows that when supplementary vertices are used, this may produce an alignment of three vertices, which deteriorates the approximation of the surface. Switching to the $\mathcal{N}_2$ stencil becomes thus necessary. Jiao and Zha (2008) scale their matrix to improve the resolution of the least-squares and solve the system with a QR method, which gives access to a condition number. This number gives an indication of the quality of the reconstruction, enabling them to switch to lower order fittings should the need arise.

In our implementation. We use the local surface reconstruction method implemented in Koffi Bi (2021) at the vertices only. The steps to compute the geometric properties with local surface reconstruction are summarized in algorithm 3. The local basis is constructed with the normal introduced by Max (1999). The surface is approximated locally by the equation $Z = f(X, Y)$ with f the height function given in eq. 3.2. The local coordinates are denoted by X, Y and Z, and a_{ij} are unknown real coefficients which we determine by a least-squares formulation. It is possible to impose $a_{00} = 0$, so that the vertex x_P belongs to the reconstructed surface, but keeping a_{00} as an unknown gave better results for $\mathcal{N}_2$ (Koffi Bi, 2021).

$$f(x, y) = a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{10}x + a_{01}y + a_{00} \tag{3.2}$$

The local equation is obtained by minimizing:

$$\sum_{m_i \in \mathcal{N}_k^\star(x_P)} \omega_i [f(X_i, Y_i) - Z_i]^2 \tag{3.3}$$

In this weighted least-squares formulation, ω_i is the weight associated with m_i, of local coordinates X_i, Y_i, Z_i, $k \in \{1, 2\}$ indicates the first and second neighborhood. The weight ω_i, is in the form $1/(1 + x^2)$, as in eq. 3.4. An integer parameter N is employed to set the weighting: the vertices of the stencil have then less influence in the resulting coefficients if they are more distant from P. We use $N = 1$, which tends to improve the results for coarse meshes, as recommended in (Koffi Bi, 2021).

$$\omega_i = \frac{1}{1 + \left[\dfrac{\|\mathbf{x}_Q - \mathbf{x}_P\| \times N}{\min\limits_{m_i \in \mathcal{N}_k^\star(m)} \|\mathbf{x}_Q - \mathbf{x}_P\|} \right]^2} \tag{3.4}$$

We use the Gauss method for the resolution of the system. Low valency and aligned vertices are detrimental to the surface reconstruction with $\mathcal{N}_1$ (Koffi Bi, 2021), which is one of the reasons why we elected to use $\mathcal{N}_2$ for our computations. Another reason is that $\mathcal{N}_2$ gives smoother contours of curvature, which is more suited as the curvature is one of our remeshing criteria. The geometric properties are then computed with the polynomial f.

Algorithm 3: Local surface reconstruction algorithm

1 Fetching stencil ($\mathcal{N}_1$ or $\mathcal{N}_2$)
2 Local basis construction
3 Weighted least-square formulation
4 System resolution
5 Computation of geometric properties

We denote the Hessian of the function $F(X,Y,Z) = Z - f(X,Y)$ by $H(F)$. As detailed in Goldman (2005), the mean curvature κ and normal to an implicit surface can be computed with an analytical formula (eqs. 3.5 and 3.7).

$$\mathbf{n} = \frac{\nabla F}{\|\nabla F\|} \tag{3.5}$$

$$\kappa = -\nabla \cdot \mathbf{n} \tag{3.6}$$

$$\kappa = \frac{\nabla F \cdot H(F) \cdot \nabla F^T - \|\nabla F\|^2 \operatorname{Tr}(H)}{2\|\nabla F\|^3} \tag{3.7}$$

$$\kappa_G = \frac{\det(H(F))}{\left(\|\nabla F\|^2 + 1\right)^2} \tag{3.8}$$

The Gaussian curvature is denoted by κ_G (eq. 3.8). We denote by κ_1 and κ_2 the principal curvatures when they exist. The principal curvatures are the maximal and minimal values of the normal curvature (a real number that measures how the surface bends in a direction tangential to the surface, Gray et al. (2017)). We can retrieve κ_1 and κ_2 with the mean and Gaussian curvature: $\kappa_1 = \kappa + \sqrt{\kappa^2 - \kappa_G}$ and $\kappa_2 = \kappa - \sqrt{\kappa^2 - \kappa_G}$. As mentioned in Meyer et al. (2003), with machine error we must pay attention to the sign of the discriminant when the principal curvatures are almost equal at the machine precision with $\kappa_1 = \kappa + \sqrt{max(0, \kappa^2 - \kappa_G)}$ and $\kappa_2 = \kappa - \sqrt{max(0, \kappa^2 - \kappa_G)}$.

3.4 Remeshing criteria in the literature

In this paragraph we give a quick overview of the remeshing criteria used in Front-tracking algorithms. The most common criterion for adding or deleting triangles is the triangle edge length l. The edge length is typically chosen with regards to the Eulerian mesh spacing h as illustrated in fig. 3.1, in an interval $[l_{min}, l_{max}]$ such as $\frac{1}{3}h < l < h$ Tryggvason et al. (2001), $\frac{1}{5}h < l < \frac{1}{2}h$ (Roghair et al., 2016).

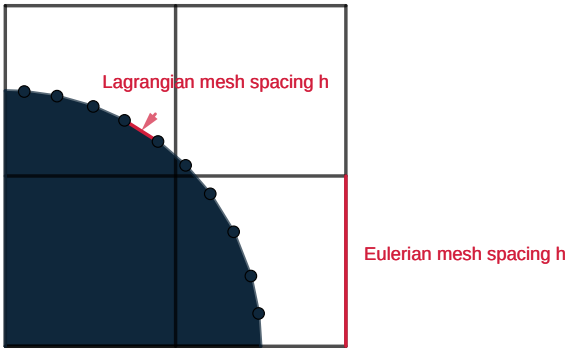


Figure 3.1: Eulerian and Lagrangian meshes

However, with fixed upper and lower limits for the edge length, the mesh will be homogeneous spatially which means there will be the same resolution in the less curved areas and in the more curved areas. This takes up more disk space and CPU time. The number of triangles may be reduced by using adaptive remeshing criteria. Roghair et al. (2016) define a "roughness" (eq. 3.9). This roughness characterizes local undulations, by evaluating the inclination of adjacent triangles with one another, also known as the dihedral angle, on the $\mathcal{N}_1$ neighborhood.

$$r(P) = \frac{1}{2} \min_{(P_1,P_2,P_3)\in\mathcal{C}} (1 + \mathbf{n}(PP_1P_2) \cdot \mathbf{n}(PP_2P_3)) \tag{3.9}$$

With $\mathcal{C} = \{P \neq P_1 \neq P_2 \neq P_3 \mid P_1 \in \mathcal{N}_1(P),\ P_2 \in \mathcal{N}_1(P) \cap \mathcal{N}_1(P_1),\ P_3 \in \mathcal{N}_1(P) \cap \mathcal{N}_1(P_2)\}$ and $\mathbf{n}(PP_1P_2)$, $\mathbf{n}(PP_2P_3)$ are the vectors normal to the two triangles constituted of the corresponding vertices.

If $1 - r > 10^{-2}$ and $l/2 > l_{min}$ (the new edge would still be within the remeshing edge interval), then the edge is split (Roghair et al., 2016). If the mesh is flat: $1 - r \leq 10^{-3}$ and $2l < l_{max}$, the edge is collapsed. In Galaktionov et al. (2000), the remeshing criteria are based on edge lengths and dihedral angles.

3.5 Our remeshing criteria

In our code, we selected the following criteria: non-dimensionalized curvature, edge length, similarity to an equilateral triangle and roughness. Instead of using the roughness to define the edge length in the edge collapsing and splitting, we prefer a smoothed curvature which gives a better transition for the edge lengths on the mesh. The aim of using the non-dimensionalized curvature is to refine automatically the front where the curvature is high compared to the edge length. It would enable us to describe properly the front at a subgrid scale and especially for the curvature computation granted that the velocity used to transport the interface is precise enough. The advantage is that it would not require to specify a reference length as a parameter before running the simulation. However, if the mesh presents large undulations, we may have large variations of curvature which result in an overly refined mesh because of the non-dimensionalized curvature criterion. This problem is mitigated with the $\mathcal{N}_2$ neighborhood, as opposed to the $\mathcal{N}_1$ neighborhood, the mean curvature contour is smoother, which limits this problem of refining at unnecessary places due to oscillations.

3.5.1 Mesh adaptation to curvature

The lagrangian mesh is adapted with the edge length and κ_{max}, the maximum of the absolute value of the principal curvatures of the interface. This value is non-dimensionalized by the edge length. If the numerical curvature is used as is, large variations in edge length are to be expected, as depicted in fig. 3.2, which is problematic for surface reconstruction and transport.

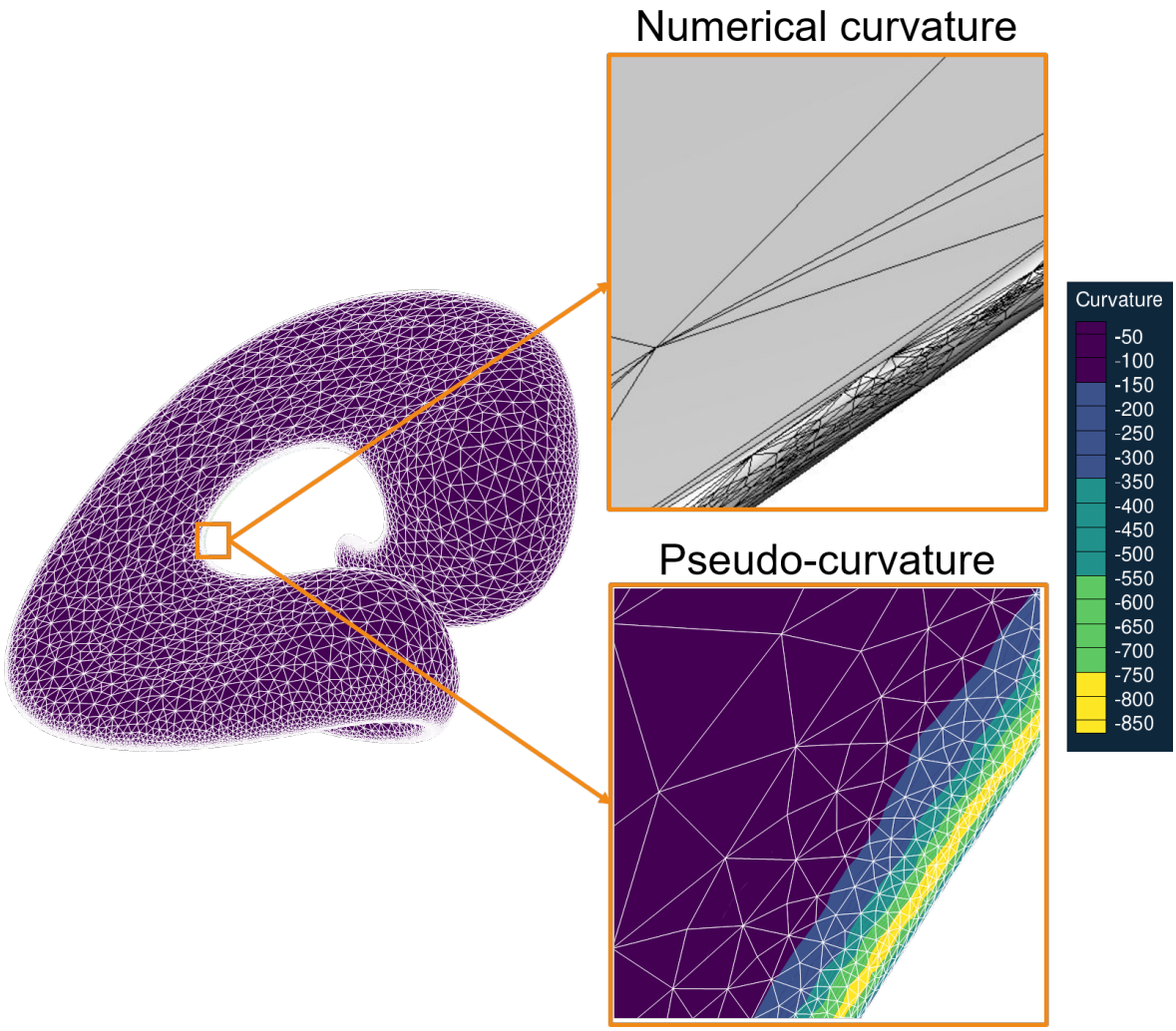


Figure 3.2: Remeshing with curvature, Enright test case (Enright et al., 2002), with numerical curvature (top), with pseudo-curvature: $\lambda = 0.5$ (bottom)

To avoid large variations in edge length, this curvature κ_{max} is replaced by a pseudo-curvature defined at the marker P by

$$\tilde{\kappa}(P) = \max\left(\kappa_{max}(P), \lambda \max_{Q \in \mathcal{N}_1(P)} [\tilde{\kappa}(Q)] + (1-\lambda)\kappa_{max}(P)\right) \tag{3.10}$$

where $\lambda \in [0; 1]$ is an adjustable parameter and $\mathcal{N}_1(P)$ is the set of markers on the first neighborhood $\mathcal{N}_1$ of P (fig. 2.3). If $\lambda = 0$, the pseudo-curvature at the point P is simply $\kappa_{max}(P)$. If $\lambda = 1$, the pseudo-curvature P is the maximum between $\kappa_{max}(P)$ and the pseudo-curvatures of the neighborhood $\mathcal{N}_1(P)$, which tends to generate a fine mesh of constant edge length, adapted to the largest principal curvature of all markers. An intermediate value of λ smooths the large curvatures to neighboring markers and ensures a better transition between edge lengths.

By denoting l the length of the edge linking the markers P and $Q \in \mathcal{N}_1(P)$, it will be split in two if $\tilde{\kappa}(P)l > \kappa d_{max}$ and the markers will be merged if $\tilde{\kappa}(P)l < \kappa d_{min}$, with for instance $\kappa d_{max} = 0.25$ and $\kappa d_{min} = 0.1$. When $\kappa d \neq 0$, its inverse may be interpreted as a number of segments per radius of curvature, where the curvature is a smoothed approximate curvature.

3.5.2 Additional criteria for mesh management

In order to avoid that the edges created by this algorithm become too small (for computational cost and relevance with the Eulerian resolution) or too large (poor resolution), the edge length is limited between λ_{min} and λ_{max}.

These limits are linked to the size of the Eulerian grid h: $\lambda_{min} = h/4$ and $\lambda_{max} = h$ when $\kappa d < \kappa d_{min}$ (when the surface is nearly flat), and otherwise $\lambda_{min} = h/n$, with $n = 20$ for example.

In addition to these criteria based on edge length, the state of the surface and valence must be controlled. Remeshing operations are cancelled if they worsen the roughness $r(P)$ at the vertex P eq. 3.9. If the valence of a vertex involved in a potential edge collapse equals 5, the edge collapse is cancelled, since it is detrimental to the computation of the curvature, at least for a $\mathcal{N}_1$ reconstruction. The $\mathcal{N}_1$ neighborhoods of the two vertices are checked for shared vertices. If they share more than two vertices, the collapsing is cancelled since it would create an invalid mesh (edge shared by more than two triangles). Such a situation may arise when a ligament stretches.

3.6 Computing a new vertex position

When splitting an edge constituted by two vertices x_{P_1} and x_{P_2}, a new vertex and two new triangles are created (the reader may refer to Koffi Bi (2021) for more details about the modifications of the connectivities during the remeshing procedures). When collapsing an edge, a vertex and two triangles are removed. In both cases, a new position is defined for the new vertex (edge splitting) or the existing vertex (edge collapsing). This new position may alter the volume, shape and curvature of the interface. For the edge splitting, the most basic method consists in taking the average of the coordinates of the two vertices constituting the edge being refined, as in table 3.1, which we refer to as edge middle. For the edge collapsing, the edge middle or the coordinates of one of the two vertices may be used. The latter would have the advantage of mitigating the cumulation of the position errors, although the triangles modified by the edge collapsing would not be as well shaped as with the former.

Yet when splitting or collapsing an edge, there are other alternatives, such as interpolation with butterfly subdivision (Brochu and Bridson, 2009), interpolation with barycentric coordinates (system of coordinates which may be applied to a triangle, as depicted in fig. 3.3) (Tryggvason et al., 2011), quadric errors to minimize the distance to the planes formed by neighboring triangles (Garland and Heckbert, 1997), the MSA as mentioned in chapter 2 (Lindstrom and Turk, 1998), and local surface reconstruction.

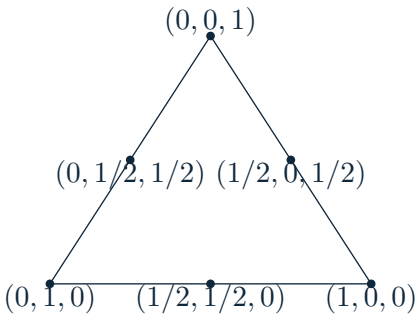


Figure 3.3: Barycentric coordinates

We described a local reconstruction of the surface, centered on a vertex of the mesh in section 3.3. Now, each reconstruction is independent, even though neighboring vertices include shared vertices in their stencils and we may use a weight in our least-squares formulation. Jiao and Wang (2012) introduce two methods based on weighted least squares polynomial fittings: the Weighted Averaging of Local Fittings (WALF) and Continuous Moving Frames (CMF) methods. They use the barycentric coordinates (fig. 3.3) of the vertices to define a new position. According to Jiao and Wang (2012), the WALF method constructs a $\mathcal{C}^0$ continuous surface. They argue that the distance between a point obtained with the WALF method and the closest point on the exact surface is due to the "discrepancy of local coordinate systems at different vertices" but above all linked to the degree of polynomials used in the least-squares fittings. On that matter, we observed that it is more difficult to use a $\mathcal{N}_1$ neighborhood for the fittings with the WALF method (in some cases the surface became non-smooth with time in some of our simulations, i.e. the method gave more errors than a basic edge middle position) than with $\mathcal{N}_2$ which is smoother. The CMF method defines local bases and weights for the local fittings. A recent investigation was done

on an expanding sphere (Gorges et al., 2022) comparing edge refinement with or without surface reconstruction or with barycentric interpolation. The Surface Reconstruction used in Gorges et al. (2022) is reconstructed at the middle of the edge while in the work of Jiao and Wang (2012), an average of two projected points is computed. In our case, we mostly use the WALF method, i.e. we average the projections over an edge. Table 3.1 summarizes the three edge refinement methods used in this work. The notation $\mathcal{P}_i[p]$ means the projection of point p onto the surface reconstructed at the vertex i/edge ij. Figure 3.4 illustrates the two $\mathcal{N}_2$ stencils and their associated surface reconstructions used in WALF method (Jiao and Wang, 2012). The edge middle is projected on each reconstructed surface and the projections are averaged. Figure 3.5 illustrates the stencil used in the edge fit (Gorges et al., 2022). The edge middle is projected on the reconstructed surface.

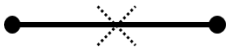
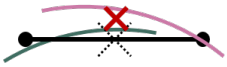
Method	Edge middle	WALF	Edge fit
			
Formula	$(\mathbf{x}_{P_1} + \mathbf{x}_{P_2})/2$	$(\mathcal{P}_1[((\mathbf{x}_{P_1} + \mathbf{x}_{P_2})/2)] + \mathcal{P}_2[((\mathbf{x}_{P_1} + \mathbf{x}_{P_2})/2)])/2$	$\mathcal{P}_{12}[((\mathbf{x}_{P_1} + \mathbf{x}_{P_2})/2)]$

Table 3.1: Edge refinement methods

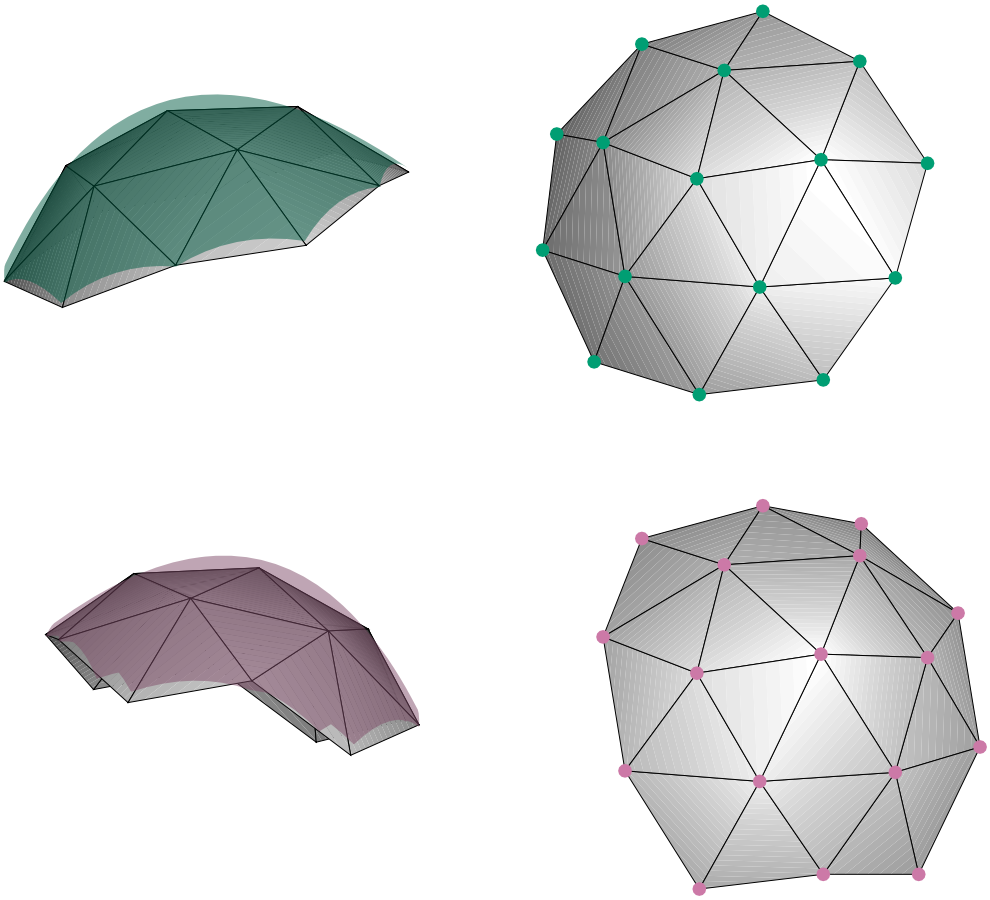
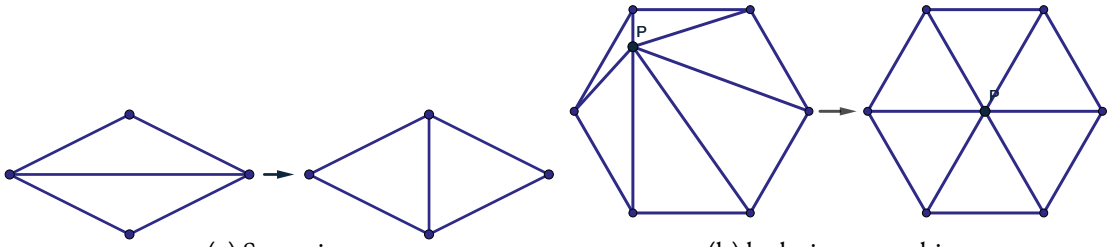


Figure 3.4: The two $\mathcal{N}_2$ stencils and their associated surface reconstructions used in WALF method (Jiao and Wang, 2012); left: stencils with the surface reconstruction, right: top view of the stencils. The edge middle is projected on each reconstructed surface and the projections are averaged.

Figure 3.5: The stencil for the edge fit (Gorges et al., 2022); left: stencil with the reconstructed surface, right: top view of the stencil. The edge middle is projected on the reconstructed surface.

3.7 Swapping

The swapping procedure consists in modifying the connectivity of two adjacent triangles, as depicted in fig. 3.6a. It can have several objectives, such as improving the valence of the mesh. Like other operations, we need criteria to determine if the operation improves or deteriorates the quality of the mesh. The advantage of swapping is that it does not modify the position of the vertices yet it does modify the curvature and volume. Some authors swap edges only if the dihedral angle is smaller than a predefined value (Frey, 2000; Jiao et al., 2010; Koffi Bi, 2021; Gorges et al., 2022). In Brochu and Bridson (2009), swapping is performed if the potential new edge length is smaller than the current one (there is a criterion on the improvement of the edge length to prevent swapping the same edges back and forth), and cancelled if the volume change exceeds a threshold or causes an intersection. The swapping is iterated several times so that there are no longer edges to be swapped. Roghair et al. (2016) swap edges if either the balance between the valences is improved or if the triangles become more equilateral (by computing a ratio (eq. 3.11) for the two current triangles and the potential new triangles). They restore the lost volume in a smoothing step, with the algorithm from Kuprat et al. (2001). In Ray (2013), edges are swapped either to improve valence or so that the incident triangles become more equilateral.



(a) Swapping (b) laplacian smoothing

Figure 3.6: Remeshing operations

Our implementation. Swapping modifies the connectivities so we make sure that swapping two triangles does not reduce the valence of a vertex to less than 5. We perform two swapping steps, one after edge collapsing and one after edge splitting (algorithm 2), as in Koffi Bi (2021). We use a quality criterion for the swapping Q_T, it

is a measure of how similar a triangle is to an equilateral triangle. It is the ratio of the diameter of the inscribed circle D_i and the radius of the circumscribed circle R_c :

$$Q_T = D_i / R_c \tag{3.11}$$

For a triangle $T \in \mathcal{T}$, and its edge lengths $d_{T,i}, i = 1, 2, 3$:

$$Q_T = \frac{(d_{T,1} + d_{T,2} - d_{T,3}) \times (d_{T,2} + d_{T,3} - d_{T,1}) \times (d_{T,3} + d_{T,1} - d_{T,2})}{d_{T,1} \times d_{T,2} \times d_{T,3}} \tag{3.12}$$

For an equilateral triangle $Q_T = 1$ and for a flat one, $Q_T = 0$. One of the objectives of edge swapping is to improve this quality. Let T_1, T_2 be two triangles which we may swap, the average quality after the operation (denoted by $(*)$) should be improved. We use the implementation of (Koffi Bi, 2021), where the harmonic average is used ($k = -1$):

$$\left[\left(Q_{T_1}^{(*)^k} + Q_{T_2}^{(*)^k}\right)/2\right]^{1/k} > \left[\left(Q_{T_1}^{(n)^k} + Q_{T_2}^{(n)^k}\right)/2\right]^{1/k} \tag{3.13}$$

We check the roughness of the vertices whose connections are modified before and after the swapping is applied and if a roughness goes under the roughness threshold, we cancel the swapping operation. That way, the change in roughness induced by the swapping is controlled. However, in our approach, we cannot have both an adaptive mesh and equilateral triangles, because there is a transition in edge size which cannot be achieved with equilateral triangles and a conforming mesh (a triangle has three neighbors at most along its edges).

3.8 Removing badly shaped triangles

If a triangle has an angle $\theta > 160°$, we refer to it as a "cap", and its largest edge is swapped if it does not spawn configurations with low valence (<5) or foldings. If a triangle has an angle smaller than 10° and an edge ratio $\min_{i \in [\![1,3]\!]}(d_{T,i}) / \max_{i \in [\![1,3]\!]}(d_{T,i}) \leq 0.3$, we refer to it as a "needle" and the smallest edge is collapsed if the conditions on valence and non-folding are met as well.

3.9 Smoothing

The smoothing operation can have several purposes: improving the distribution of the points on the surface (have a more homogeneous mesh) or reduce the spikes or undulations which may arise from coupling errors with Navier-Stokes for example.

3.9.1 Laplacian smoothing

Laplacian smoothing shifts every vertex toward a weighted average of its neighboring vertices. Aside from smoothing out some of the geometric features, it has the effect of equalizing edge lengths, which gives a better vertex distribution. On the other hand, it has undesirable effects like shrinking the volume enclosed by the surface and vertex sliding: even if the surface is flat, the vertices are shifted. This problem can be mitigated by applying the smoothing only when necessary, by checking the roughness for instance. There is also a limit on how much we want to smooth sharp features, some may be due to numerical errors, others may be physical.

The mesh can be interpreted as a signal, where the high frequencies would be the noise and the low frequencies the features we wish to retain. As pointed out by Desbrun et al. (1999), the noise of the mesh can be attenuated with a diffusion equation:

$$\frac{\partial \mathbf{x}}{\partial t} = \lambda L(\mathbf{x}) \tag{3.14}$$

With a discrete Laplacian operator for graphs:

$$L(\mathbf{x}_P) = \frac{1}{card \mathcal{N}_1} \sum_{Q \in \mathcal{N}_1(P)} (\mathbf{x}_Q - \mathbf{x}_P) \tag{3.15}$$

Which gives

$$\mathbf{x}_p^* = 1 + \lambda dt \frac{1}{\operatorname{card} \mathcal{N}_1} \sum_{Q \in \mathcal{N}_1(P)} (\mathbf{x}_Q - \mathbf{x}_P) \tag{3.16}$$

According to Desbrun et al. (1999), the stability criterion requires $\lambda dt < 1$, otherwise oscillations appear on the surface. The explicit Laplacian smoothing is thus written for a vector X containing the coordinates of the vertices of the mesh:

$$X_{n+1} = (I + \lambda dt L) X_n \tag{3.17}$$

He then proposes an implicit procedure over the mesh X which removes the stability issues:

$$(I - \lambda dt L) X_{n+1} = X_n \tag{3.18}$$

Taubin (1995) proposes a smoothing method without shrinkage, by applying two steps with a positive and negative factor λ and μ. Different weights can also be assigned to the vertices on the $\mathcal{N}_1$ neighbourhood: as in the "scale-dependent umbrella operator" (Fujiwara, 1995; Desbrun et al., 1999).

3.9.2 Other smoothing methods

Desbrun et al. (1999) uses a mean curvature flow equation (eq. 3.19), to smooth the surface without the vertices sliding. Kuprat et al. (2001) develops methods based on vertices or edges which conserve the discrete volume locally. In Toutant et al. (2012), a curvature-based volume conserving smoothing is presented. Meyer et al. (2003) developed a smoothing with weights based on the principal curvatures to keep features while smoothing out noise. Jiao (2007) developed a procedure called "null-space smoothing", which tends to preserve the features of the interface and limits volume deterioration (compared to a basic Laplacian smoothing) while smoothing the mesh thanks to an eigenvalue analysis. If the vertex is on a smooth part of the mesh, the null space corresponds to the plane tangential to the surface at the vertex. If a vertex is on a ridge, it is moved along the line tangent to the ridge and if a vertex is on a corner, it stays on the corner. In the same line of work, a "variational smoothing" is presented in Jiao et al. (2011), the aim is to get closer to ideal triangles (for instance equilateral) by minimizing two energy functions while keeping the displacement virtually tangential to the surface with the "null-space".

$$\frac{\partial \mathbf{x}}{\partial t} = \lambda \kappa \mathbf{n} \tag{3.19}$$

3.9.3 Related to smoothing

The use of a smoothing algorithm can also be partly avoided by modifying the transport equation, using only the normal component of the velocity for the simulation of bubbly flows (du Cluzeau et al., 2019). However, it supposes that the normal is well computed and it raises the issue of crossflows, the remeshing algorithm would need to handle the tangential movement of the interface somehow.

3.9.4 Our implementation

We use either the edge-based smoothing from Kuprat et al. (2001) or a Laplacian smoothing combined with a projection on a reconstructed surface (section 3.3). The latter needs more caution as it may fold the mesh. Smoothing can be applied implicitly or explicitly. Since the mesh $\mathcal{T}^{(n-1)}$ is supposed to have a sufficient quality, we deem the explicit smoothing sufficient, and it can prove useful in case the code is parallelized, instead of smoothing the whole mesh implicitly. In our implementation, we smooth the mesh explicitly, by projecting the barycenter (eq. 3.20, fig. 3.6b in 2D) on the reconstructed surface $\mathcal{P}_{x_p}\left(x_p^*\right)$, which limits the volume shrinkage. If we have adaptive remeshing, smoothing can add more work to the collapsing and splitting steps though by moving vertices from their desired location. By placing the vertex to the barycenter, it is equivalent to setting $\lambda dt = 1$ in eq. 3.20, where it is unstable for a global explicit smoothing step. The difference here is that we smooth one vertex at a time, which is different from a simultaneous operation on the mesh and that we project the averaged

position onto the reconstruction surface. In that procedure, it is expected that the vertices slide. We may smooth vertices with a given roughness to limit the accumulation of errors or at a given frequency (every n iteration). We check that the roughness does not become smaller than a threshold such as $r = 0.9$. We may also perform an intersection check before each vertex displacement.

$$\mathbf{x}_p^* = \frac{1}{\operatorname{card} \mathcal{N}_1} \sum_{Q \in \mathcal{N}_1(P)} \mathbf{x}_Q \tag{3.20}$$

3.10 Mesh validity

During the remeshing operations or the transport, the following issues can arise: mesh folding, also known as tangling, and mesh intersections. The mesh can be checked before (remeshing) or after (transport) so that no such event occurs.

3.10.1 Intersection handling

Brochu and Bridson (2009) implemented a collision resolution procedure which rolls back some operations or perturbs the mesh to maintain an intersection-free mesh. In Roghair et al. (2016) no particular test is made for the transport of the front. They argue that with the pressure increase in the liquid film and a sufficiently small time step meshes will not intersect in their computations. They perform a volume correction step and in case of intersections, move the intersecting vertices back and redistribute the volume to the non-intersecting vertices. Du et al. (2006) developed a topological method which can be used to repair tangled meshes by computing intersections of the triangles with an Eulerian cell. Some authors add an artificial force to prevent drops from colliding (Bois, 2017).

Our implementation. In our simulations, we may either test intersections after some local remeshing step (when a vertex is moved or edge is swapped for instance), or after global remeshing steps (all triangles refined) or not at all. The detection of intersections is accelerated by keeping a localization of the Lagrangian mesh on the Eulerian mesh, we may thus check only the triangles in the vicinity (by checking the triangles located in the surrounding Eulerian cells) and not the whole triangle mesh. If there is an intersection during the mesh refinement with projection on a reconstructed surface, the new point is placed on the edge instead of the reconstruction. For the transport or volume correction, the code returns an error if the test returns an intersection, for it is likely due to a resolution error with the Navier-Stokes equations or a too large time step.

3.10.2 Folded triangles

In several operations, such as edge collapsing (with or without projection), splitting (with projection), swapping, smoothing (with projection), it is not guaranteed that the mesh will not fold, characterized by two triangles T_1, T_2 where $\mathbf{n}_{T_1} \cdot \mathbf{n}_{T_2} < 0$. In the case of surface reconstruction, folding may be prevented for coarse meshes with the weights from Jiao and Wang (2012): if a normal is too different from the first normal approximation, it is filtered out (section 3.3). Ray (2013) uses weighted Laplacian smoothing constrained in the "null-space" (Jiao, 2007) to resolve mesh tangling.

3.10.3 Limiters

Ray et al. (2014) use surface reconstruction in some of their remeshing operations. Geometrical limiter based on a mean edge length are applied to decide if a point should be projected on the reconstructed surface. If it exceeds the threshold value, they resort to the volume conserving smoothing from Kuprat et al. (2001).

In our implementation. If we use limiters based on a length, sometimes we prevent the appearance of spikes. However, it can also introduce dimples in the surface because at some parts the offset to the surface of the mesh is null and next to it there are points which were projected on a reconstructed surface. These dimples could then trigger the curvature-based remeshing criteria and add more vertices. This behaviour is undesirable, and we

check the improvement of the roughness (or at least that the roughness is maintained above a threshold value like $r = 0.9$) instead before using surface reconstruction or swapping an edge (eq. 3.9).

3.11 Summary

In addition to classical remeshing operations (edge splitting, collapsing, swapping, smoothing), we presented a criterion to adapt the mesh to the curvature of the interface and the use of polynomial approximation to improve edge splitting and collapsing. During the remeshing and the transport, foldings or intersections may occur, so the validity of the mesh needs to be verified.

Now that we have presented our remeshing methods and criteria, we use them in the next chapter in two test cases to evaluate the level of errors that can be attained for curvature and position when a surface deforms and position errors cumulate.

4 Curvature of a surface

In this chapter, the influence of the remeshing procedure on the position and curvature errors is evaluated on two analytical cases. The volume error is investigated for the first case as well.

Remeshing operations serve the purpose of improving the representation of the surface and/or to improve the mesh for the curvature evaluation. However, they also introduce errors. For instance, edge collapsing may alter the volume, shape and curvature. We study two cases of transported surface, where the shape and curvatures are known with analytical functions. We can see the effect of accumulated errors from transport combined with remeshing. In this study, the numerical curvatures are computed on the triangle mesh with surface reconstruction on a $\mathcal{N}_2$ stencil (section 3.3, on page 24).

Presentation of the results. For the sake of visibility, the markers of the figures are slightly shifted left and right to avoid overlappings when studying the influence of a parameter, like in fig. 4.4 (on page 43). When a circular marker is empty, the remeshing was done without adaptation to the curvature with the non-dimensionalized curvature κd, otherwise κd is used and its target value κd_{max} is the value indicated in the legend, and we choose $\kappa d_{min} = \kappa d_{max}/4$ (section 3.5). For the abscissa, N is the total number of vertices at the end of the simulation, κ is the maximum mean curvature in absolute value, h is the reference edge length at the start of the computation. The minor ticks in the log-log figures are of the form $i \times 10^j, i \in [2, 3, 4, 5, 6, 7, 8, 9]$.

4.1 Expanding and shrinking sphere

The aim of this test is to evaluate the errors due to mesh refinement for a simple case: a sphere of radius $R = 1/10$ expands with a constant radial velocity $\mathbf{v} = V\mathbf{e}_r$, with $V = 1$, until it reaches a radius $R = 3/10$ at $t = 2R/V$ and then the motion is reversed until the sphere returns to its original radius at $t = 4R/V$ like in Gorges et al. (2022). They compared an edge fit method with the basic edge middle and barycentric interpolation (an interpolation method based on barycentric coordinates (fig. 3.3), introduced in Tryggvason et al. (2011)). The vertices are transported with exact velocities, so the errors are due to the remeshing procedure and the time integration. The remeshing method is employed at every time step.

Temporal integration. In our study, we use a first order Runge-Kutta scheme. The time step is $\Delta t = 0.5R/V$, except at the last iteration before the reversal: we set $\Delta t = 2R/V - t^{(n)}$ so that $t^{(n+1)} = 2R/V$ and the exact radius is $R = 0.3$. After that, the time step is $\Delta t = 0.5R/V$ with the reversed velocity $\mathbf{v} = -V\mathbf{e}_r$. We also set the last time step so that the last instant is $t = 4R/V$.

Mesh. We do a mesh convergence with four initial meshes comprised of 320, 1280, 5120 and 20480 triangles. The reference edge length is the mean initial edge length $l_m^{(0)}$, computed with the discretization of a sphere of radius $R = 0.1$. The edge length interval is: $[\lambda_{min}, \lambda_{max}]$, with $\lambda_{min} = l_m^{(0)}/2$ and $\lambda_{max} = 4l_m^{(0)}/3$. The reference edge lengths are given in table 4.1. In this case the exact interface is a sphere, the mean curvature is constant. The curvature-based remeshing criterion κd is deactivated in this study to concentrate on the influence of the edge splitting, collapsing and volume correction methods.

$\operatorname{card}\mathcal{T}$	320	1280	5120	20480
$l_m^{(0)}$	3.0×10^{-2}	1.5×10^{-2}	7.5×10^{-3}	3.8×10^{-3}

Table 4.1: Expanding and shrinking sphere, reference edge lengths for the four triangle mesh

PLAN OF THE STUDY. In the first part, we study the influence of the edge splitting methods only, for a mesh discretizing a sphere which expands and then shrinks. The errors (curvature, radius and volume) are computed for all vertices of the mesh $\mathcal{X}_\epsilon = \mathcal{X}$ at the middle of the simulation, when the radius is maximal, and at the end of the simulation. Usually, almost the whole mesh (i.e. the majority of the triangle edges) should be refined homogeneously at the same time in the first test since only the edge splitting is activated, the initial mesh is quite homogeneous and the magnitude of the velocity is spherically symmetric. In the second part, the only difference is that edge collapsing is activated. In the third part, the edge collapsing is deactivated but the influence of the volume correction procedure is examined. Finally, we run simulations with edge collapsing and volume correction.

ERRORS COMPUTED ON THE VERTICES. In Cohen-Steiner et al. (2004) the l^p "distance" (the definition here is one-sided, only from X to Y, and should be symmetrized with a contribution from Y to X to be a true distance) between a surface X and an approximating surface Y is defined as:

$$l_p = \frac{1}{S_X} \iint_{x \in X} \operatorname{dist}^p(x, Y)\, dx \tag{4.1}$$

with $\operatorname{dist}(x, Y) = \inf_{y \in Y} \|x - y\|$, where $\|\cdot\|$ is the Euclidean distance, and S_X is the area of the surface X. In our work, we associate each vertex i with the area S_i, which is the sum of the third of the area of each adjacent triangle: $S_i = \frac{1}{3} \sum_{T \in \mathcal{N}_1(i)} S_T$, as illustrated in fig. 4.1.

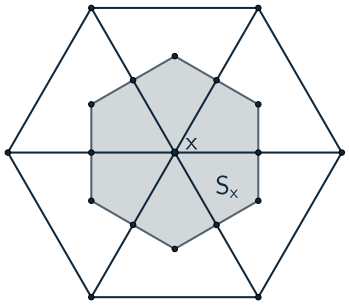


Figure 4.1: Area associated to the vertex, a third of the sum of the areas of the adjacent triangles, delimited by the polyline linking the barycenters of the adjacent triangles and the midpoints of the adjacent edges

We denote by κ_i the mean curvature computed at the vertex i with the surface reconstruction with the $\mathcal{N}_2$ stencil (section 3.3, page 24). At any time of the simulation, the exact surface is a sphere, so the reference value for the vertex i, denoted κ_i^e, is simply $-1/R(t)$ since we chose the exterior normal for the orientation of the surface. For each vertex i of radius r_i, the reference position is the point on the sphere at the minimal Euclidean distance: $r_i^e = R(t)$. In this work, we study the relative errors in the discrete l_2 and l_∞ norms:

$$l_2 = \sqrt{\frac{\sum S_i (\kappa_i - \kappa_i^e)^2}{\sum S_i (\kappa_i^e)^2}} \tag{4.2}$$

$$l_\infty = \frac{\max|\kappa_i - \kappa_i^e|}{|\max(\kappa_i^e)|} \tag{4.3}$$

Volume error... We have seen in section 2.10 that volume conservation is an issue and that it may be mitigated by improving the computation of the new vertex position during edge collapsing and splitting (section 3.6), or by applying a volume correction step. Gorges et al. (2022) examined a relative volume error for their test case of expanding and shrinking sphere. The volume at the start of the simulation is generally an approximation when it is computed based on the Lagrangian mesh (eq. 2.29), we will denote it by $\mathcal{V}$ as depicted in fig. 4.2.

...relative to the exact volume In this study, we define the volume error as the relative error between the discretized volume $\mathcal{V}$ and the exact volume of the sphere $\mathcal{V}^e$: $|\mathcal{V} - \mathcal{V}^e|/\mathcal{V}^e$. For instance, we have initially $|\mathcal{V}^{(0)} - \mathcal{V}^e|/\mathcal{V}^e \approx 0.8\%$ for a mesh approximating a sphere of radius $R = 0.1$ with 1280 faces (the vertices are exactly at the radius $R = 0.1$, which we denote $r_i = r_i^e$). This volume error diminishes as the number of triangles increases when the position is exact. In this case, we have to decide if we prefer correct vertex positions or a correct volume, the latter requiring to use a larger radius than the radius of the actual sphere $R(t)$ to have $\mathcal{V} = \mathcal{V}^e$ as illustrated by fig. 4.2.

...relative to the discretized volume We also compute the relative error between the final and initial discrete volumes $\mathcal{V}$ and $\mathcal{V}^{(0)}$. If we want to preserve the numerical volume $\mathcal{V}$ during the edge splitting (useful during the expansion phase only), the best way to remesh is to use the most basic remeshing method, which places the new vertex at the middle of the edge. This prevents mesh intersections as well when running simulations with several bubbles for instance. However the discrete volume is not conserved at the end of the simulation with the edge middle since the new points no longer are on the planes of the original triangles of $\mathcal{T}^{(0)}$.

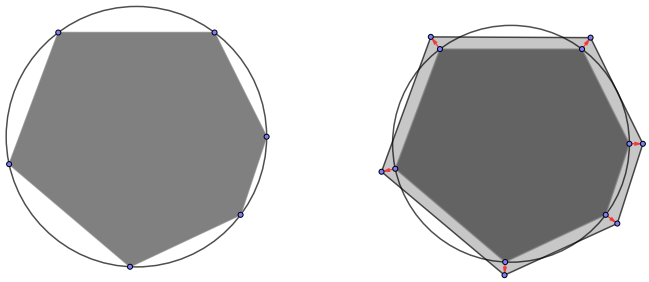


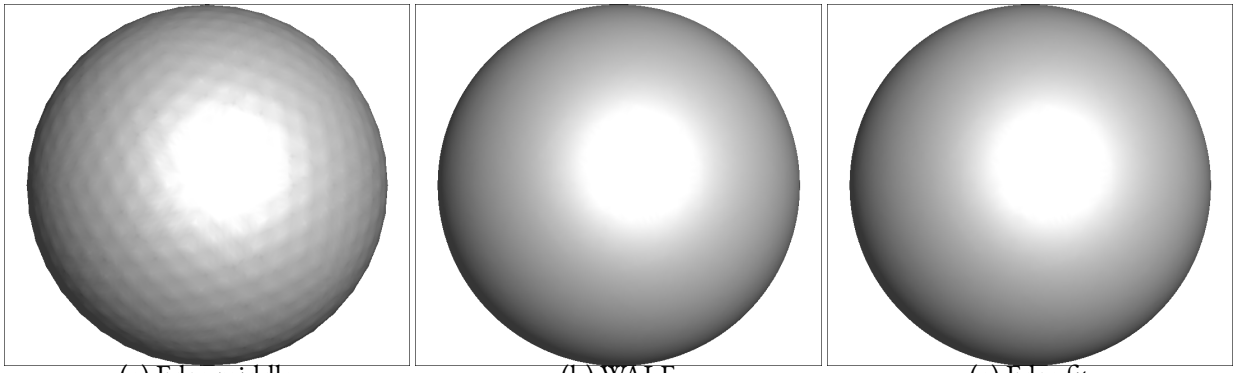
Figure 4.2: Exact and discrete volume, the discrete volume $\mathcal{V}$ is represented in gray (left). The vertices can be moved to retrieve the exact volume but the position is no longer exact (right).

4.1.1 Comparison of edge refinement methods, without collapsing, without volume correction (fig. 4.4, on page 43)

4.1.1.1 Qualitative comparison

We compare the edge refinement method presented in Gorges et al. (2022), which we refer to as "edge fit" (section 3.6) with the basic edge refinement at the middle of the edge and an edge refinement based on averaging two positions from reconstructed surface called WALF (Jiao and Wang, 2012). We only perform edge splitting, no volume correction step is applied. During the shrinking phase, since the edge collapsing is deactivated, both the old and new vertices are kept. The mesh is therefore comprised of vertices with lower position errors (initial vertices which were only transported to and fro) and new vertices which were either placed on the middle of a triangle edge or on a reconstructed surface. The result of the three methods at the end of the simulation are presented in

fig. 4.3, page 40. For the mesh with 1280 triangles, we can clearly see a difference between the basic edge refinement method and the two other methods based on local surface reconstruction. The surface from the former method is dimpled because of the position errors of the newly created vertices during the expansion, which do not lie on the sphere. From this illustration we can therefore expect larger errors on the radius and curvature for the splitting at the middle of the edge. The edge fit and WALF give similar results visually, hence the subsequent comparison with the l_∞ and l_2 norms.



(a) Edge middle (b) WALF (c) Edge fit

Figure 4.3: Comparison of edge refinement methods: edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30)

4.1.1.2 Quantitative comparison

Figure 4.4 illustrates the results of the simulation at the middle and end of the simulation. For a given abscissa, the markers of the figure are shifted left and right so there is no overlap, which could make it difficult to distinguish the three methods. The results are presented with the l_∞ norm here, since the l_2 norm gives similar trends as illustrated in fig. C.1, on page 116.

Middle of the simulation. At the middle of the simulation, when the exact radius equals $3/10$, all the edge refinements are performed since there is no other remeshing procedure such as edge collapsing afterwards and then the sphere shrinks. The curvature error in l_∞ norm seems to saturate already for the middle of the edge, as illustrated by the black markers in fig. 4.4, on page 43. For the four meshes, the WALF and edge fit methods do not seem to saturate for the curvature errors, and the edge fit method performs better. The edge middle splitting does not saturate for the radius error in l_∞ norm for the four meshes, its error is higher than the WALF and edge fit splittings. The edge fit performs better for the radius error. For the relative volume error, the edge middle gives worse results but does not saturate. The WALF and edge fit have similar error levels, except for the mesh with 320 triangles where the WALF has an error nearly a decade lower. The WALF and edge fit methods display 2^{nd} and 4^{th} order convergence for curvature and radius. The edge middle presents a 2^{nd} order convergence for the radius.

End of the simulation. The error levels are higher at the end of the simulation (fig. 4.4, page 43). The trends are the same for the curvature and radius errors at the end and middle of the simulation. The edge middle splitting saturates, while the other two do not. The edge fit gives lower errors. If we take the exact volume $\mathcal{V}^e$ as the reference, the volume error of the edge middle method is higher than the WALF and edge fit methods. If we take the initial discrete volume $\mathcal{V}^{(0)}$ as the reference, the trends are more regular and the error levels are similar for the three methods. When comparing the edge refinement methods, the new vertices indeed do not have the same position and do not form the same edge. Then the different edge lengths are treated differently in the subsequent remeshing procedures. The number of triangles are thus not exactly the same at the end of the simulation, but they are close. The approximate number of triangles at the end of the simulation $\mathcal{T}^f$, here without collapsing, is given in table 4.2. We can see here the usefulness of a collapsing procedure: the cost in memory and computation

time (indicated here by the number of triangles) can increase greatly without it. We have for example a mean edge length of 7.79×10^{-3} at the end of the simulation with the initial mesh of 320 triangles, while the reference length was 2.98×10^{-2} as indicated in table 4.1.

$\operatorname{card} \mathcal{T}$	320	1280	5120	20480
$\operatorname{card} \mathcal{T}^f$	5110	19720	80600	325640

Table 4.2: Expanding and shrinking sphere, approximate number of triangles at the beginning ($\mathcal{T}$) and end of the simulation ($\mathcal{T}^f$) without collapsing

4.1.1.3 Summary

Here several procedures were deactivated, the result would be quite different if the other procedures were activated: collapsing, swapping and smoothing. The edge middle saturates for curvature error. The surface reconstruction (WALF or edge fit) improves curvature, position and volume (relative to $\mathcal{V}^e$) errors compared to the edge middle, and the edge fit performs slightly better for the curvature and radius errors.

4.1.2 Comparison of edge refinement methods, with collapsing, without volume correction (fig. 4.5, on page 44)

Middle of the simulation. Edge collapsing mostly occurs after the velocity reversal, so we see little difference with the results without collapsing at the middle of the simulation (fig. 4.5, on page 44).

End of the simulation. With our edge length interval $[\lambda_{min}, \lambda_{max}]$, edge collapsing is effectively applied, the number of triangles at the end of the simulation (table 4.3) is reduced by 89 % when comparing to the simulations without edge collapsing (table 4.2).

$\operatorname{card} \mathcal{T}$	320	1280	5120	20480
$\operatorname{card} \mathcal{T}^f$	570	2220	9110	35170

Table 4.3: Expanding and shrinking sphere, approximate number of triangles at the beginning and end of the simulation without collapsing

With collapsing, the errors on curvature are lower than without collapsing for all three methods, and the errors of WALF and edge fit are more similar (fig. 4.5, page 44). The edge middle still saturates for curvature error. There is little difference for the radius error with and without edge collapsing. The volume errors are higher with collapsing than without collapsing, for instance, the errors of WALF and edge fit are nearly a decade higher. The volume errors are higher for the edge middle method than for surface reconstruction methods.

4.1.3 Comparison of edge refinement methods, with collapsing, with volume correction (fig. 4.6, on page 45)

Now, we consider the edge fit method and study the influence of volume correction while keeping the edge collapsing activated. We compare three computations without and with volume correction (HR and VBC, see section 2.10, on page 18) in fig. 4.6. When volume correction is activated and the edge collapsing is not, the results are similar (fig. C.2, on page 117). The volume errors with volume correction are not represented since they are smaller than the predefined activation threshold $\epsilon = 10^{-14}$ for the volume correction. The VBC is applied with almost homogeneous velocities ($v = \mathbf{e}_r$ except for newly created vertices, where we compute the average of the velocities of the vertices constituting the edge being refined) on an almost spherical surface so it gives similar results to the homothety (HR) in this test case.

Middle of the simulation. The volume correction does not seem to influence much the curvature error (fig. 4.6, on page 45). There is little difference for the position errors for the first mesh for the three cases, yet their order of convergence is halved when volume correction is activated. Both the exact volume and the exact position cannot be maintained, as illustrated by fig. 4.2: when the exact volume is conserved, the vertices no longer lie on the exact surface. The volume error without volume correction is of the order of 10^{-5}–10^{-3} when comparing to the exact volume.

End of the simulation. The volume correction does not seem to influence much the curvature error at the end of the simulation either (fig. 4.6, on page 45). All errors are higher at the end of the simulation, when comparing with the middle of the simulation with the edge fit method.

4.1.4 Summary

We have seen with this test case that surface reconstruction helps mitigate the position, curvature and volume error due to edge splitting when no volume correction is used. The curvature error saturates for the edge middle method while it does not for surface reconstruction methods for the four meshes. The edge fit method from Gorges et al. (2022) performs mostly better than the WALF method with $\mathcal{N}_2$. The error on curvature does not seem affected by volume correction in this test case. The order of convergence on position error is however affected. With this case the exact curvature is constant at each timestep, the shape is simple, the mesh is virtually homogeneous. The next test case will allow us to test the adaptive remeshing criteria along with the three edge splitting methods on a surface with spatially varying curvature.

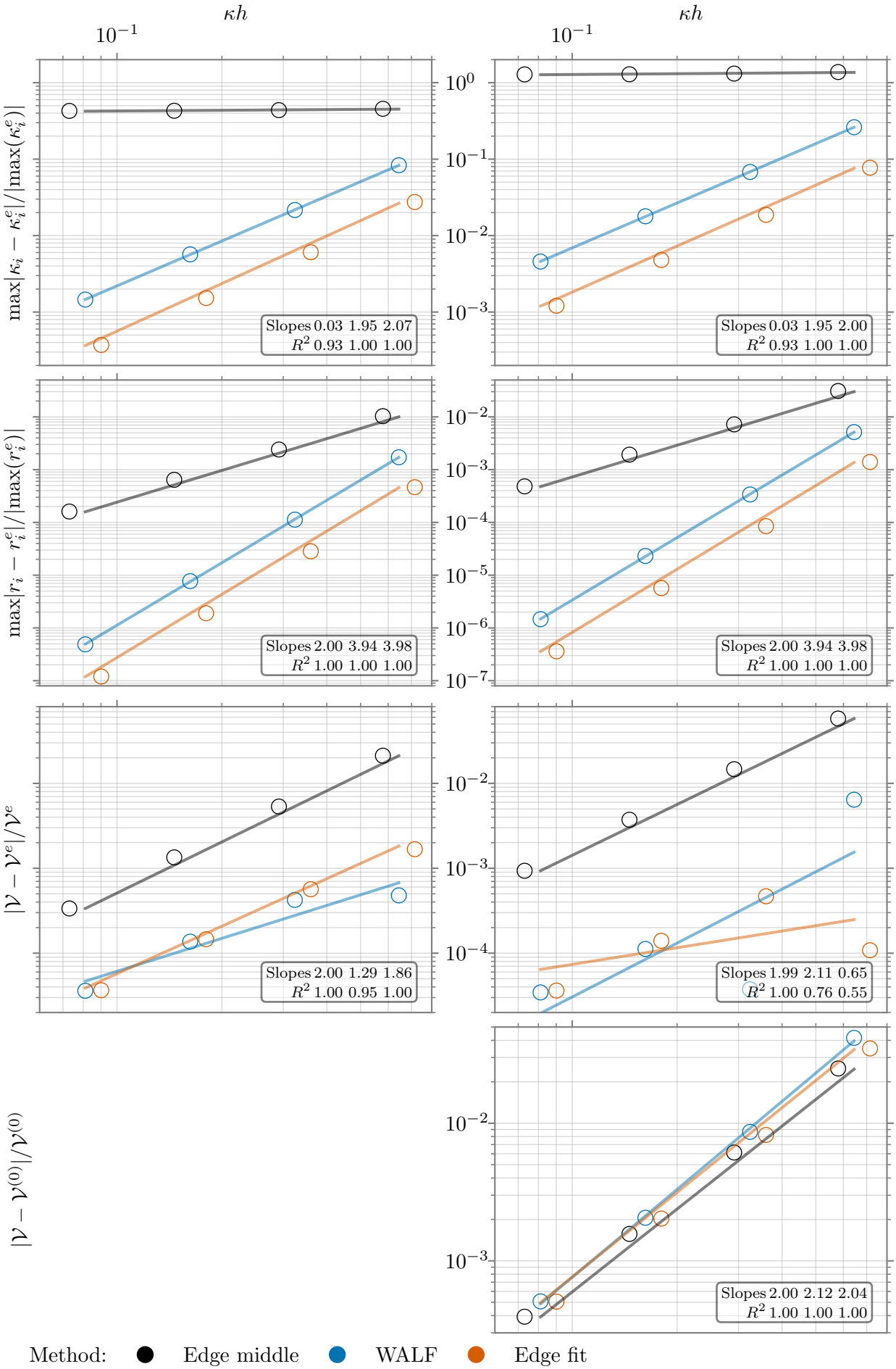


Figure 4.4: Sphere case, no edge collapsing, no volume correction, errors at the middle (left) and end of the simulation (right), edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30)

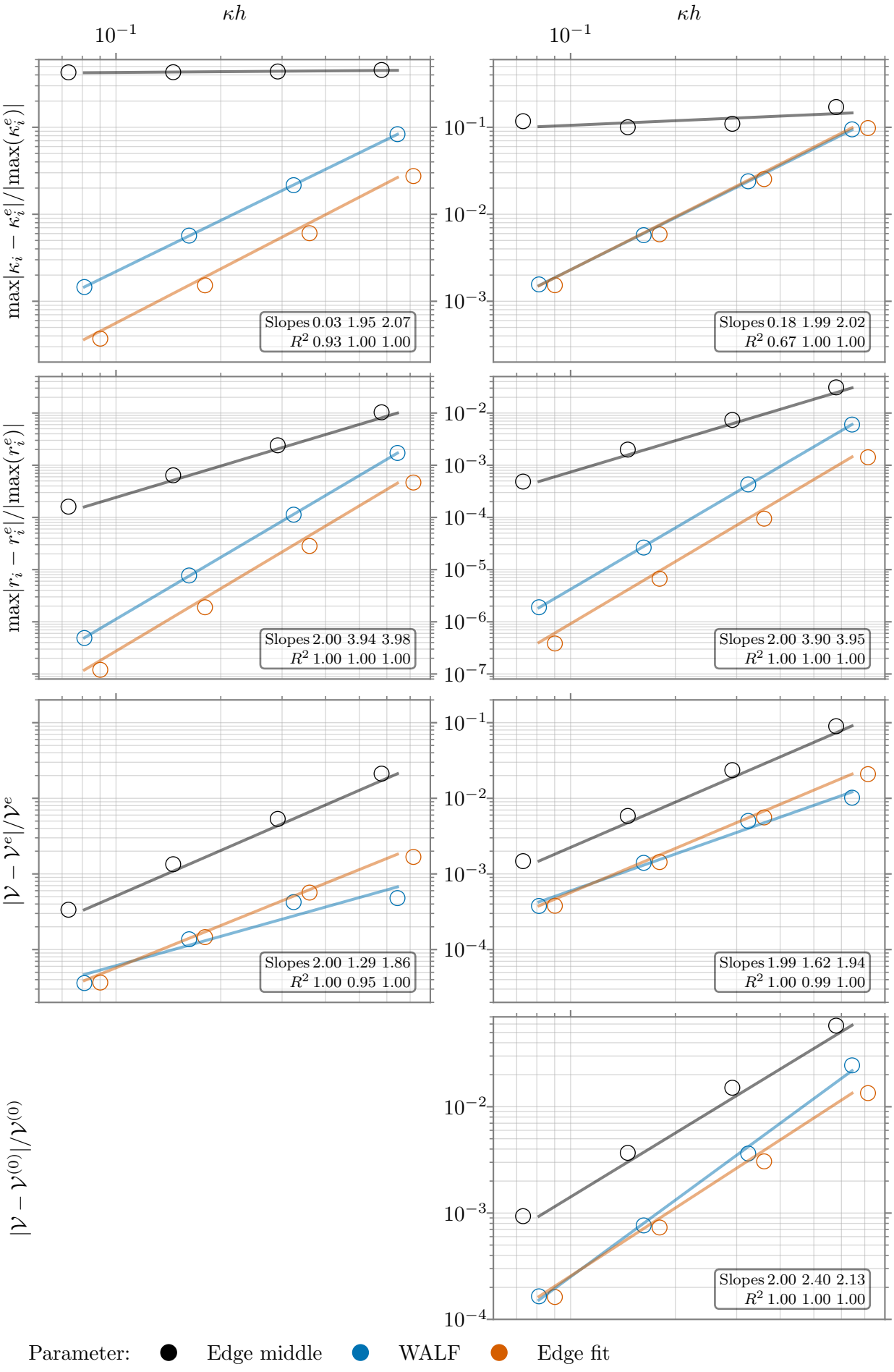


Figure 4.5: Sphere case, with edge collapsing, no volume correction, errors at the middle (left) and end of the simulation (right), edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30)

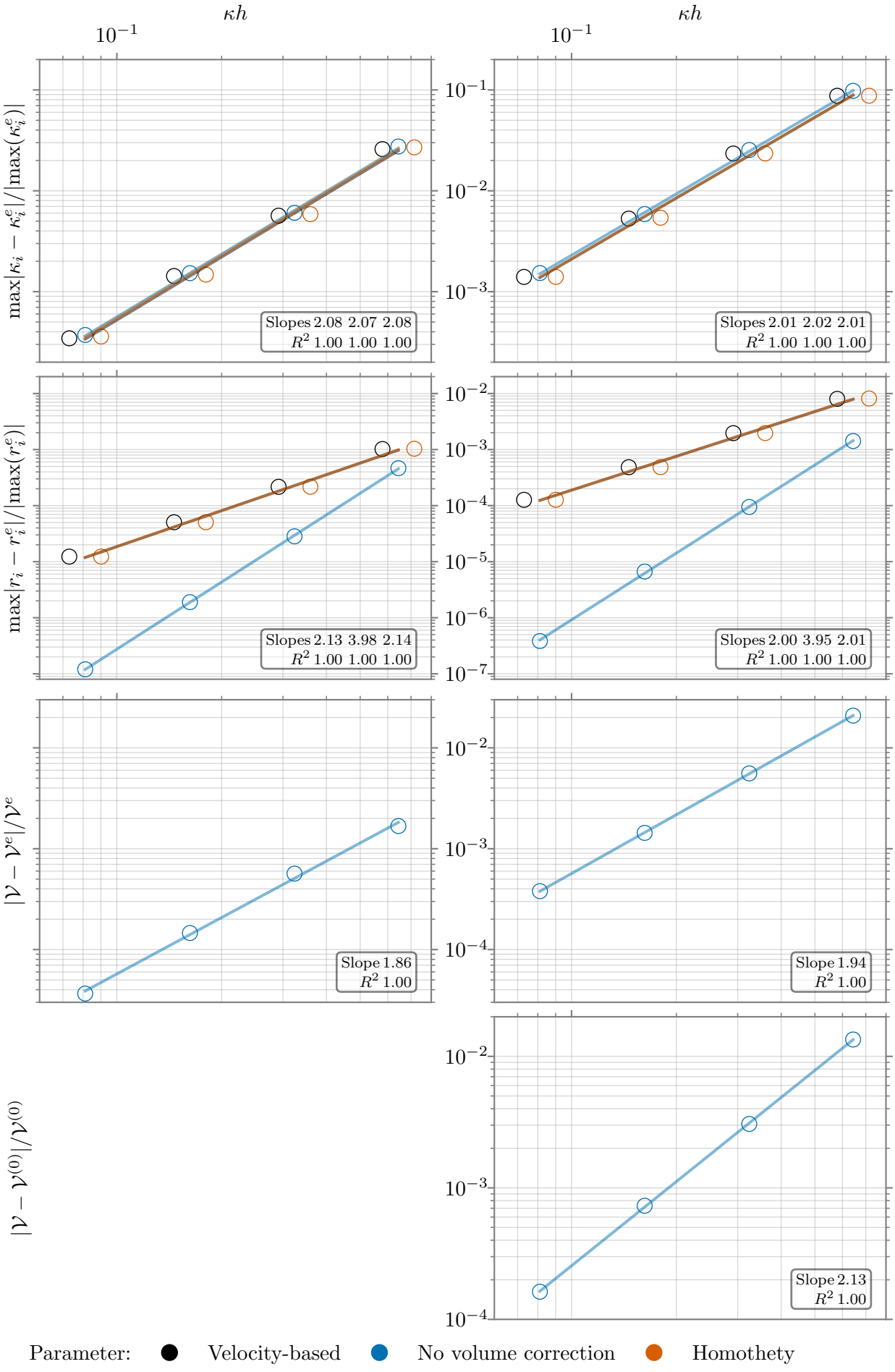


Figure 4.6: Sphere case, with edge collapsing, with volume correction, errors at the middle (left) and end of the simulation (right), velocity-based volume correction (VBC), no volume correction and homothety (HR) see section 2.10, on page 18

4.2 Curvature estimation and remeshing of a surface deformed in a radial velocity field

The aim of this section is to examine the remeshing procedure, especially the adaptation of the mesh to the curvature for a test case with an analytical solution from Koffi Bi et al. (2022). The mesh is adapted to the curvature with κd: κ is the curvature and d the edge length (section 3.5, on page 27). We can also see the influence of the transport: the errors may be accumulated, combining transport and remeshing. We continue the comparison of the three edge refinement methods (section 3.6, on page 29).

A divergence-free velocity field is imposed in a spherical coordinate system $(\mathbf{e}_r, \mathbf{e}_\theta, \mathbf{e}_\phi)$ either analytically or on the Eulerian grid (eq. 4.4) with $q = 1/30$. The constant q may be associated with the algebraic volume flow rate through the sphere of radius r: $4\pi q$. The initial interface is a plane sheet at $z_0 = 1/100$ in the Cartesian coordinate system $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$. The orientation of this open surface is given by the normal $\mathbf{n}$ so that $\mathbf{n} = \mathbf{e}_z = (0, 0, 1)$. The initially flat surface inflates with time with a positive flow rate.

$$\mathbf{v} = \frac{q}{r^2}\mathbf{e}_r, \theta > \pi/2 \tag{4.4}$$

4.2.1 Curvatures and normal of a surface of revolution

For a surface of revolution, the curvatures and normal can be computed analytically (Gray et al., 2017) in the Cartesian coordinate system. The angle θ belongs to $[0, \pi/2[$. The mean curvature is κ, κ_1 and κ_2 are the principal curvatures and κ_G is the Gaussian curvature. With the analytical velocity we can retrieve the analytical position. By integrating eq. 4.4 we obtain the radius:

$$r(\theta, t) = \sqrt[3]{3qt + \left(\frac{z_0}{\cos(\theta)}\right)^3} \tag{4.5}$$

By denoting $a(\theta) = r(\theta)\sin(\theta)$ and $b(\theta) = r(\theta)\cos(\theta)$, the curve can be parametrized in 2D in the Cartesian coordinate system by:

$$\theta \mapsto (a(\theta), b(\theta)) \tag{4.6}$$

The 3D surface is generated by rotating the two-dimensional curve (eq. 4.6) about the z-axis.

$$(\theta, \phi) \mapsto (a(\theta)\cos(\phi), a(\theta)\sin(\phi), b(\theta)) \tag{4.7}$$

We introduce $\mathbf{x} = (x, y, z)$ the parametrization:

$$\mathbf{x}(\phi, \theta) = r(\theta)(\cos(\phi)\sin(\theta), \sin(\phi)\sin(\theta), \cos(\theta)) \tag{4.8}$$

The formulae for the curvatures and normal of a surface of revolution are given in Gray et al. (2017) where $a \neq 0$ and $a'^2 + b'^2 \neq 0$:

$$\kappa = \frac{a'(a''b' - a'b'') - b'(a'^2 + b'^2)}{2|a|(a'^2 + b'^2)^{3/2}} \tag{4.9}$$

$$\kappa_G = \frac{-b'^2 a'' + a'b'b''}{a(a'^2 + b'^2)^2} \tag{4.10}$$

$$\kappa_1 = \frac{\mathrm{sgn}(a)(a''b' - a'b'')}{(a'^2 + b'^2)^{3/2}} \tag{4.11}$$

$$\kappa_2 = \frac{-b'}{|a|\sqrt{a'^2 + b'^2}} \tag{4.12}$$

$$\mathbf{n} = \frac{\text{sgn}(a)}{\sqrt{a'^2 + b'^2}}\big(\cos(\phi)b', \sin(\phi)b', a'\big) \tag{4.13}$$

In our case, we have $a'^2 + b'^2 \neq 0$ (eq. 4.14) and $a(0) = 0$. However, the expressions of κ, κ_G, κ_1 and κ_2 are defined at $\theta = 0$.

$$a'^2 + b'^2 = \frac{z^6 + 6qt\cos^5(\theta)z^3 + 9q^2t^2\cos^8(\theta)}{\cos^4(\theta)(z^3 + 3qt\cos^3(\theta))^{4/3}} \tag{4.14}$$

We take the orientation where the normal is pointing upward at $t = 0$ and thus the mean curvature is negative at $\theta = 0$ for $t > 0$. Since $\theta \in [0, \pi/2[$ the expressions in our case simplify to:

$$\kappa(\theta) = \frac{-3qt\cos^5(\theta)(3\sin^2(\theta)z^6 - 2\cos^2(\theta)z^6 + 12qt\cos^3(\theta)\sin^2(\theta)z^3 - 12qt\cos^5(\theta)z^3 - 18q^2t^2\cos^8(\theta))}{(z^3 + 3qt\cos^3(\theta))^{1/3}(z^6 + 6qt\cos^5(\theta)z^3 + 9q^2t^2\cos^8(\theta))^{3/2}} \tag{4.15}$$

$$\kappa_G(\theta) = \frac{-9q^2t^2\cos(\theta)^{10}(z^3 + 3qt\cos^3(\theta))^{1/3}(4\sin^2(\theta)z^3 - \cos^2(\theta)z^3 - 3qt\cos^5(\theta))}{(z^6 + 6qt\cos^5(\theta)z^3 + 9q^2t^2\cos^8(\theta))^2} \tag{4.16}$$

$$\kappa_1(\theta) = \frac{3qt\cos^5(\theta)(4\sin^2(\theta)z^3 - \cos^2(\theta)z^3 - 3qt\cos^5(\theta))(z^3 + 3qt\cos^3(\theta))^{2/3}}{(z^6 + 6qt\cos^5(\theta)z^3 + 9q^2t^2\cos^8(\theta))^{3/2}} \tag{4.17}$$

$$\kappa_2(\theta) = \frac{-3qt\cos^5(\theta)}{\sqrt{z^6 + 6qt\cos^5(\theta)z^3 + 9q^2t^2\cos^8(\theta)}(z^3 + 3qt\cos^3(\theta))^{1/3}} \tag{4.18}$$

Note that κ_1 and κ_2 can also be obtained with the mean and Gaussian curvature: $\kappa_1 = \kappa + \sqrt{\kappa^2 - \kappa_G}$ and $\kappa_2 = \kappa - \sqrt{\kappa^2 - \kappa_G}$. The unit normal to the surface (initially pointing upward $\mathbf{n} \cdot \mathbf{e}_z > 0$) is given by:

$$\mathbf{n} = \frac{1}{\sqrt{a'^2 + b'^2}}\big(\cos(\phi)\big(r\sin(\theta) - r'\cos(\theta)\big), \sin(\phi)\big(r\sin(\theta) - r'\cos(\theta)\big), r\cos(\theta) + r'\sin(\theta)\big) \tag{4.19}$$

Principal directions. The principal directions (directions in which the normal curvature equals the maximal and minimal curvatures) are the partial derivatives $\mathbf{x}_\phi$ and $\mathbf{x}_\theta$ for a surface of revolution when they exist ($\theta \neq 0$ in our case) (Gray et al., 2017), as depicted in fig. 4.7:

$$\mathbf{x}_\phi = r(\theta)(-\sin(\phi)\sin(\theta), \cos(\phi)\sin(\theta), 0) \tag{4.20}$$

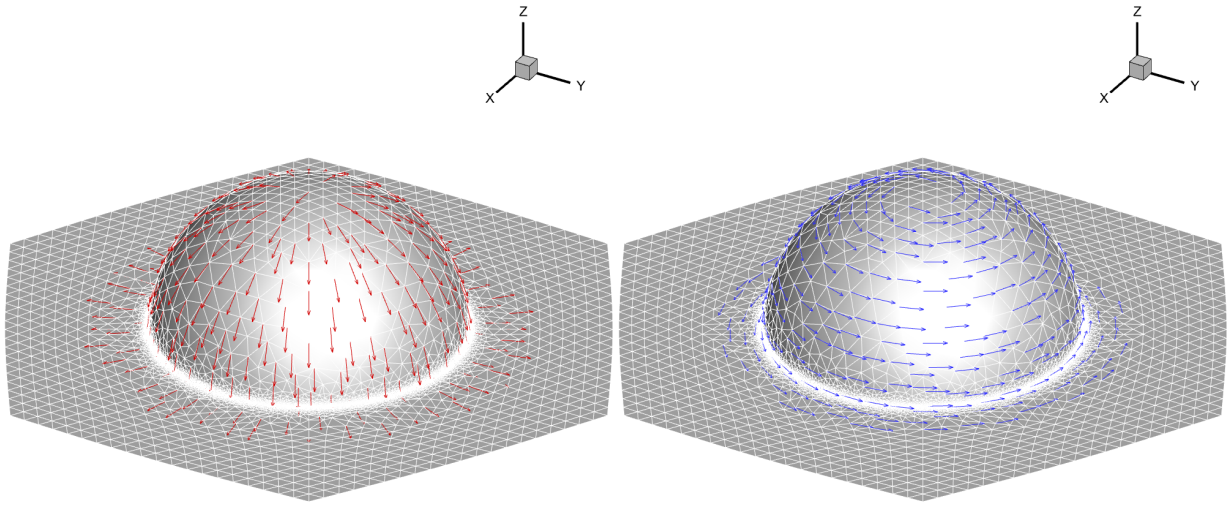
$$\mathbf{x}_\theta = \big(\cos(\phi)\big(r(\theta)\cos(\theta) + r'(\theta)\sin(\theta)\big), \sin(\phi)\big(r(\theta)\cos(\theta) + r'(\theta)\sin(\theta)\big), -r(\theta)\sin(\theta) + r'(\theta)\cos(\theta)\big)\big) \tag{4.21}$$

4.2.2 Transport of the mesh

4.2.2.1 Transport scheme

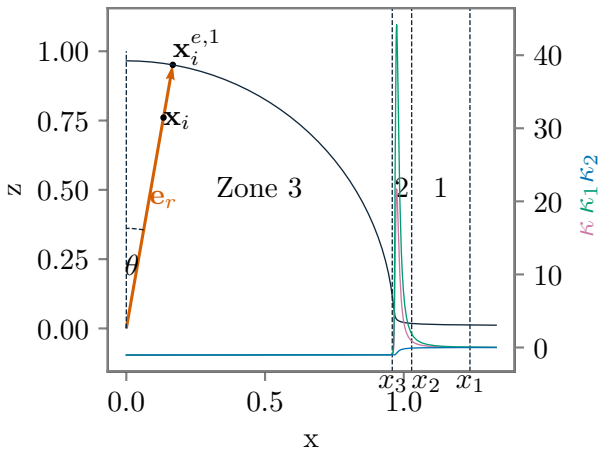
The mesh is transported with three different schemes to appreciate the combination of transport and remeshing errors: exact position, exact transport and interpolated velocity.

Exact position. The vertices are placed at the exact position along their radius at eachtime t: each vertex x_i of coordinates $\mathbf{x}_i = \big(r(t^{(n)}), \theta(t^{(n)}), \phi(t^{(n)})\big)$ is moved to $\mathbf{x}_i^e = \big(r(t^{(n+1)}, \theta(t^{(n)})), \theta(t^{(n)}), \phi(t^{(n)})\big)$ with eq. 4.5, as depicted in fig. 4.8a. The vertices are placed at the exact position at the end of the remeshing as well, with their new angles $\theta(t^{(n+1)})$ and $\phi(t^{(n+1)})$: $\mathbf{x}_i^e = \big(r(t^{(n+1)}, \theta(t^{(n+1)})), \theta(t^{(n+1)}), \phi(t^{(n)})\big)$.

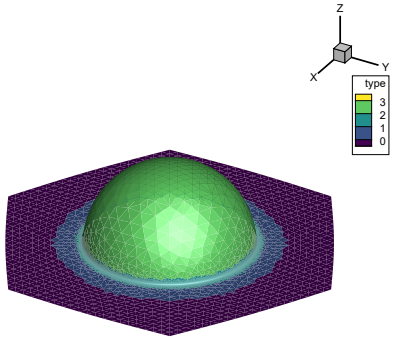


(a) $\mathbf{x}_\theta$ (b) $\mathbf{x}_\phi$

Figure 4.7: Normalized principal directions



(a) Shape, curvatures, reference point



(b) Error zones $\mathcal{Z}_1$, $\mathcal{Z}_2$, $\mathcal{Z}_3$, $\mathcal{Z}_0$ is not evaluated

Figure 4.8: Shape of the interface, mean curvature κ, principal curvatures κ_1 and κ_2, definition of the reference position $x_i^{e,1}$, zones of the domain

Exact transport. This is equivalent to a transport with the exact velocity since the velocity is constant. With this transport scheme, transport and remeshing errors are combined. Each vertex x_i of coordinates $\mathbf{x}_i = \left(r^{(n)}, \theta^{(n)}, \phi^{(n)}\right)$ is moved to:

$$\mathbf{x}_i^* = \left(\sqrt[3]{3q\Delta t + \left(r^{(n)}\right)^3}, \theta^{(n)}, \phi^{(n)}\right) \tag{4.22}$$

Interpolation. From $t = t^{(n)}$ to $t = t^{(n+1)} = t^{(n)} + \Delta t$, each vertex x_i is transported according to (eq. 2.13, on page 15) by a second-order Runge-Kutta scheme. The velocity $\mathbf{v}_i$ of the vertex x_i is interpolated from the Eulerian grid with the Parabolic Edge Reconstruction Method (McDermott and Pope, 2008). The exact velocities are set on the Eulerian grid with eq. 4.4.

4.2.2.2 Time stepping

Each vertex is transported at each iteration with a given time step Δt. The time step can be determined with the exact maximal velocity magnitude $v = q\left(3qt + {z_0}^3\right)^{-2/3}$ and either:

- an Eulerian Courant number $Co = v\Delta t/h = 1/2$ where h the Eulerian mesh spacing is the reference length.
- a Lagrangian Courant number $Co = v\Delta t/d = 1/2$, the minimal edge length of the Lagrangian mesh serves as the reference length.

The Lagrangian Courant number is mostly used when both the transport with interpolation and adaptive refinement to the curvature are used in the comparisons. However, with more iterations, errors may accumulate more. Otherwise we employ the Eulerian Courant number.

4.2.3 Remeshing

Eulerian resolution. In all simulations we define an Eulerian grid spacing h (fig. 4.9). For the transport with interpolation, the velocities are computed on the Eulerian grid with this spacing h and then interpolated at the marker positions. In the interval $[\lambda_{min}, \lambda_{max}]$, with $\lambda_{min} = h/2$ and $\lambda_{max} = 3h/2$, based on the Eulerian grid spacing h, there is no splitting or collapsing of triangle edges when the curvature criterion is not involved.

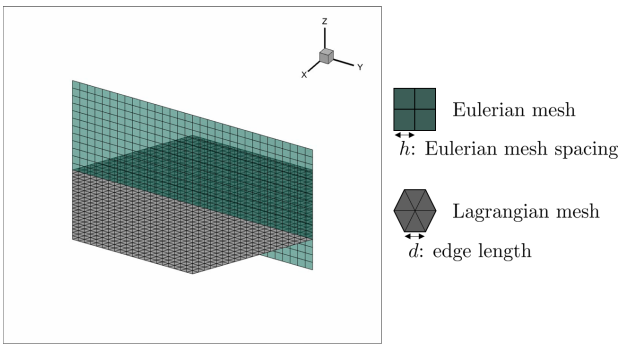


Figure 4.9: Eulerian and lagrangian meshes

Curvature-based criterion. In this test case, the numerical curvature κ_i of a vertex x_i of coordinates (r, θ, ϕ) can be replaced with the exact curvature $\kappa_i^e(\theta)$ (eqs. 4.17 and 4.18) in the curvature-based remeshing criterion (eq. 3.10, on page 28). In a first part of this study, the exact curvature is used and then we can appreciate the influence of the approximation of the curvature in the remeshing criterion.

Procedures. The edge collapsing, splitting, swapping and removal of bad triangles are activated. Volume correction is not activated for the surface is open. The smoothing procedure is not activated either. A basic Laplacian smoothing would not give satisfactory results since the lost volume cannot be recovered, though this loss is mitigated when combining it with surface reconstruction.

4.2.4 Surface initialization

4.2.4.1 Flat surface

For this test case, the simulation is performed in two steps. First, the surface is initialized as a homogeneously discretized (equilateral triangles) hexagonal sheet at $t = 0$ at $z = 10^{-2}$. The initial mesh resolution is denoted $d \in [\lambda_{min}, \lambda_{max}]$, with $\lambda_{min} = h/2$ and $\lambda_{max} = 3h/2$. Since the surface is flat, the curvature criterion, when it is activated, does not yet come into play.

4.2.4.2 Spike generated by the first transport

Next, the mesh should be transported for the first time. Nevertheless, because of the position of the sheet $z = 10^{-2}$, there is a large angle between the position vector of the vertex at $\left(0, 0, 10^{-2}\right)$ (always present by construction of the homogeneous mesh) and that of the neighboring vertices, resulting in an important difference in velocity (eqs. 4.4 and 4.5). At the first time step, the surface may thus appear almost flat if not for a spike at its center. With such a mesh the criterion of refinement based on the curvature κd would not be respected when it is activated. With a spike, the curvature-based remeshing would tend to overly refine the mesh around the spike, which could lead to large errors, especially when creating new vertices with the WALF or the edge fit. We indeed do not filter the undulations or weight with the normal in the surface reconstruction (section 3.3, on page 24). We perform an initialization procedure with a succession of remeshing and repositioning to handle this spike, which is described below.

4.2.4.3 Succession of remeshing and repositioning

We choose a time $t = t_{init} > 0$ so that the surface is no longer flat and we adapt the mesh. We remesh and reposition the vertices on the exact surface at $t = t_{init}$ as long as the curvature-based remeshing criterion κd is not respected for the whole mesh:

$$(\kappa d)_{\mathcal{X}} = \max_{x_P \in \mathcal{X}} \left(\kappa_{max}(x_P) \max_{P_1 \in \mathcal{N}_1(P)} \|\mathbf{x}_{P_1} - \mathbf{x}_P\| \right) < \kappa d_{max} \tag{4.23}$$

Here the curvature is non-dimensionalized by the maximal edge length on the $\mathcal{N}_1$ stencil. Once the mesh corresponds to the remeshing criterion κd, the surface can be transported up to $t = 9s$ with the transport scheme under study (exact position, exact transport, interpolated velocity). The maximal curvature of the exact surface evolves through time, as depicted in fig. 4.10. We initialize the mesh based on the exact curvatures computed at

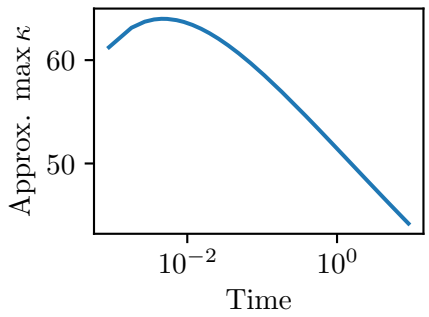


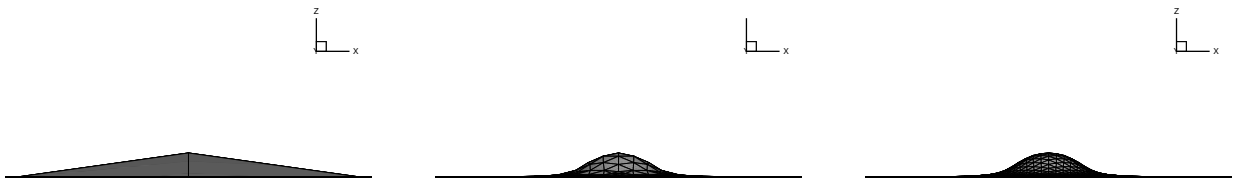
Figure 4.10: Approximate maximal principal curvature in absolute value: $\max_{i \in 0, n_\theta - 1, n_\theta = 1000} \kappa(89° * i/(n_\theta - 1))$

discrete positions to respect the criterion κd. We do not have access to the exact maximal principal curvature in absolute value over the whole continuous surface. For different initialization times, the curvatures, reference lengths and meshes are different and so are the curvature errors in l_∞ and l_2 norms that vary as we can see in table 4.4. We chose arbitrarily $t_{init} = 10^{-4}s$. The influence of κd is illustrated in fig. 4.11 and the curvature errors

t_{init}	κd	l_∞	l_2	Vertices
0.01	1/4	0.120	0.07	4627
0.001	1/4	0.215	0.104	2747
0.0001	1/4	0.121	0.124	1657

Table 4.4: Curvature errors at several initializations for radial case

are given in table 4.5. We can see here that the minimal roughness is not very helpful in this case to analyse the results. At this instant, $\max(z) \approx 2.2 \times 10^{-2}$, and we have $\kappa(\theta = 0) \approx \kappa_1(\theta = 0) \approx \kappa_2(\theta = 0) \approx -40.876$. We chose this initialization time because it is relatively small and gives a small number of vertices so that a large part of the simulation time is simulated outside of this initialization procedure, and the maximal principal curvature is high already. In this case, we can see that when the curvature-based refinement is deactivated ($\kappa d = 0$), the error is larger at the end of this initialization procedure. Note that we still impose a minimum mesh resolution based on the Eulerian grid, even when the criterion κd is deactivated, which can be observed in fig. 4.11a.



(a) 0 segments per radius of curvature

(b) 2 segments per radius of curvature

(c) 4 segments per radius of curvature

Figure 4.11: Initialization at $t = 10^{-4}s$

κd	$1/(\kappa d)_{\mathcal{X}}$	l_∞	l_2	Roughness	Vertices
0	0.242	0.992	0.992	0.998	1261
1/2	2.291	0.294	0.393	0.988	1401
1/4	4.489	0.120	0.124	0.996	1657

Table 4.5: Curvature errors at $t = 10^{-4}s$ for several κd, $(\kappa d)_{\mathcal{X}}$ is the numerical value computed with eq. 4.23

4.2.5 Zones of computation

The errors are computed up to $\theta < \theta_0 = \arctan(L_0/z_0)$, with $L_0 = 1$ and $z_0 = 10^{-2}$ (zone limit x_1 in table 4.6). The boundaries of the open mesh are thus not included in the error computations. We define three zones $\mathcal{Z}_1$, $\mathcal{Z}_2$, $\mathcal{Z}_3$ within this computation domain, delimited by $\theta_2 = 89°$ and $\theta_3 = 83°$, as illustrated in fig. 4.8b. The aim of delimiting these zones is to appreciate the influence of the curvature of the surface on the computation of errors. The mean and principal curvatures are depicted in fig. 4.8a. The values of the principal and mean

curvatures at the zone limits are given in table 4.6. We are especially interested in zones 2 and 3, the former has the largest curvature, the latter is almost spherical. We could expect similar trends as in section 4.1 for the errors in zone 3. The principal directions are not defined at $\theta = 0$.

Abscissa	κ_1	κ_2	κ	$90 - \theta(^\circ)$
x_1	5,518E-02	-7,264E-03	2,396E-02	5,729E-01
x_2	1,849E+00	-8,078E-02	8,843E-01	1
x_3	-8,664E-01	-1,035E+00	-9,506E-01	7
$\arg\max\lvert\kappa(\theta)\rvert$	4.419E+01	-7.619E-01	2.171E+01	1.927E+00
$\arg\max\lvert\kappa_1(\theta)\rvert$	4.419E+01	-7.637E-01	2.171E+01	1.930E+00
$\arg\max\lvert\kappa_2(\theta)\rvert$	-1.036E+00	-1.036E+00	-1.036E+00	$[89, 90]$

Table 4.6: Zone definitions, the mean and principal curvatures are indicated for the zone limits (between zones $\mathcal{Z}_1$, $\mathcal{Z}_2$, $\mathcal{Z}_3$ and $\mathcal{Z}_0$), the approximated location of the maximal curvatures in absolute values and their values are indicated

4.2.6 Comparison of remeshing methods

We compare the remeshing methods by the meshes they produce. Theses meshes went through different remeshing operations and perhaps even different number of time steps if the time-step is set with the minimal edge length (subsection 4.2.2.2). The meshes are compared with curvature and position errors since our purpose in the long run is to know the position of an interface precisely and to be able to evaluate the surface tension accurately while keeping down the computation and memory cost.

4.2.6.1 Possible causes of differences in errors

When comparing two remeshing methods, if one method presents larger errors in l_∞ and l_2 norms, several factors may come into play. The distribution of the vertices of the $\mathcal{N}_2$ neighborhood may influence the curvature errors. A vertex and its neighbors may have a larger position error, decreasing the precision of the surface reconstruction. One method may employ more remeshing operations or time steps in the case of the Lagrangian Courant number, thus accumulating more position errors. Larger errors may also stem from the angle θ of the vertex. It is possible that depending on the zone (zones 1: almost flat, 2: higher curvature, 3: almost spherical depicted in fig. 4.8b), the error level may be different. It is often difficult to draw conclusions as to why one remeshing method fared better, especially for the l_∞ norm. The relative error in l_2 norm oscillates less than its counterpart in l_∞ norm.

4.2.6.2 Abscissa in the figures

In the next figures such as fig. 4.13, we chose the Eulerian grid spacing h as the reference length, it is the maximal edge length over the whole mesh. We approximate numerically the (absolute) maximal mean curvature to compute the abscissa κh with h the Eulerian grid spacing. The abscissa κh is different from κd, with d the edge length, related to the Lagrangian mesh and used to control the mesh with the curvature. We also draw relative errors against the number of vertices N at the end of the computation, which is an indication of the memory cost. Depending on the parameter κd, the remeshing procedure may produce quite different numbers of vertices since the edge length is allowed to be lower than $h/2$.

4.2.6.3 Reference values

To compute the errors on the vertices in l_2 and l_∞ norms, like in eqs. 4.2 and 4.3, we need to define a reference position on the exact surface for each vertex. In the case of the expanding and shrinking sphere, the reference position was easily defined because the exact surface was a sphere at the middle and end of the simulation. For each vertex of radius r_i, the reference position is the point on the sphere at the minimal Euclidean distance, the

position error is simply: $|r_i - R|$, and the mean curvature error is simply $|\kappa_i - (-1/R)|$. In this test case, we have a mesh at the end of the simulation and we want to compare it to the exact surface. The closest point on the exact surface to the vertex x_i is not found as easily as in the case with the sphere [1]. When using the transport with exact positions, we simply compute the reference values at the same angle $\theta^{(n)}$ for the vertex x_i of coordinates $\mathbf{x}_i = \left(r^{(n)}, \theta^{(n)}, \phi^{(n)}\right)$ like in Koffi Bi et al. (2022). However, when the transport is not exact, the vertices do not necessarily lie on the exact surface. We define a first reference position: each vertex $x_i(t)$ of coordinates $\mathbf{x}_i = \left(r^{(n)}, \theta^{(n)}, \phi^{(n)}\right)$ is associated with the point $x_i^{e,1}$ located on the analytical surface $S^e(t)$, of coordinates $\mathbf{x}_i^{e,1} = \left(r(\theta(t^{(n)}), t^{(n)}), \theta(t^{(n)}), \phi(t^{(n)})\right)$ with eq. 4.5 (on page 46), as illustrated in figs. 4.8a and 4.12. This is the same procedure as in the "exact transport" (subsection 4.2.2.1). The vertex x_i is associated with the point $x_i^{e,1}$ on the exact surface, they have the same angle $\theta(t^{(n)})$. It should be noted that when computing the position error with $x_i^{e,1}$, along $\mathbf{e}_r(x_i)$, $x_i^{e,1}$ is not necessarily the closest point on the exact surface as illustrated by fig. 4.12. We can approximate the cooordinates of the closest point, denoted $x_i^{e,2}$, on the exact surface to the vertex x_i by

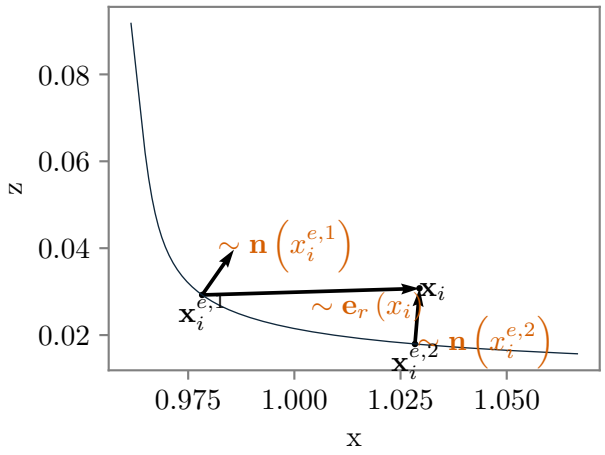


Figure 4.12: Minimal distance and radial distance to the exact surface: $x_i^{e,1}$ is not the position at the minimal distance, contrary to $x_i^{e,2}$, whose normal to the surface is colinear to the vector $(\mathbf{x}_i - \mathbf{x}_i^{e,2})/\left\|\mathbf{x}_i - \mathbf{x}_i^{e,2}\right\|$

minimizing $g(\theta) = (r(\theta)\cos(\phi)\sin(\theta) - x_i(1))^2 + (r(\theta)\sin(\phi)\sin(\theta) - x_i(2))^2 + (r(\theta)\cos(\theta) - x_i(3))^2$ with $\mathbf{x}_i = (x_i(1), x_i(2), x_i(3))$ the coordinates of the vertex x_i of the Lagrangian mesh. With Newton's method and the angle $\theta^{(0)} = \theta(t^{(n)})$ as the initial value, we can estimate the angle of the closest point on the exact surface. The surface is continuous and differentiable, with no corner (the initial surface extends infinitely in xy plane) or sharp point, so the minimal distance from one point to the surface is obtained by solving $f(\theta) = 0$, with:

$$f(\theta) = \frac{\partial g(\theta)}{\partial \theta} \tag{4.24}$$

We solve for $f(\theta) = 0$ with Newton's method: $\theta^{(m+1)} = \theta^{(m)} - f(\theta^{(m)})/f'(\theta^{(m)})$. If $\theta^{(m+1)} < 0$ we take $\theta^{(m+1)} = 0$ and $\theta^{(m+1)} > \pi/2$ we take $\theta^{(m+1)} = \pi/2$. The stopping criterion is $\left(\theta^{(m+1)} - \theta^{(m)}\right)/\theta^{(m+1)} < \epsilon$ with $\epsilon = 10^{-14}$. At convergence (by denoting θ the angle with the smallest distance), we should have a smaller position error: $\sqrt{g(\theta)} < \sqrt{g(\theta^{(0)})}$ and $\mathbf{n}\left(\theta^{(0)}\right) \cdot \frac{\mathbf{x}_i - \mathbf{x}(\theta^{(0)})}{\left\|\mathbf{x}_i - \mathbf{x}(\theta^{(0)})\right\|} < \mathbf{n}(\theta) \cdot \frac{\mathbf{x} - \mathbf{x}(\theta)}{\|\mathbf{x} - \mathbf{x}(\theta)\|}$. We keep the angle with the lowest distance from the vertex x_i to the exact surface (we do not check the improvement on the normal, which should be colinear to the vector $\frac{\mathbf{x} - \mathbf{x}(\theta)}{\|\mathbf{x} - \mathbf{x}(\theta)\|}$).

[1] With our test case, we could define a reference dicretized surface $\mathcal{T}_n$ from a computation with a fine Lagrangian mesh transported with exact positions and compute a "distance" between the two meshes. We could then compute a distance between the reference mesh and the discretized surface under study like in Garland and Heckbert (1997); Hu et al. (2017), as mentioned in appendix D.1.

4.2.7 Plan of the study

The curvature is computed with the weighted least-squares formulation presented in section 3.3. The curvature and position errors are computed for all vertices of the mesh in zones $\mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \mathcal{Z}_3$, $\mathcal{Z}_2$ and $\mathcal{Z}_3$ at the end of the simulation at $t = 9s$. The main parameters employed in this study are summarized in table 4.7. In the first part, we study the curvature error with exact positions, to appreciate the influence of the curvature-based remeshing criterion κd, the parameter λ involved in smoothing the curvature information for the adaptation of the mesh to the curvature and the effect of weighting in the least-squares formulation for the evaluation of the curvature. In the second part, we employ the exact transport to appreciate the performance of the edge refinement methods (section 3.6, on page 29) when position errors accumulate. We also appreciate the influence of weighting on WALF and the edge fit method. In the third part, we employ the transport with interpolated velocities. The reader may find supplementary information in appendix D.

$q(m^3/s)$	$z(m)$	$t_i(s)$	$t_f(s)$
$1/30$	0.01	0.0001	9

Table 4.7: Main parameters used in the radial transport case

Presentation of the results. In the subsequent figures (such as fig. 4.13), when a circular marker is empty, the remeshing was done without adaptation to the curvature with the non-dimensionalized curvature κd, with κ the curvature and d the edge length. Otherwise κd is used and its target value κd_{max} is the value indicated in the caption, and we take $\kappa d_{min} = \kappa d_{max}/4$ (section 3.5). The computations with a small κd criterion (especially $\kappa d = 1/32$) which use more computational resources are not presented. The total number of vertices at the end of the simulation is denoted N, κ is the maximal mean curvature in absolute value, h is the reference edge length corresponding to the Eulerian grid. For a given error, we plot two figures on a row: on the left, we plot against κh, and on the right against $1/\sqrt{N}$. For the figure on the left, the markers are aligned on a vertical, corresponding to the number of Eulerian grid cells in the x-direction: $\kappa h \approx 10.8, 5.4, 2.7, 1.3, 0.67, 0.34$, that is $nx = 8, 16, 32, 64, 128, 256$ (fig. 4.13). When reading curvature errors for a given simulation, the reader may read the size of the mesh on the right with the equivalent point (at the same ordinate), as illustrated in fig. 4.13. For the sake of visibility, the markers of the figures on the left are slightly shifted left and right to avoid overlappings when studying the influence of a parameter, such as in fig. 4.15.

4.2.8 Exact position

The exact position is used here (eq. 4.5), so the errors come only from the surface reconstruction method (section 3.3, on page 24) and the distribution of the vertices along the surface.

4.2.8.1 Influence of number of segments per radius (fig. 4.13, on page 55)

We only study the classic edge refinement at the middle, since the position is exact, the difference would be in the distribution of the points on the surface with the WALF and edge fit methods. Indeed, the improvement of the position of the new vertex with surface reconstruction cannot be observed when the vertices are placed on the exact surface. Figure 4.13 shows the influence of the number of segments per radius in $\mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \mathcal{Z}_3$, which we call the whole domain.

Curvature error against κh (left figure). The relative error in both l_2 and l_∞ (fig. D.1, on page 119) norms generally decreases with the number of segment per radius in the case of the exact position: for a given Eulerian resolution, indicated by κh, we have here $\epsilon_- > \epsilon_{1/2} > \epsilon_{1/4}$ and so on. With a fine enough homogeneous mesh, the criteria $\kappa d = 1/2^i$ should be satisfied, and we should have $\epsilon_- \approx \epsilon_{1/2^i}$, which can be noticed with the highest resolution ($\kappa h \approx 0.3$): we have $\epsilon_- \approx \epsilon_{1/2}$, this homogeneous nearly satisfies $\kappa d < 1/2$. In zone 3, this is more noticeable for the two finest Eulerian resolutions (fig. D.2, on page 120). For coarse Eulerian resolution

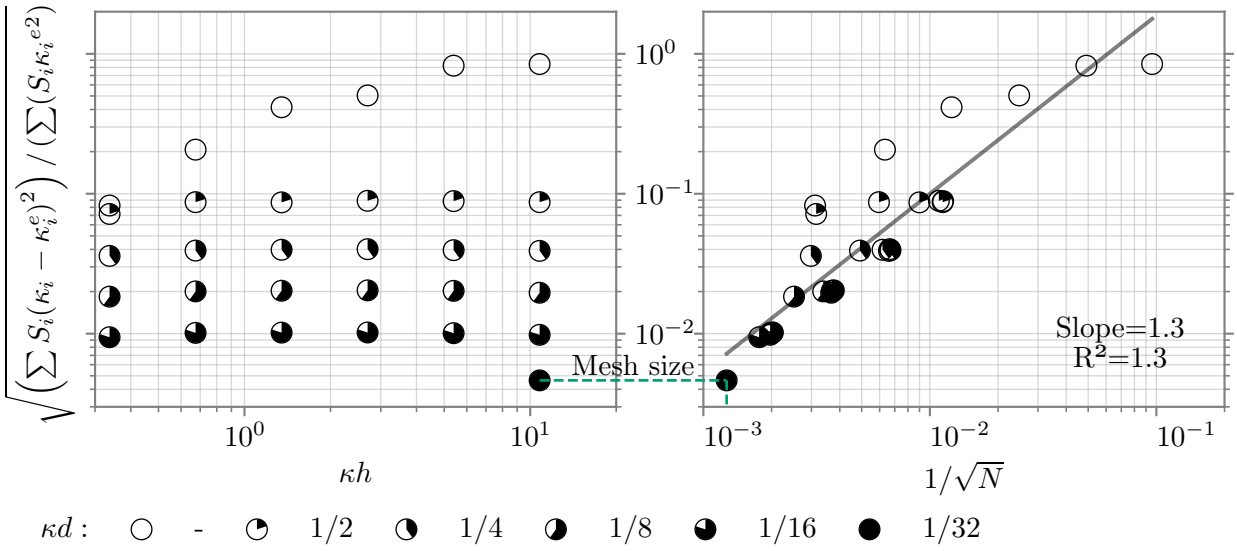


Figure 4.13: Transport with exact position, whole domain ($\mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \mathcal{Z}_3$), on the left κh with κ the maximal mean curvature, h the Eulerian grid spacing, the more a marker is filled, the smaller κd is, with d the edge length (when curvature adaptation is deactivated, we denote it by '-' or $\kappa d = 0$), N the number of vertices, the figure is in log-log scale; as illustrated by the dashed line, one may read the inverse of the square root of the number of vertices by reading the plot on the right, with the abscissa of the point at the same ordinate, l_∞ norm in fig. D.1 (on page 119)

h, the Eulerian resolution h has little influence once the curvature criterion is activated. In l_∞, the error level oscillates more for the same κd and varying κh.

Curvature error against κh (left figure) and $1/\sqrt{N}$ (right figure). For a homogeneous mesh ($\kappa d = 0$), using a finer resolution (smaller κh) implies a higher number of vertices at the end of the simulation as illustrated on the left figure (fig. 4.13) and by fig. 4.14. For the same Eulerian resolution (κh), adapting the mesh more to the curvature with κd implies a cost (regarding the number of vertices) superior or similar to a simulation with a smaller adaptation to the curvature. An Eulerian grid of $\kappa h \approx 0.67$ with $\kappa d = 0$ requires more vertices to reach the same resolution as a mesh with $\kappa h \approx 10.8$ and $\kappa d = 1/2$. For the same κd and different κh, we could expect the difference in the number of vertices to come from zone 1 and 3, which are respectively flat and virtually spherical. From Koffi Bi et al. (2022), we know that the error on the curvature estimation with surface reconstruction converges for an exact position and an homogeneous mesh, up to $\kappa h \approx 2{\cdot}10^{-2}$ at least. The difference here is that the mesh is virtually homogeneous if our curvature criterion is deactivated and adapted to the curvature otherwise, so there are more points in zone 2, the zone with the largest mean curvature. For coarse Eulerian resolutions, an equivalent number of vertices and exact positions, we could expect a mesh refined with a given κd to give a lower error in l_2 norm. We can see that an Eulerian grid $nx = 8$ ($\kappa h \approx 10.8$) with $\kappa d = 2$ has nearly $7{\cdot}10^3$ vertices and a lower error level in both l_2 and l_∞ than a homogeneous mesh $nx = 128$ and $2{\cdot}10^4$ vertices. Figures D.2 and D.3 show the influence of the number of segments per radius in zones 3 and 2. Zone 3 is virtually spherical and generally presents less errors than zone 2 in l_2 norm, but in some cases it is the other way around in l_∞ norm.

Summary. With exact position, refining with curvature can improve the evaluation of curvature for coarse meshes.

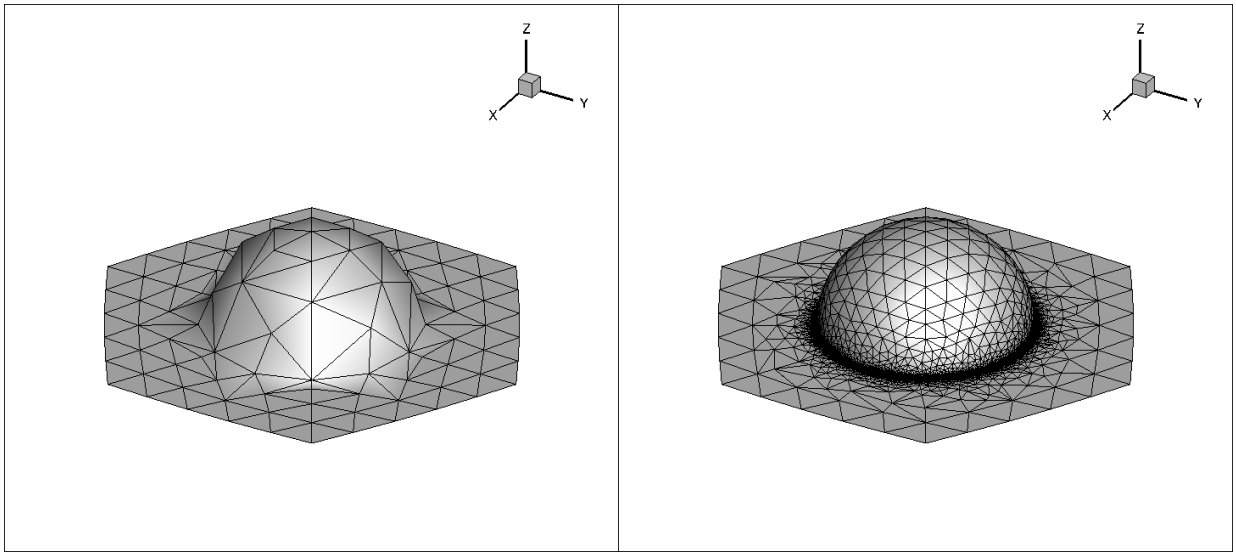


Figure 4.14: Transport with exact position, $nx = 8$, $\kappa d = 0$ (left) and $\kappa d = 1/2$ (right)

4.2.8.2 Influence of refinement diffusion (fig. 4.15, on page 56)

The refinement diffusion is controlled with a parameter λ: with $\lambda = 1$, it tends to produce a mesh homogeneously adapted to the largest curvature, while with $\lambda = 0$, the mesh is coarser and more heterogenous (eq. 3.10). With the exact position, when we diffuse the information of the curvature (for the mesh adaptation with κd) less with the parameter λ, as expected we generally use less vertices and get a larger error for coarse Eulerian resolutions (figs. 4.15, D.5 and D.6). For finer Eulerian resolutions (κh), the refinement diffusion should have less influence, which is already for zone 3. We chose arbitrarily $\lambda = 1/2$.

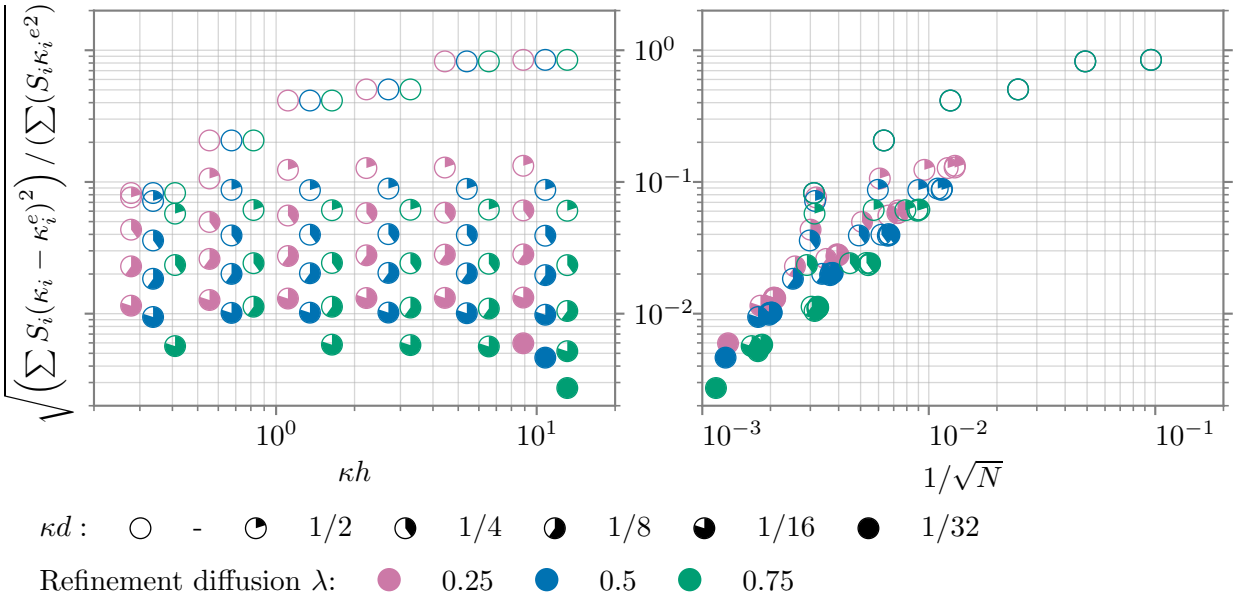


Figure 4.15: Exact position, whole domain, influence of refinement diffusion λ, l_∞ norm in fig. D.4 (on page 122)

4.2.8.3 Influence of weighting in least-squares (fig. 4.16, on page 57)

We study the influence of the weighting for the surface reconstruction when computing the curvature (section 3.3, on page 24). The weighting proves useful (fig. 4.16), especially in l_∞ norm (fig. D.7, on page 125) when curvature

adaptation is activated ($\kappa d \neq 0$), which means the mesh is more heterogenous. The weighting with one of Wendland's Radial Basis Functions (RBFs) ($\Psi_{3,1} = (1-r)_+^4(r+1)$), where $(1-r)_+ = \max(0, 1-r)$ Wendland (1995) and $r = \|\mathbf{x}_{P_i} - \mathbf{x}_P\|/R$, with R a reference length for the stencil, performs better than eq. 3.4 for the evaluation of the curvature with exact positions for the whole domain, but sometimes it is worse for zone 3 (fig. D.8, on page 126). We define R_{max} and R_{mean} the maximal and average distance of the vertices of the stencil $\mathcal{S}$ to the origin:

$$R_{max} = \max_{Q \in \mathcal{N}_2(P)} \|\mathbf{x}_Q - \mathbf{x}_P\| \tag{4.25}$$

$$R_{mean} = \frac{1}{\operatorname{card} \mathcal{S}} \sum_{x_Q \in \mathcal{N}_2(x_P)} \|\mathbf{x}_Q - \mathbf{x}_P\| \tag{4.26}$$

For the second neighborhood, $\operatorname{card} \mathcal{S} = \operatorname{card} \mathcal{N}_2 + 1$ ($\mathcal{N}_2$ does not include the origin of the stencil). The weight is null for the farthest point with $R = R_{max}$. The radius should be large enough to include enough points to determine the six coefficients of the polynomial (eq. 3.2, on page 25), but it is also expected that with larger radius of normalization, the accuracy is reduced. A comparison with different reference lengths is given in figs. D.10 and D.11, on pages 128 and 129, $R = 3R_{mean}/2$ improves the results slightly. Larger radii such as $R = 2R_{mean}$ or $R = 2R_{max}$ perform worse. From now on, the computation of the curvature with $\mathcal{N}_2$ is performed with RBF weighting with $R = 3R_{mean}/2$.

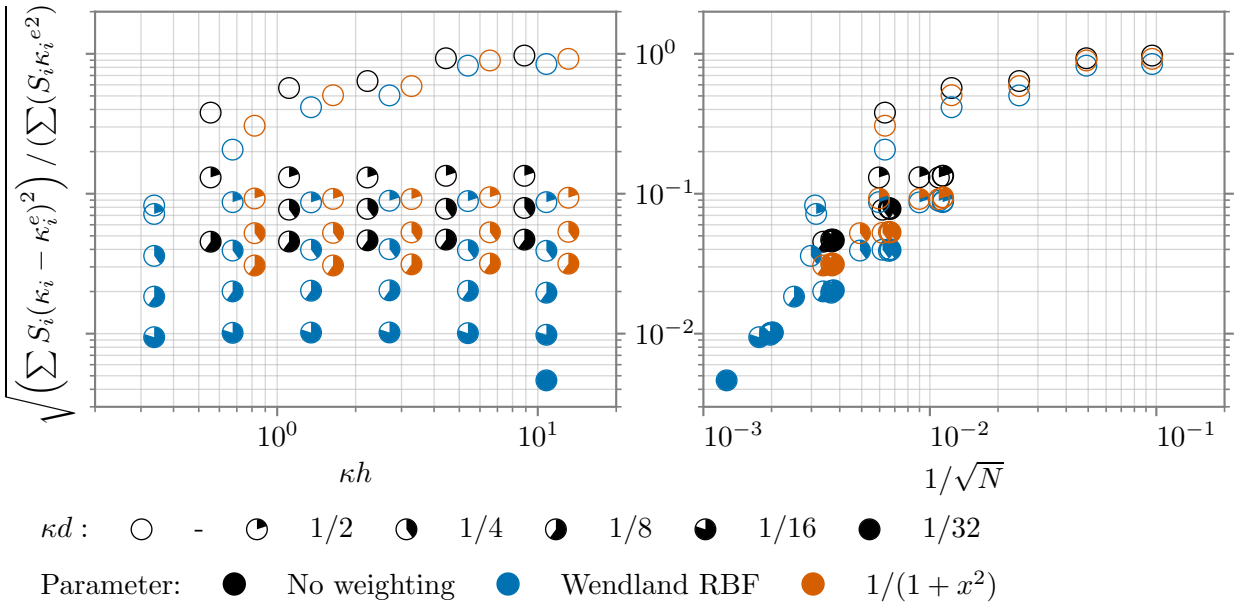


Figure 4.16: Exact position, whole domain, influence of weighting, Wendland's RBF (Wendland, 1995)$R = 3R_{mean}/2$, l_∞ norm in fig. D.7 (on page 125)

4.2.8.4 Summary

Up until now, adapting the mesh to the curvature with κd reduces the curvature error when compared to a homogeneous mesh with exact positions, for coarse Eulerian resolutions (κh). The curvature error can generally be reduced by diffusing more the curvature criterion and using a weighting with the least-squares. From now on, we use RBF weighting with $R = 3R_{mean}/2$ for the computation of the curvature with $\mathcal{N}_2$, as well as a refinement diffusion $\lambda = 0.5$.

Now, we will study the computation of curvature with an exact transport.

4.2.9 Exact transport

With the exact transport (eq. 4.22), the vertices no longer lie on the exact surface. With this transport method, remeshing errors accumulate and have an effect on the position and curvature errors. As shown in fig. 4.12, we may define at least two reference positions on the exact surface for a vertex. The reference position along $\mathbf{e}_r$ is referred to as $x^{e,1}$, the distance is called 'radial distance'. The numerical minimal distance is referred to as $x^{e,2}$, the distance is called 'minimal distance'. The curvature is computed with $\mathcal{N}_2$ with RBF weighting with $R = 3R_{mean}/2$. When the transport is not exact, the remeshing operations accumulate errors and may be amplified by the transport. In this section, we will compare the edge middle, WALF and edge fit (section 3.6, on page 29). We found that the edge fit performed better without weighting and WALF performed better with RBF weighting with $R = 3R_{mean}/2$ as shown later in subsection 4.2.9.2. From now on, edge fit refers to edge fit without weighting and WALF to WALF with RBF weighting.

4.2.9.1 Influence of the definition of the reference position (fig. 4.17, on page 59)

We compare the errors with the reference position along the radius $x_i^{e,1}$ and at the minimal distance $x_i^{e,2}$ for the edge fit without weighting (Gorges et al., 2022), depicted in fig. 4.17. There is little difference in the curvature error in l_2 and l_∞ norms fig. D.12 (on page 130). The position error is quite different in l_2 and l_∞ norms, except when $\kappa d = 0$. There is little difference in position errors between the two reference positions zone 3 (fig. D.13). The position error is lower with the minimimal distance in l_2 and l_∞ norm, especially for zone 2 (fig. D.14), where points with small position errors ($< 10^{-14}$) were not presented ($\kappa h \approx 10, \kappa d = 0$). This may be due to the fact that few vertices belong to zone 2 for low resolutions and in some cases there is no remeshing so the vertices are only transported.

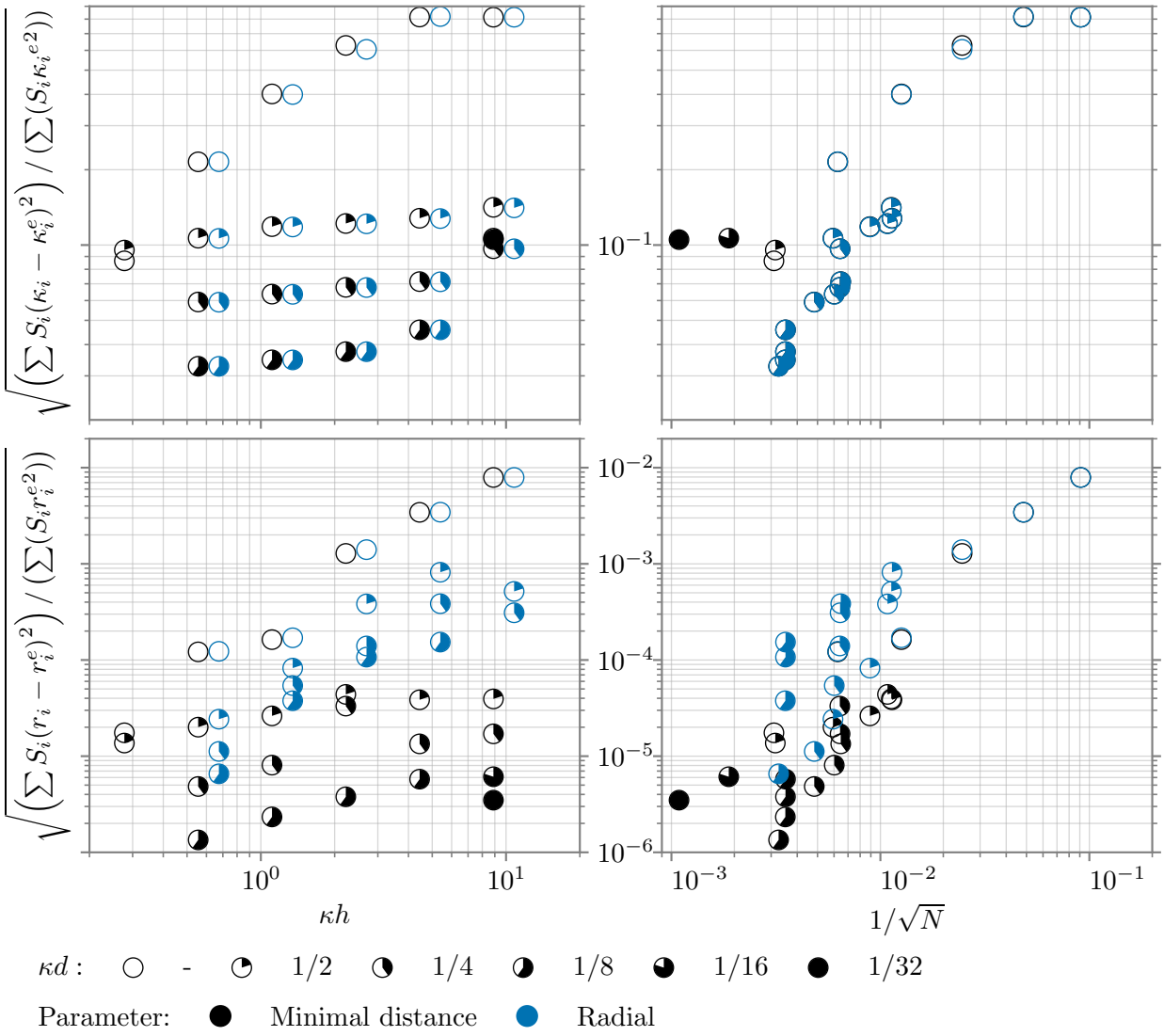


Figure 4.17: Exact transport, whole domain, edge fit (Gorges et al., 2022), influence of the definition of the reference position, l_∞ norm in fig. D.12 (on page 130)

From now on, the reference position will be defined with the minimal distance if not mentioned otherwise. We denote the reference values for the i-th vertex $r_i^e = r_i^{e,2}$ and $\kappa_i^e = \kappa_i^{e,2}$.

4.2.9.2 Influence of weighting for the new vertex position (fig. 4.18, on page 60)

We have seen previously that weighting the coordinates of the vertices may improve the evaluation of curvature with surface reconstruction when the positions are exact. We will now attempt to improve the new vertex position by using weighting in surface reconstruction (section 3.6, on page 29). WALF with the exact transport works generally better with RBF weighting (fig. 4.18), for the curvature and position error. In l_∞ norm, $\kappa d = 1/2$ with RBF weighting still improves the evaluation of the curvature while it is not the case for the other two alternatives (fig. D.15, on page 133). Edge fit with exact transport works better without weighting for the position error in l_2 norm (fig. D.16). This may be explained by the fact that the stencil of the edge fit method includes only eight points while there are six coefficients to be determined for the surface reconstruction and for surface reconstruction, the number and distribution of points is important. The computations crash with RBF weighting and edge fit in some cases, when position errors accumulate and provoke self-intersections during the transport.

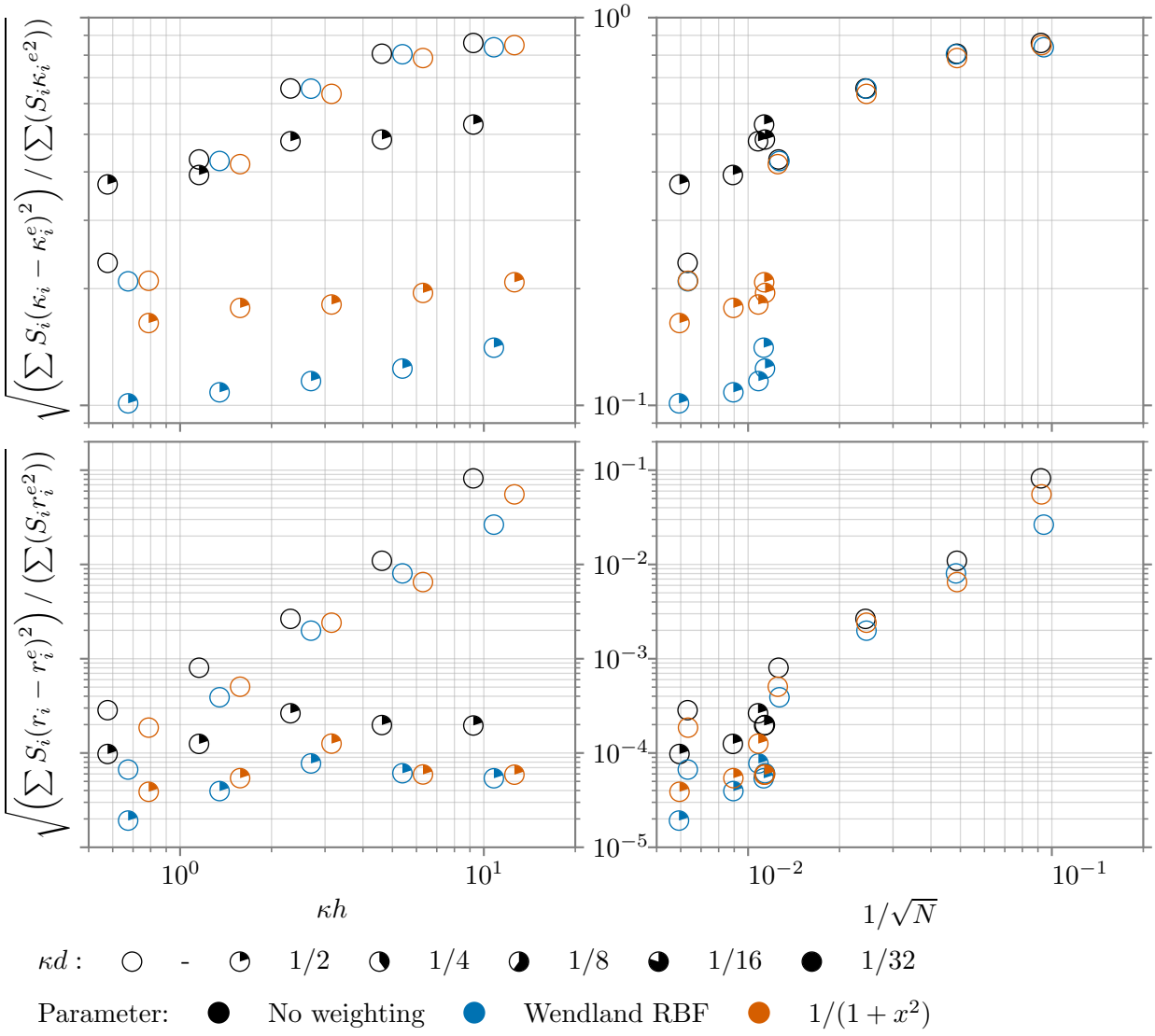


Figure 4.18: Exact transport, whole domain, reference position with minimal distance, WALF (Jiao and Wang, 2012), influence of weighting for the new vertex position, l_∞ norm in fig. D.15 (on page 133). Only $\kappa d = 0$ and $\kappa d = 1/2$ are presented.

4.2.9.3 Comparison of edge refinement methods with minimal distance (fig. 4.19, on page 61)

We compare the three edge refinement methods (section 3.6): the edge middle, WALF (Jiao and Wang, 2012) and edge fit Gorges et al. (2022). This comparison of the three edge refinement methods with minimal distance is also presented with the radial reference position in figs. D.17 to D.20 (on pages 135, 136, 137 and 138). The edge middle generally gives the worst results (fig. 4.19), aside from curvature error in l_2 norm when $\kappa d = 0$: the results are similar for the three edge refinement methods. The position error is always higher with the edge middle than with surface reconstruction (WALF and edge fit). Using curvature refinement with the edge middle may improve the curvature and position errors in some cases but the results are not improved much or worsened from $\kappa d \approx 1.35$. For coarse meshes ($\kappa d \approx 10.8$), using curvature refinement may lead to self-intersections such as with $\kappa d = 1/2$. For the same resolution, surface reconstruction does not produce self-intersections. Therefore, me may conclude that the curvature refinement alone is not a guarantee of lower error levels. As we saw earlier with the expanding and shrinking sphere (section 4.1, on page 37), the edge middle tends to saturate for the curvature error before methods based on surface reconstruction do.
Surface reconstruction without adaptation to the curvature ($\kappa d = 0$) reduces the position error compared to the

edge middle, but not the curvature error generally. Combining surface reconstruction with a curvature-based remeshing criterion, here κd, may help reduce error levels with the exact transport.

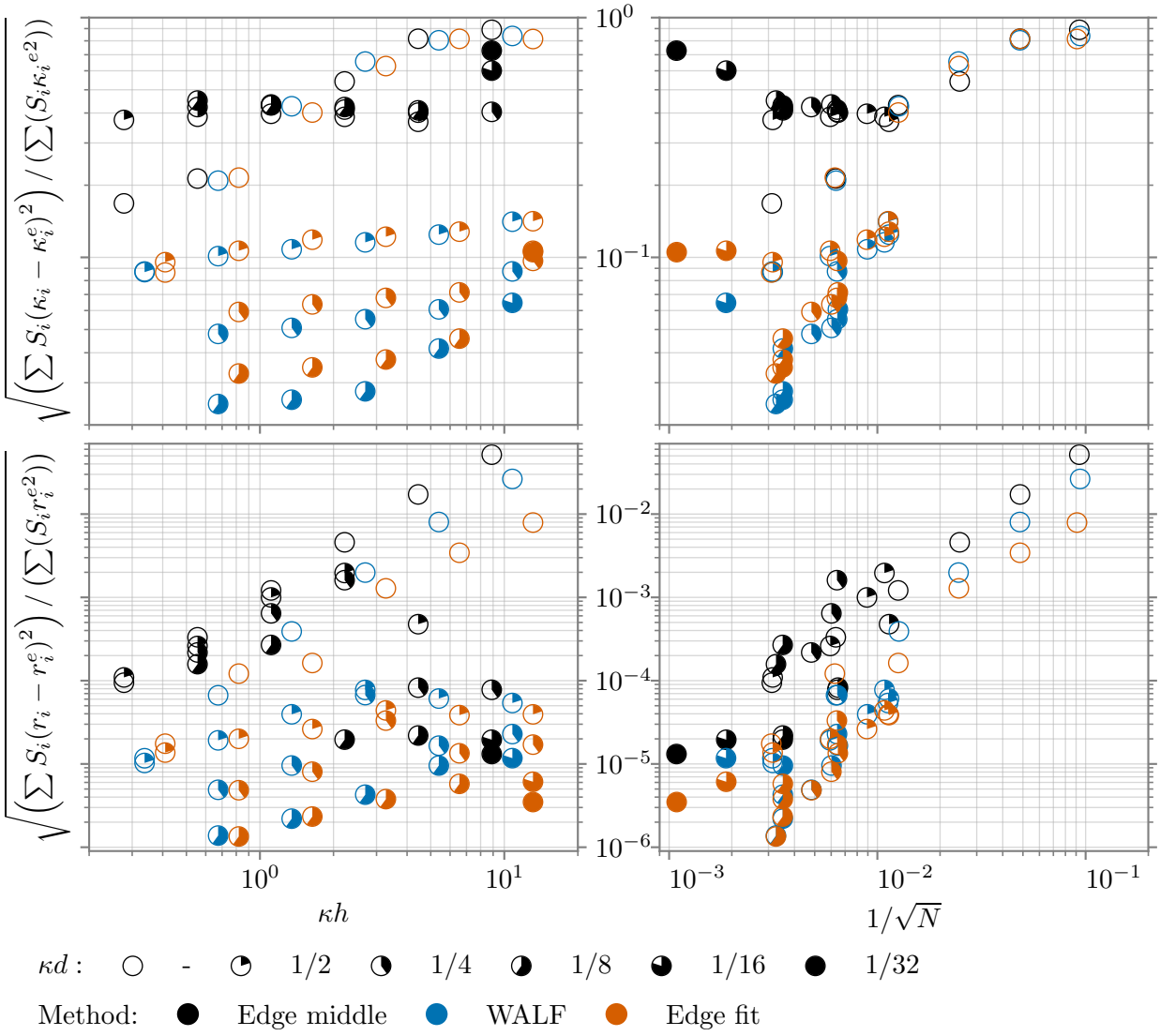


Figure 4.19: Exact transport, whole domain, reference position with minimal distance, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), l_∞ norm in fig. D.21 (on page 139)

Lowering κd combined with surface reconstruction improves the curvature and position error in l_2 norm, at the cost of increasing the number of vertices at the end of the simulation denoted N, except when the Eulerian resolution is fine enough (the criteria $\kappa d = 1/2^i$ should be satisfied, and we should have $\epsilon_- \approx \epsilon_{1/2^i}$, as we can see at $\kappa h \approx 0.3$ for the curvature and position error). At some point, for a fine enough Eulerian resolution, the criterion κd does not influence the mesh anymore, it proves useful only for coarse Eulerian grids. In l_∞ norm (fig. D.21, on page 139), with increasing κd, the improvement on curvature error is small and the improvement on position error is not guaranteed, sometimes it is worse.
The edge fit method is mostly better than WALF for the position error, aside from ($\kappa h \approx 0.7, \kappa d = 0$), but the difference is less noticeable for the curvature error. From $\kappa d \leq 1/4$, WALF seems to perform better for curvature error. The curvature error is lower in l_∞ norm with WALF when curvature refinement is activated, but one must keep in mind that the cost is not always equivalent when comparing two simulations with the same Eulerian resolution.

Figures D.22 and D.23 present the results for zones 3 and 2. With $\kappa d = 0$, the curvature error with surface reconstruction is lower than with the edge middle in zone 3 (from $\kappa h \approx 1.34$ in l_2 norm and at $\kappa h \approx 0.67$ in l_∞ norm) but not in zone 2 aside from $\kappa h \approx 0.67$ in l_∞ norm. We may compare the results of zone 3 when $\kappa d = 0$ which is nearly spherical with the results of the expanding sphere (section 4.1, on page 37). There seems to be a saturation for curvature error when placing the new vertex at the middle of the edge, which happens earlier (at higher κh) than with surface reconstruction. This saturation of the curvature error does not appear in zone 2 at $\kappa h \approx 0.3$. Since zone 2 presents larger errors than zone 3, there is no improvement of curvature error for surface reconstruction without curvature adaptation in the whole domain. Edge fit with the exact transport generally gives lower position errors than WALF and edge middle in zone 3. The low position errors in zone 2 for the coarsest Eulerian resolution with surface reconstruction were not represented. They may be explained by the small number of vertices: these vertices may have not been modified by remeshing operations and only transported with exact increments of position.

When surface reconstruction is used (WALF or edge fit), ($\kappa h \approx 10.8, \kappa d = 1/2$) is interesting compared to ($\kappa h \approx 0.3$ i.e. nx=128, $\kappa d = 0$) without surface reconstruction for the curvature error. There are less vertices for a similar curvature error in l_2 norm. $\kappa d = 1/2$ improves the position and curvature errors in l_2 norm yet the computational cost indicated by the number of vertices at the end of the simulation is great. In l_∞ norm, using more segments per radius of curvature does not always decrease the curvature or position error.

4.2.9.4 Summary

WALF can be improved with RBF weighting, however edge fit often gives better results with exact curvatures.

4.2.9.5 With numerical curvatures for the remeshing criteria (fig. 4.20, on page 63)

Previously, the curvature used in the remeshing criterion was computed at the closest point (subsection 4.2.6.3, on page 52) on the exact surface with the analytical formula (eq. 4.17). From now on, the curvature is computed numerically, with a $\mathcal{N}_2$ neighborhood with RBF weighting (fig. 4.20). When using the numerical curvature instead of analytical curvatures for the remeshing criterion (fig. 4.19), the results may be worse. While some simulations still present an improvement in position and curvature errors in l_2 norm when using surface reconstruction with curvature adaptation (κd) compared to a simulation without curvature adaptation, other simulations present larger errors in curvature and position such as $\kappa h \approx 10.8$ and $\kappa h \approx 5.4$ (especially $\kappa h \approx 5.4$, stemming from zone 2 (fig. D.26)). In l_∞ norm, the curvature is mostly worsened by curvature adaptation, even with surface reconstruction, while position error is still reduced, aside from coarse meshes with WALF (fig. D.24, on page 142).

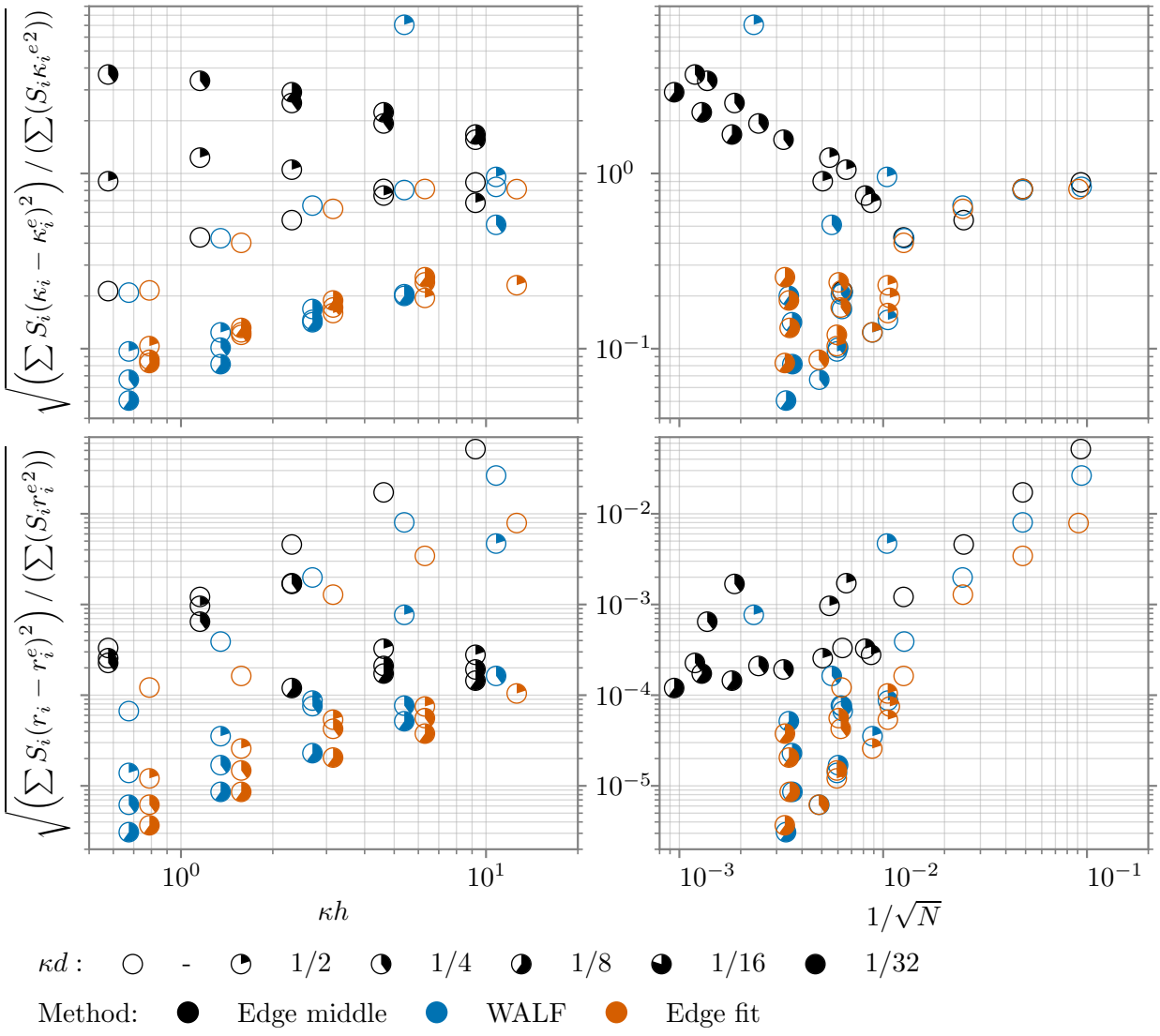


Figure 4.20: Exact transport, whole domain, reference position with minimal distance, adaptation to numerical curvature, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), l_∞ norm in fig. D.24 (on page 142)

4.2.9.6 Influence of the time-step with numerical curvatures for the remeshing criteria

We observed that adapting the mesh to numerical curvature can worsen the error in curvature. In some cases, adapting the time-step with a reference length based on the smallest edge length d instead of the Eulerian mesh spacing h (subsection 4.2.2.2, on page 49) may alleviate this issue as depicted in fig. 4.21 for the WALF method. In other cases, making more iterations may be detrimental by accumulating more errors.

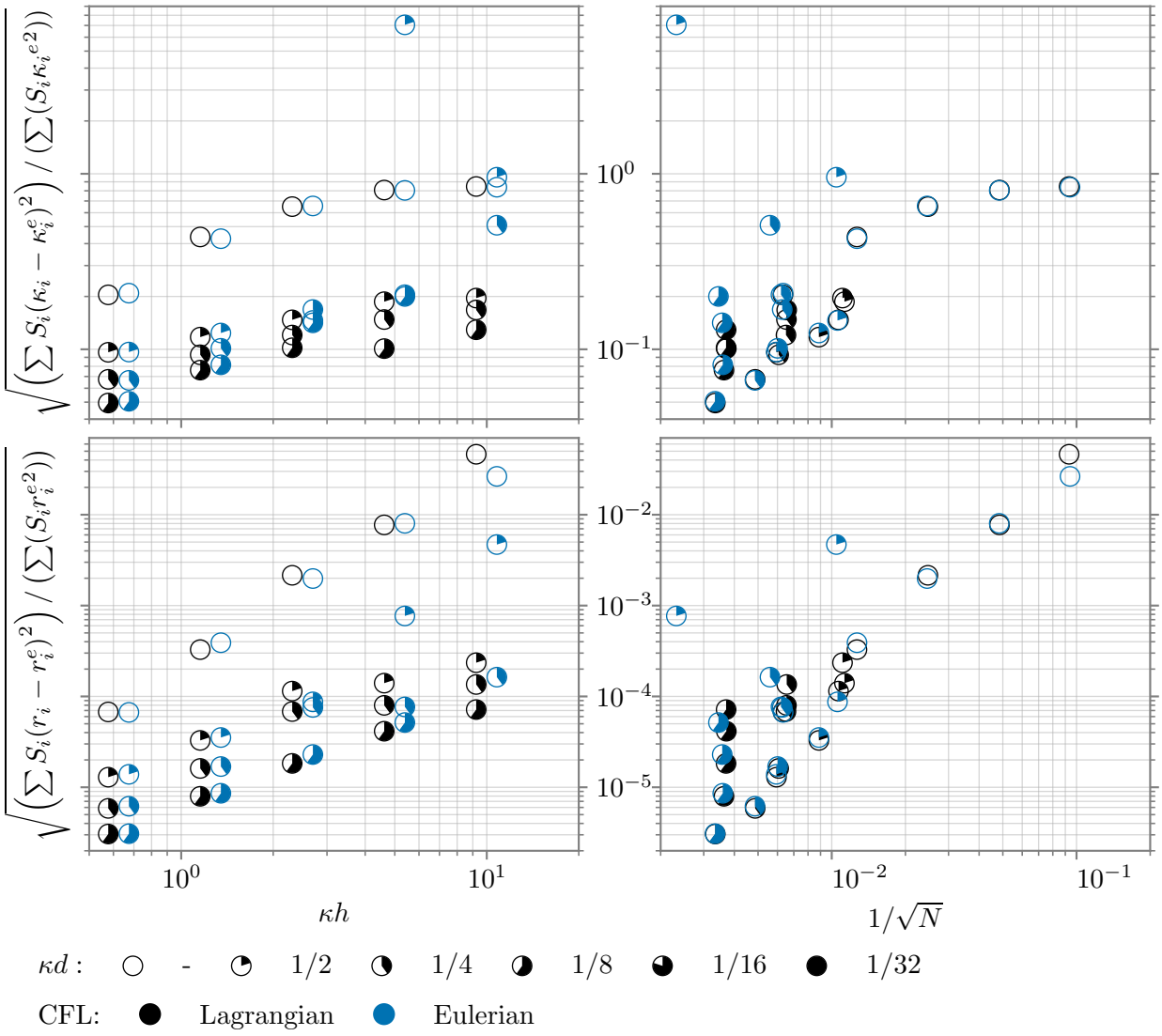


Figure 4.21: Exact transport, whole domain, reference position with minimal distance, adaptation to numerical curvature edge middle, Lagrangian and Eulerian CFL, WALF (Jiao and Wang, 2012), l_∞ norm in fig. D.27 (on page 145)

4.2.9.7 Summary

The results are worse with numerical curvatures. Using a Lagrangian CFL does not necessarily improve the curvature and position errors.

4.2.10 Interpolation

Now we use spatial interpolation instead of the exact transport. The computations with PERM interpolation need a more strict CFL condition if curvature-based remeshing is used. With CFL $= 0.5$, based on the Eulerian grid, the computations with curvature-based mesh refinement crash (fig. 4.22): the precision of the transport with interpolation is not enough with curvature adaptation and the remeshing procedure refines the mesh at 'unphysical features' generated by the interpolation scheme, which produces many small triangles and mesh intersections eventually, at which point the simulation is stopped. We may then need to use a CFL based on the Lagrangian mesh with the edge fit method which seems to be the most precise edge refinement method out of the three methods we tested.

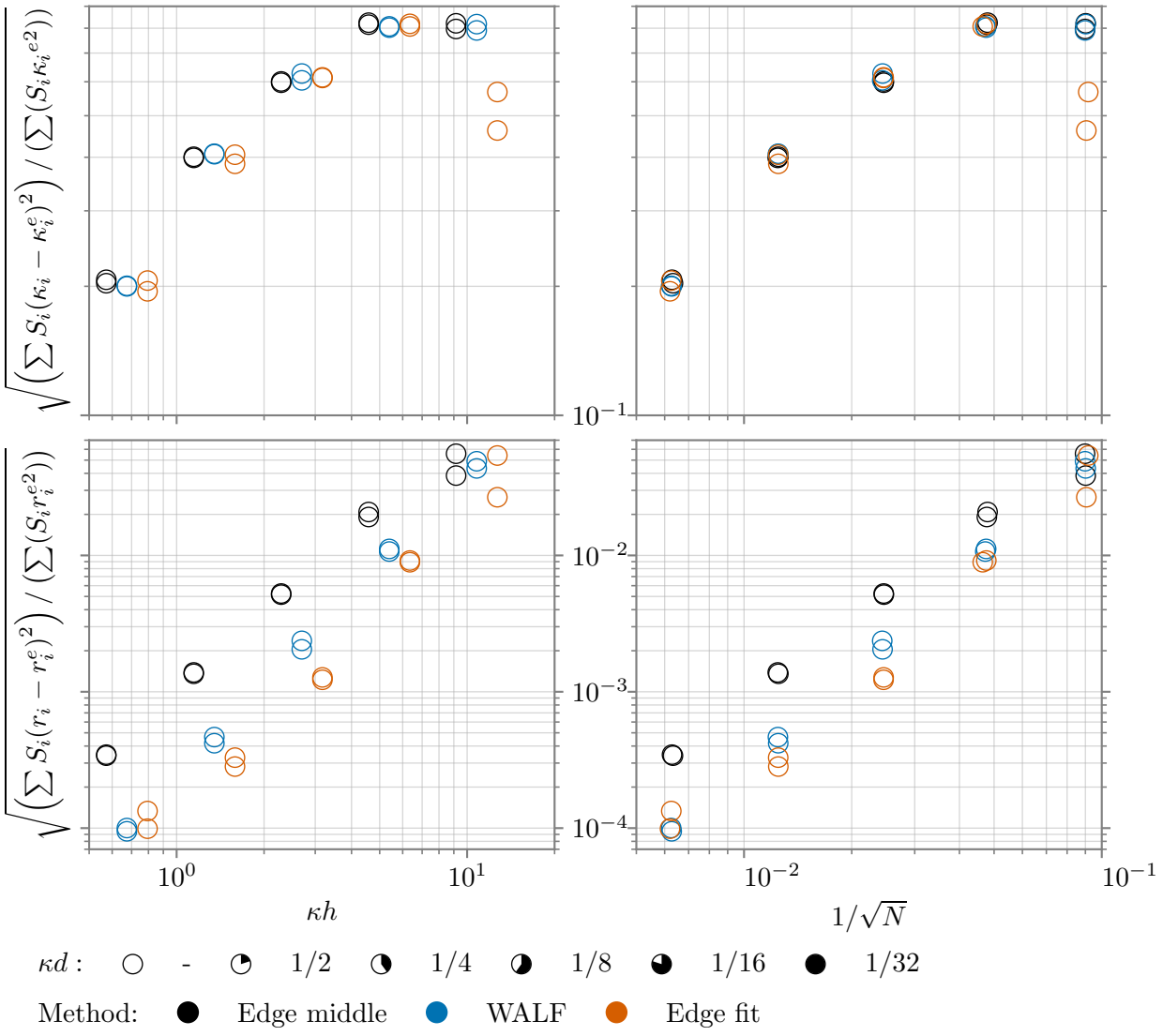


Figure 4.22: Interpolation with Parabolic Edge Reconstruction Method (PERM), whole domain, reference position with minimal distance, adaptation to numerical curvature, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30)

Even with a Lagrangian CFL, there is no guarantee the simulation will not crash with PERM. Only the simulations with $\kappa d = 0$ ran without intersections (fig. 4.23). Because of the interpolation, the position error is larger than with exact transport and the curvature is no longer exact and it serves as a criterion for the remeshing. Peskin and Q1 interpolations gave better results with curvature adaptation, as depicted in fig. 4.25. For the presented results with $\kappa d = 0$, there is no intersection, contrary to the simulations with PERM. Adapting the mesh to the curvature with $\kappa d = 1/2$ reduces the errors for position and curvature on finer meshes in l_2 norm.

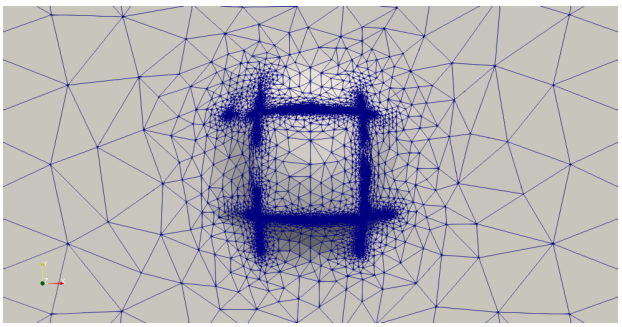


Figure 4.23: Interpolation, example of crash due to mesh intersections, adaptation to numerical curvature ($\kappa d = 1/2$), $nx = 32$,Lagrangian CFL, edge fit (Gorges et al., 2022)

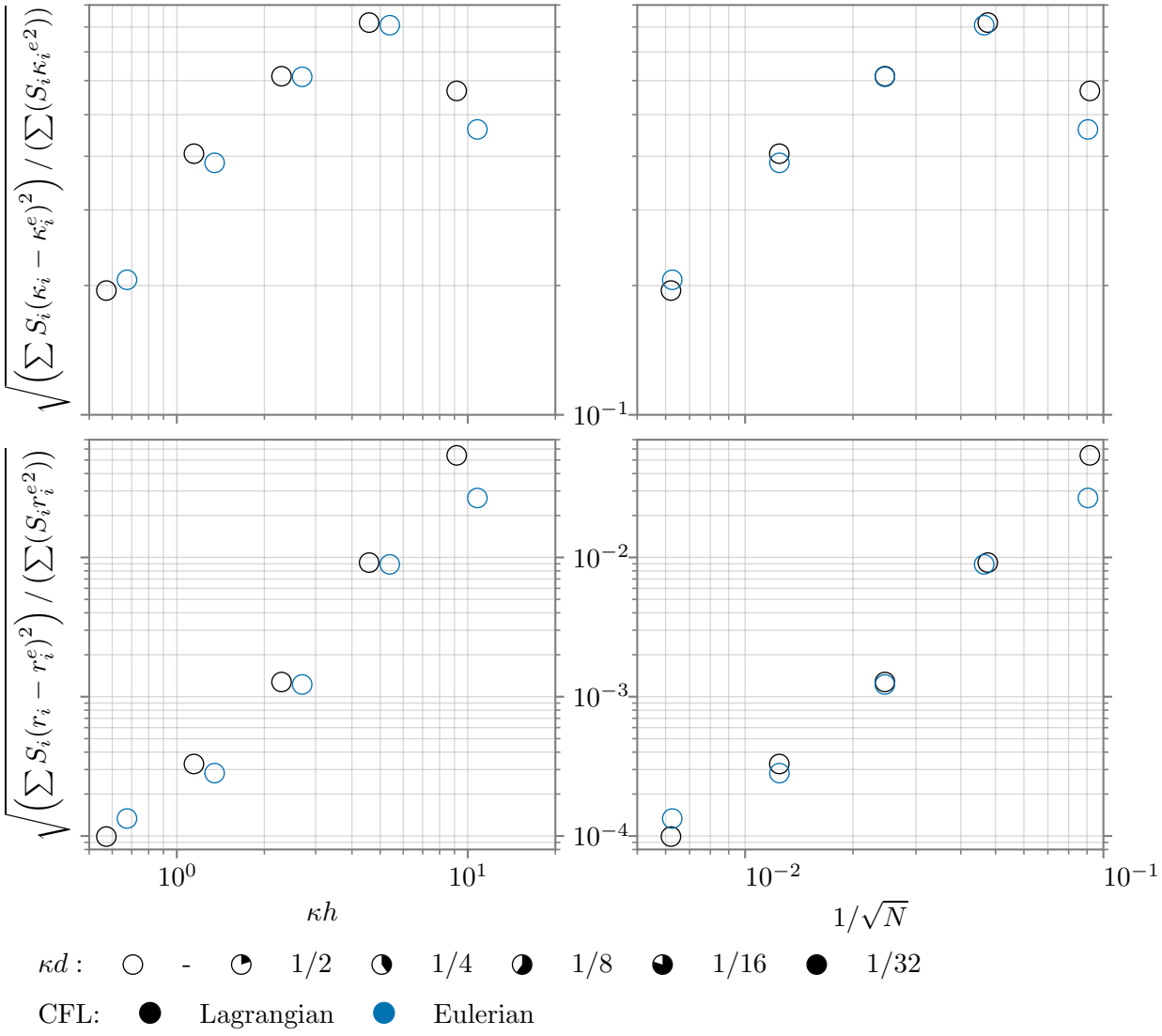


Figure 4.24: Interpolation with PERM, Lagrangian CFL, whole domain, reference position with minimal distance, adaptation to numerical curvature edge middle, edge fit (Gorges et al., 2022)

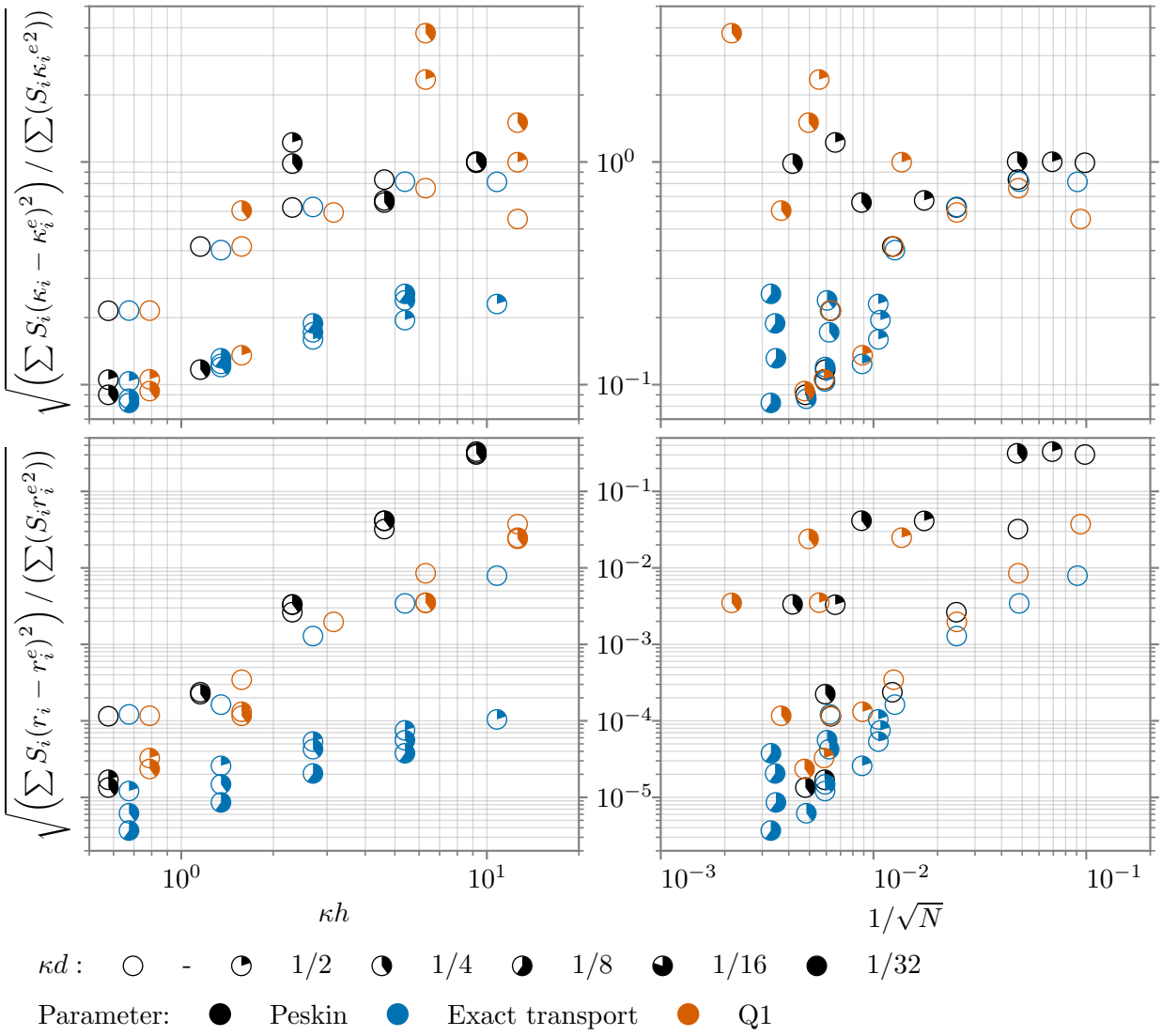


Figure 4.25: Interpolation with Peskin and Q1 compared to exact transport, whole domain, reference position with minimal distance, adaptation to numerical curvature, edge fit (Gorges et al., 2022)

4.3 Conclusion

First, we saw that when the vertices lie on the exact surface, adapting the mesh to the curvature with κd improves the evaluation of the curvature. The results may further be improved by considering the weighting of the coordinates for the least-squares and the diffusion of the curvature in the remeshing criterion κd. We now evaluate the curvature with RBF weighting with $R = 3R_{mean}/2$ since it gave better results with the exact position.
With the exact position or transport, $\kappa d = 1/2$ generally improves the results for position and curvature error but it is not always worth using more segments per radius of curvature. Adapting the mesh to the curvature entails important costs regarding the number of vertices. The results are not regular once the position is not exact, even in l_2 norm: adapting the mesh with more than $\kappa d = 1/2$ does not guarantee a more precise evaluation of the curvature and position. Several reasons may explain the differences: accumulated errors in position (due to remeshing operations and transport), valence, distribution of the vertices on the surface and their location may be more difficult to represent (the errors are not computed at the same positions).
When the transport is not exact, the edge refinement method plays an important role, and adapting the mesh to the curvature no longer guarantees a reduction of the position and curvature errors. Adapting the time-step to the size of the smallest edge may limit errors but it is not guaranteed. As we saw earlier with the expanding

and shrinking sphere (section 4.1, on page 37), the edge middle tends to saturate for the curvature error before methods based on surface reconstruction do. The edge fit gave better results than WALF in most cases.
The aim of adaptive remeshing is to use less vertices to represent the features of the interface, yet it requires a precise computation of the curvatures, a precise transport and a precise edge refinement method, otherwise it may be counterproductive. In our case, the PERM interpolation here applied on regular grids could not be used with adaptive remeshing. The velocity field may not be regular enough for the PERM interpolation, and finer grids may be necessary to be able to adapt the mesh to the curvature. We have seen that Peskin and Q1 are less likely to produce intersections, this may be explained by their smaller stencil than the PERM interpolation. It might be interesting to use other interpolation schemes and AMR for the Eulerian grid.

Remarks. Because of the diffusion, the number of segments per radius is not always respected: $\tilde{\kappa_i} > \kappa_i^e$. This means that more vertices are used in zone 3 because of the transition in refinement. In this study, we did not employ mesh smoothing. With the exact position, smoothing might improve the distribution of the markers, improving in turn the results sometimes. When the position is not exact, the smoothing introduces position errors, which is detrimental to the accuracy. The evaluation of the curvature may lead to unnecessary mesh operations, downgrading the accuracy of the discretization of the surface. As mentioned by Popinet (2018), it might be interesting to consider *spectral mesh processing* (Lévy and Zhang, 2010) to filter the curvature. The basic approach would require to define an operator on the mesh and compute the eigenvectors efficiently.

Now that we have studied the remeshing operations without Navier-Stokes equations nor topological changes, we perform in the next part simulations of elementary two-phase flows, coalescence and breakup.

Part III

Elementary two-phase flows, coalescence and breakup

5 Rising bubble

In which we investigate the influence of the numerical methods on the dynamics of rising bubbles in viscous liquids, in different regimes with qualitative and quantitative comparisons. The implemented Front-Tracking method is compared with a VOF method.

5.1 An overview on rising bubbles

In chapter 4, we have studied the remeshing procedures on analytical test cases, without solving the Navier-Stokes equations. Now, we study the whole Front-Tracking method with the resolution of the Navier-Stokes equations. The simulation of rising bubbles is a common test case and it is often validated with analytical solutions or experiments. Quantitative benchmarks have been proposed for two-dimensional rising bubbles (Hysing et al., 2009). The terminal bubble rise velocity is often employed for the validation with experiments (Sussman et al., 2007). Geometric properties are also used: bubble mass center, bubble tip height (Sharaf et al., 2017), bubble extensions (or diameters) in Cartesian directions or sphericity (a measure of how similar a shape is to a sphere).

Bhaga and Weber (1981) conducted experiments on rising bubbles in a liquid column, where the concentration of sugar in water is varied to investigate different regimes of bubble rise. Hua and Lou (2007) performed axisymmetric simulations with a Front-Tracking method to reproduce such experiments. Some cases could not be reproduced without three-dimensional computations: the bubbles are no longer symmetric in the wobbling regime. Three-dimensional simulations are performed with a Front-Tracking method and AMR in Hua et al. (2008). For the qualitative comparison against the experiments, it is difficult to distinguish the shape of the bubble. The size of the computational domain influences the rise of the bubble. They use domains of width $8D$, with D the diameter of the bubble, based on a study of sensitivity and note that an experiment by Krishna et al. (1999) concludes the same. Pianet et al. (2010) studied several averaging schemes for density and viscosity for axisymmetric simulations of rising bubbles. Pivello et al. (2014) performs three-dimensional rising bubble simulations in the wobbling regime with a Front-Tracking method and AMR. Anjos et al. (2014) used an ALE FEM to simulate three different cases of rising bubbles from Bhaga and Weber (1981). Baltussen et al. (2014) compared the height functions with the Integral Formulation for different regimes of rising bubbles, and provided guidelines for the choice of the surface tension formulation, depending on the Eötvös number (Eo) and Morton number (Mo) numbers (eqs. 5.2 and 5.3, on page 72). Liu et al. (2023); Gorges et al. (2023) simulated cases from Bhaga and Weber (1981) with a Front-Tracking method.

5.2 Comparison to an experiment

Tripathi et al. (2015) simulated the rise of an air bubble for different Ga and Eo numbers and thus identified five regimes: axisymmetric, skirted, zigzagging or spiralling, peripheral breakup, and central breakup. Sharaf et al. (2017) complemented this work with an experimental and numerical investigation in the same ranges of Ga and Eo. Both works used the solver Gerris (Popinet, 2003) with a domain of $15D \times 15D \times 60D$ and $15D \times 15D \times 15D$ respectively. We focus on four regimes of bubble rise: spherical, oblate, dimpled and skirted. The physical properties used in our simulations are presented in table 5.1. The bubbles may be described by the subsequent non-dimensional numbers, where the radius R is the reference length, ρ_2 and μ_2 are the fluid

properties of the liquid:
Morton number:

$$\mathrm{Mo} = \frac{g\mu_2^4}{\rho_2\sigma^3} \tag{5.1}$$

Galilei number:

$$\mathrm{Ga} = \frac{\rho_2\sqrt{gR^3}}{\mu_2} \tag{5.2}$$

Eötvös number:

$$\mathrm{Eo} = \frac{\rho_2 g R^2}{\sigma} \tag{5.3}$$

μ_1	ρ_1	μ_2	ρ_2	σ	R	Mo	Ga	Eo
1×10^{-5}	1	3.197×10^{-1}	1235	6.340×10^{-2}	7.800×10^{-3}	3.3×10^{-1}	8.3	1.2×10^{1}
1×10^{-5}	1	9.678×10^{-1}	1254	6.240×10^{-2}	1.927×10^{-2}	2.8×10^{1}	1.1×10^{1}	7.3×10^{1}
1×10^{-5}	1	1.700×10^{-1}	1222	6.420×10^{-2}	5.200×10^{-3}	2.5×10^{-2}	8.4	5.0
1×10^{-5}	1	9.600×10^{-3}	1100	6.840×10^{-2}	8.300×10^{-4}	2.4×10^{-7}	8.6	1.1×10^{-1}

Table 5.1: Fluid properties for the dimpled, skirted, oblate and spherical bubbles studied in Sharaf et al. (2017)

We compare the influence of some parameters related to the Navier-Stokes equations to improve the results independently from the tracking of the interface. The numerical and experimental results from Sharaf et al. (2017) as well as computations with an implementation of Weymouth and Yue (2010)'s VOF method are compared to the Front-tracking method. Then, the influence of the Lagrangian mesh is studied.

Sources of differences between the experiments and simulations. The sources of differences between the simulations and experiments are for example: pollution with surfactants (the liquid is replaced in Sharaf et al. (2017)), volume of the bubble (initial volume, and the increase of volume as the bubble rises in the experiment), initial (involuntary) movement, initial position, initial shape (not perfectly spherical). Sharaf et al. (2017) supposed they could not reproduce the central breakup regime because of the initial shape of the bubble. There are also measuring errors for the following geometrical and fluid properties: bubble tip height, volume, viscosity, density, surface tension coefficient.

Presentation of the results. The height of the tip of the bubble z_{tip} is non-dimensionalized by the radius R, and the time is non-dimensionalized by $\sqrt{g/R}$. We plot the slice of the Lagrangian mesh in the $\mathbf{X}$ plane, as well as the 0.5 isocontour of the volume fraction for a comparison with the VOF method. The bubble tip height from the simulations is compared with the bubble tip height from the experiment z_{tip}^{exp}. For the cases of a spherical, oblate and dimpled bubbles with a Morton of respectively 2.4×10^{-7}, 2.5×10^{-2} and 3.3×10^{-1}, experimental errors are provided by Sharaf et al. (2017) and we illustrate the error bars. For quantitative comparisons, the non-dimensionalized bubble tip position is extracted. For the VOF simulations, z_{tip} is obtained by a linear interpolation of the volume fraction at the highest cell with a non-null volume fraction. For the Front-Tracking method, the bubble tip is simply the maximal non-dimensionalized marker height. We compute the maximal error for the experimental points between the bubble tip height z_{tip} and the experiment and the simulations denoted ϵ_p, and the maximal relative error for the velocity denoted ϵ_v. The velocity v is obtained by a least-square fit of the points at the same abscissae as the experimental points. The contours of the bubbles are obtained in the $\mathbf{X}$ plane, by slicing the Lagrangian mesh for the Front-Tracking method and by computing the 0.5 isocontour of the volume fraction for the VOF method. The 0.5 isocontour could also be extracted for the Front-Tracking method, and differences are to be expected between the two representations (as we will see later in fig. E.1). Depending of the slicing plane, the isocontour may be more or less jagged. These representations are used for qualitative comparisons only. The black horizontal lines indicate the bubble tip position from the experiment z_{tip}^{exp} and $z_{tip}^{exp} - \Delta z$ with Δz the experimental uncertainty.

Eulerian mesh. We use a $8D \times 8D \times 15D$ domain, with D the initial diameter of the bubble, with the bubble centered at $Z = 2D$. We present simulations with 8, 16 and 32 cells per bubble diameter.

Lagrangian mesh. We perform simulations with different initial meshes comprised of 80, 320, 1280, 5120 and 20480 triangles. The reference edge length is the mean initial edge length $l_m^{(0)}$, computed with the discretization of a sphere of radius R, which is provided in table 5.1. The edge length interval is: $[\lambda_{min}, \lambda_{max}]$, with $\lambda_{min} = l_m^{(0)}/2$ and $\lambda_{max} = 4l_m^{(0)}/3$. The curvature-based remeshing criterion κd is deactivated to focus on the influence of the resolution of the Lagrangian mesh. We may either initialize the mesh with exact positions or exact discrete volume. When not mentioned otherwise, the mesh is initialized with exact positions.

Resolution. The Navier-Stokes equations are solved with a threshold of 10^{-4} on the preconditioned residual of the BiCGStab(2), non-dimensionalized by the right-hand side vector. The initial time-step is denoted Δt and a convective CFL of 0.5 is employed.

Coupling of the Front-Tracking method with the Navier-Stokes equations. The velocities of the markers are interpolated with Peskin's interpolation. Peskin interpolation gives smoother interfaces than linear interpolation and PERM, the latter tends to produce self-intersections when the triangle edge lengths are smaller than the Eulerian resolution. The Momentum Preserving method (subsection 1.5.2, on page 7) is used, and the same Third-order Strong Stability-Preserving Runge-Kutta method (SSP-RK3) scheme with the velocity extrapolations is used for the time integration of the transport equations of the markers (eq. 2.13, on page 15). We use the edge fit method for the edge splitting and collapsing. We use Kuprat et al. (2001)'s smoothing as well as the Velocity-Based Correction for the volume correction (section 2.10, on page 18). No topological method is activated to speed up the computations with the Front-Tracking method. We use 8 subdivisions for the Ray-Casting method (appendix B, on page 111). When not mentioned otherwise, we use the height functions for the surface tension, in order to limit the difference between the VOF and Front-Tracking simulations.

5.2.1 Influence of the viscosity average

We compare four average schemes used for the viscosity (subsection 1.3.2, on page 5): arithmetic, harmonic, mixed and discontinuous averages in fig. 5.1 for $\mathrm{Mo} = 28$ with the VOF method. The results from Sharaf et al. (2017) are presented on the two columns on the left. A slice is represented at the middle. On the right, the following figures are presented from top to bottom: bubble tip position and error on the bubble tip position (compared to the experiment). The maximal position error and the relative error on velocity is indicated in the legend. The mixed average gives better results when comparing to the experimental terminal velocity v and the bubble tip height. There are 32 cells per diameter. The initial timestep is $\Delta = 10^{-4}$ and the timestep. There is a great sensibility of the shape to the average of the viscosity: the shapes are quite different at $t = 8$. The mixed average has a tendency to limit the stretching of the bubble. If the stretching of the bubble is prioritized, the arithmetic average seems better for 32 cells per diameter and $\mathrm{Mo} = 28$. If the tip height and velocity are favored, the averaging schemes sorted by decreasing precision are: mixed, harmonic, discontinuous and arithmetic.

A similar comparison is presented for $\mathrm{Mo} = 2.5 \times 10^{-2}$ (fig. 5.2, on page 75) with the VOF method. The mixed average gives not only lower errors for the tip height and velocity, but also a shape more similar to the experiment. The shapes given by the arithmetic and mixed average are presented in fig. 5.3 with the VOF method for 8, 16 and 32 cells per diameter. For the three meshes, the mixed average has a lower velocity error compared to the arithmetic average. For a spherical drop and 8 cells per diameter, the mixed average is also more precise (fig. 5.4, on page 77).

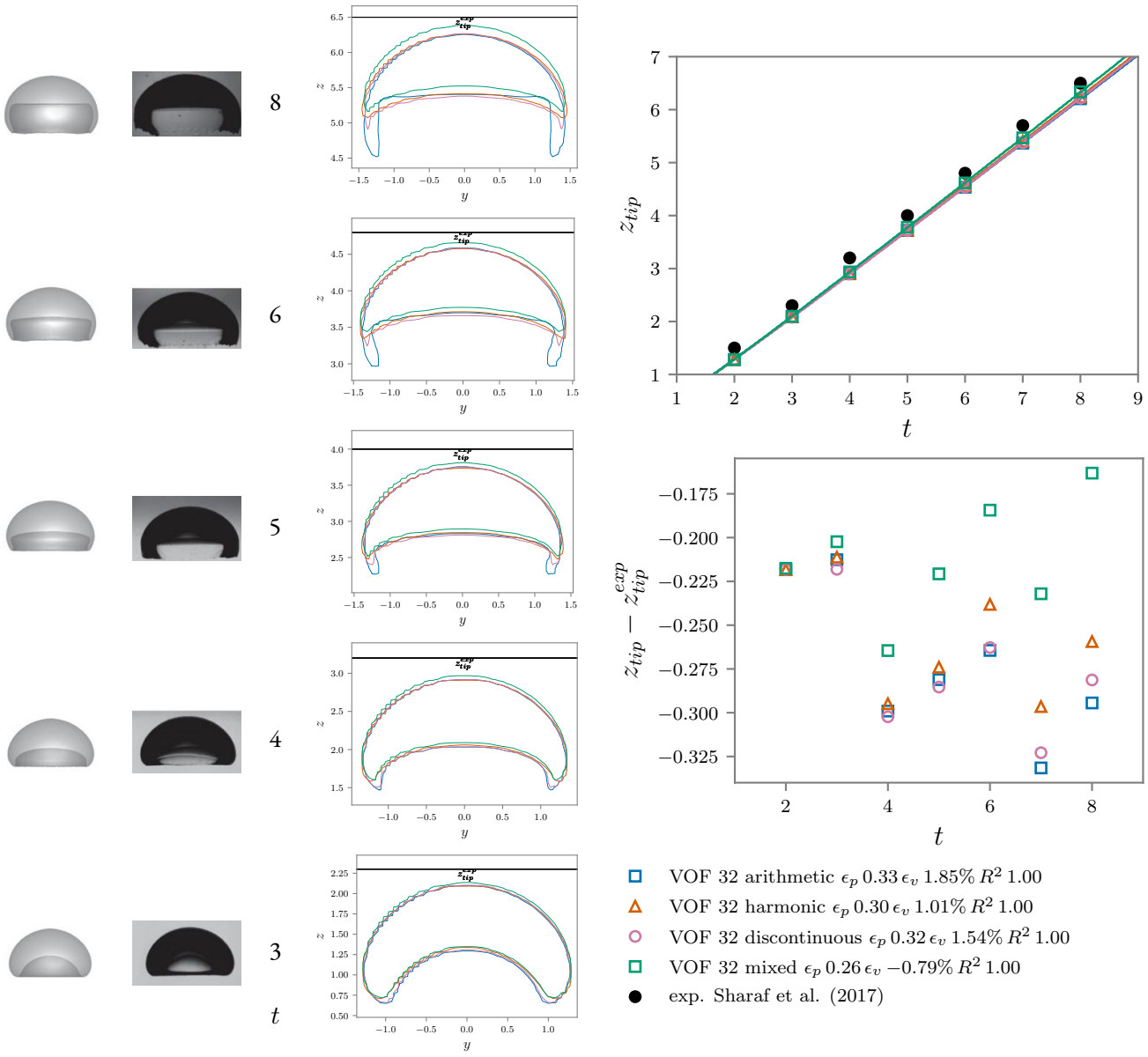


Figure 5.1: Influence of the viscosity average, Morton 28, VOF, 32 cells per diameter, $\Delta t = 10^{-4}$, maximal error for the experimental points between the bubble tip height z_{tip} the experiment and the simulations denoted ϵ_p, and the maximal relative error for the terminal velocity denoted ϵ_v, obtained by a least-square fit on the experimental points, the minimum of the correlation coefficient R^2 between the experiment and the simulation is indicated. The results from Sharaf et al. (2017) are indicated in black with their experimental errors. The first two rows are figures from a simulation and experiment from Sharaf et al. (2017). Reproduced from Sharaf et al. (2017), with the permission of AIP Publishing. The bubble tip height is non-dimensionalized by the radius R, t is the non-dimensionalized time.

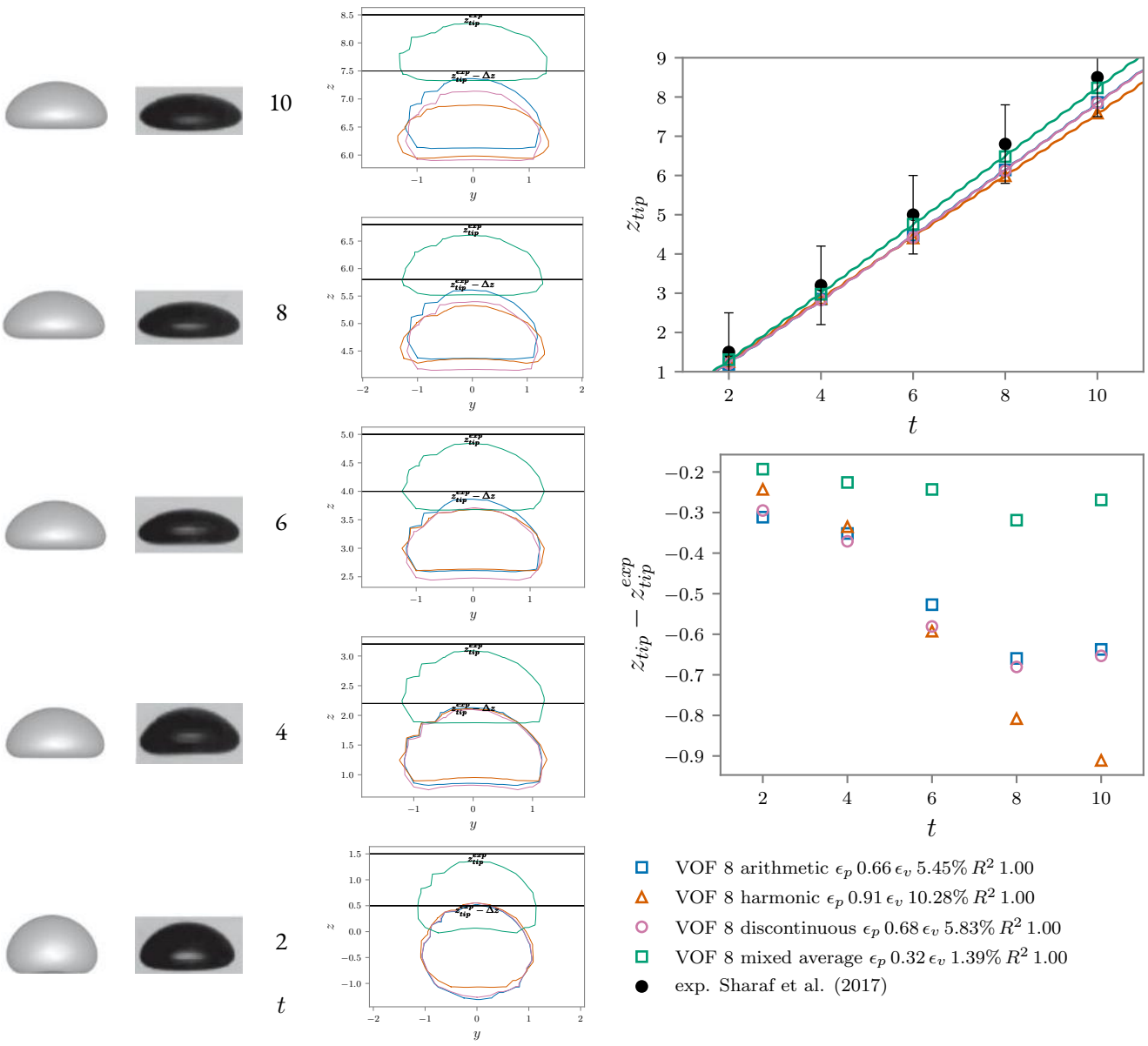


Figure 5.2: Influence of the viscosity average, Morton 2.5×10^{-2}, VOF, 8 cells per diameter, $\Delta t = 10^{-3}$. The results from Sharaf et al. (2017) are indicated in black with their experimental errors. The first two rows are figures from a simulation and experiment from Sharaf et al. (2017). Reproduced from Sharaf et al. (2017), with the permission of AIP Publishing. The bubble tip height is non-dimensionalized by the radius R, t is the non-dimensionalized time.

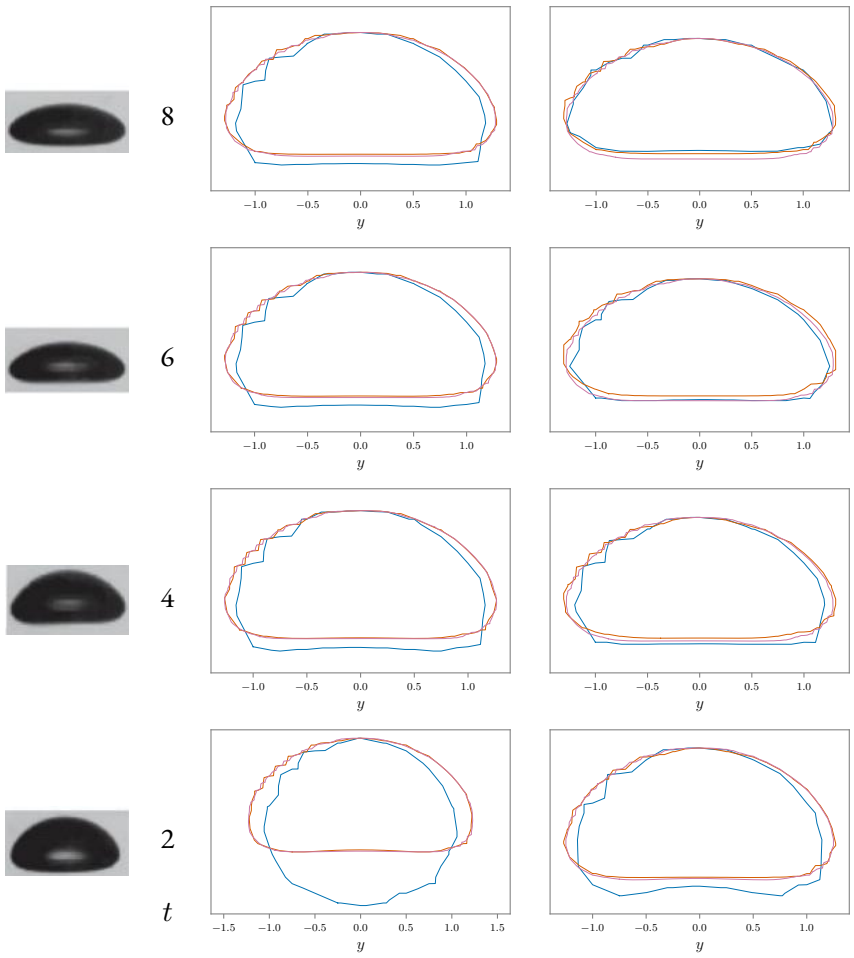


Figure 5.3: Comparison of the shapes of the bubble with the arithmetic (left) and the mixed (right) viscosity average, Morton 2.5×10^{-2}, VOF, with 8, 16 and 32 cells per diameter. The contours are aligned at the top to compare the shapes while ignoring the differences in bubble tip height. The figures in the first row are from an experiment from Sharaf et al. (2017). Reproduced from Sharaf et al. (2017), with the permission of AIP Publishing. t is the non-dimensionalized time.

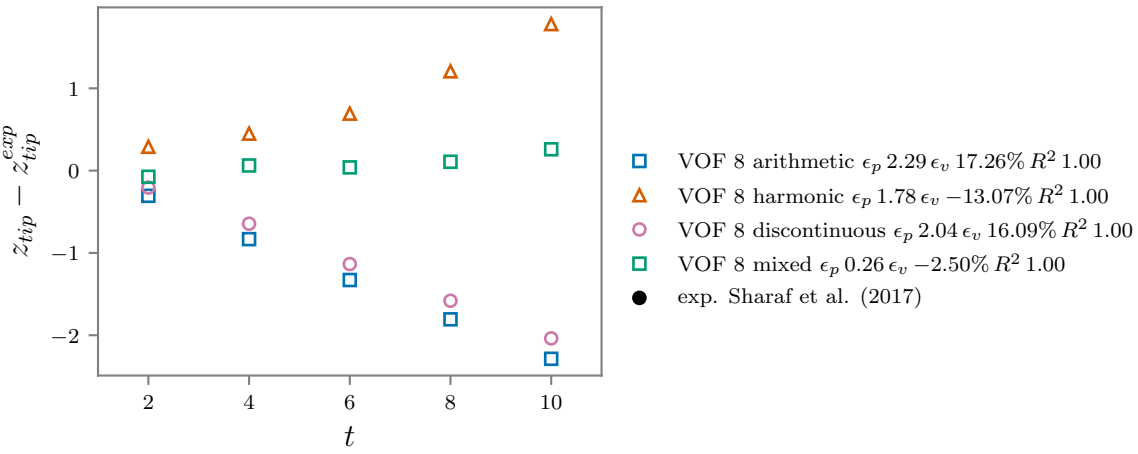


Figure 5.4: Influence of the viscosity average, Morton 2.4×10^{-7}, VOF

Summary. From now on, we use the mixed average for the viscosity, based on the improvement of the velocity.

5.2.2 Influence of the Lagrangian resolution and initialization

The Eulerian grid spacing is denoted h and d is the minimal edge length of the Lagrangian mesh. We run several simulations with a fixed resolution for the Lagrangian mesh, with 1280, 5120 and 20480 faces, corresponding to a ratio h/d of 2.6, 2.6 for the case Morton 28 (fig. 5.5). If the ratio of h the Eulerian grid spacing and d the minimal edge length of the Lagrangian mesh is higher, the interface stretches more and the velocity is closer to the velocity from the experiment. With 20480 faces and 16 cells per diameter, the simulation with the Front-Tracking method agrees more with the experiment than with the VOF method and 32 cells. The velocity is closer to the experiment, and the stretching of the interface seems more similar to the experiment with the Front-Tracking method. However, the thickness of the stretched part is not measured.

We run several simulations with a fixed resolution for the Lagrangian mesh, with 80, 320 and 1280 faces for 8 cells per diameter figs. 5.6 and 5.7. We also compare two initialization procedures, where we either initialize with exact positions or exact discrete volume. For fine Lagrangian meshes, there is not a great sensibility to the initialization method. The shape and velocity error are sensible to the Lagrangian resolution for $Mo = 2.5 \times 10^{-2}$ and $Mo = 3.3 \times 10^{-1}$.

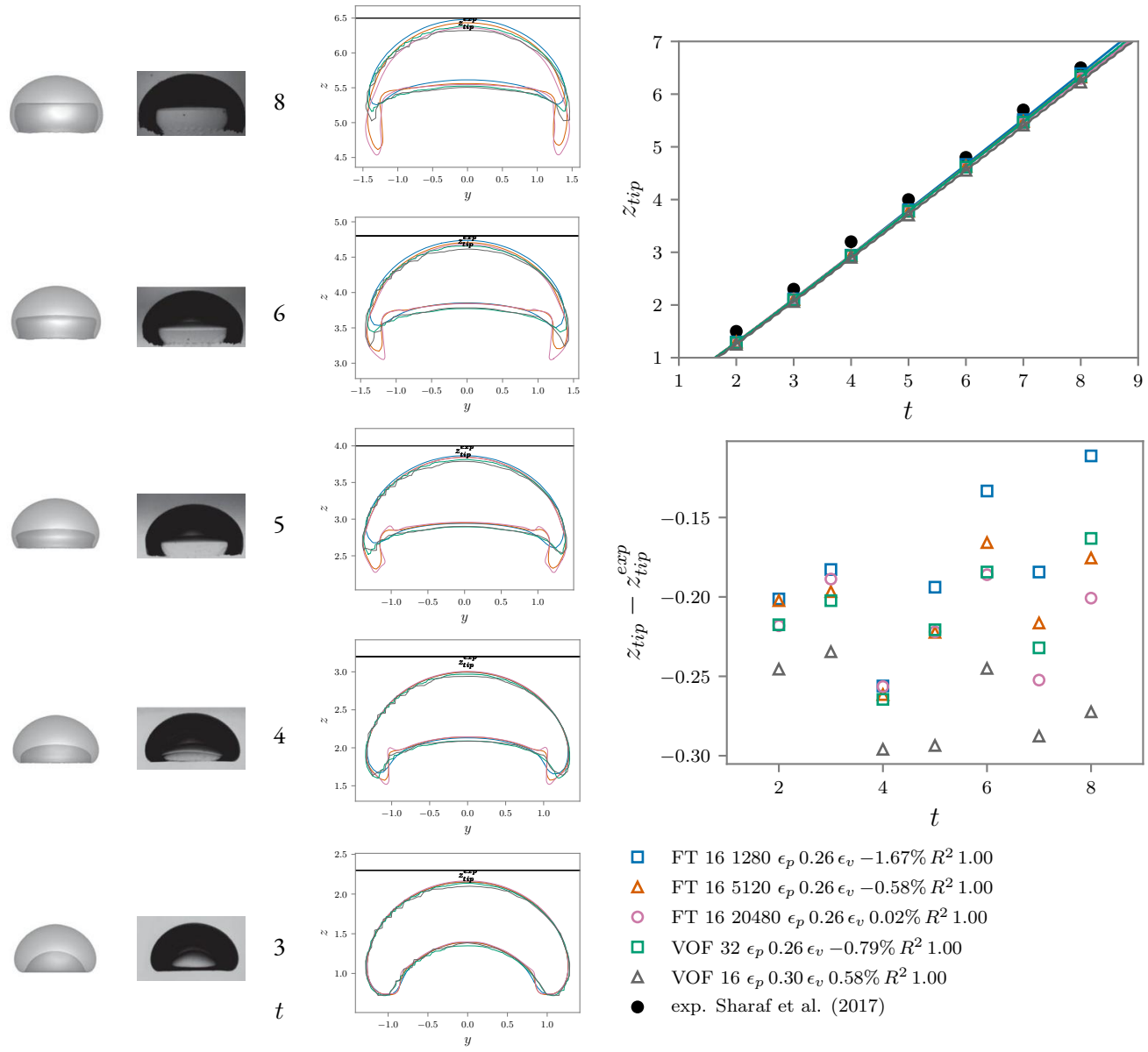


Figure 5.5: Influence of the Lagrangian mesh, Morton 28, VOF and Front-Tracking, 32 cells per diameter, $\Delta t = 10^{-4}$. The results from Sharaf et al. (2017) are indicated in black with their experimental errors. The first two rows are figures from a simulation and experiment from Sharaf et al. (2017). Reproduced from Sharaf et al. (2017), with the permission of AIP Publishing.

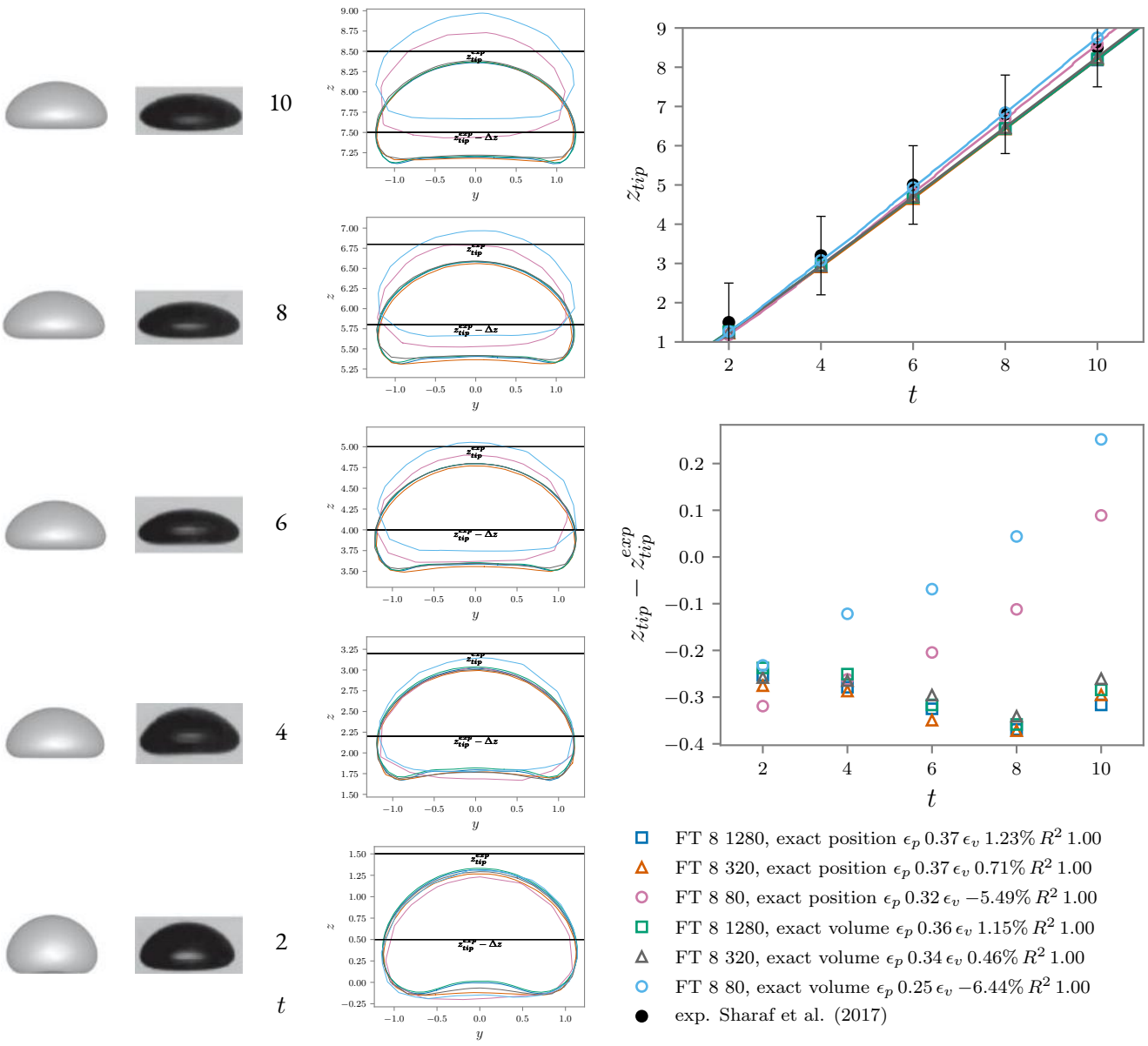


Figure 5.6: Influence of the resolution of the Lagrangian mesh with a fixed resolution, Morton 2.5×10^{-2}, Front-Tracking. The results from Sharaf et al. (2017) are indicated in black with their experimental errors. The first two rows are figures from a simulation and experiment from Sharaf et al. (2017). Reproduced from Sharaf et al. (2017), with the permission of AIP Publishing.

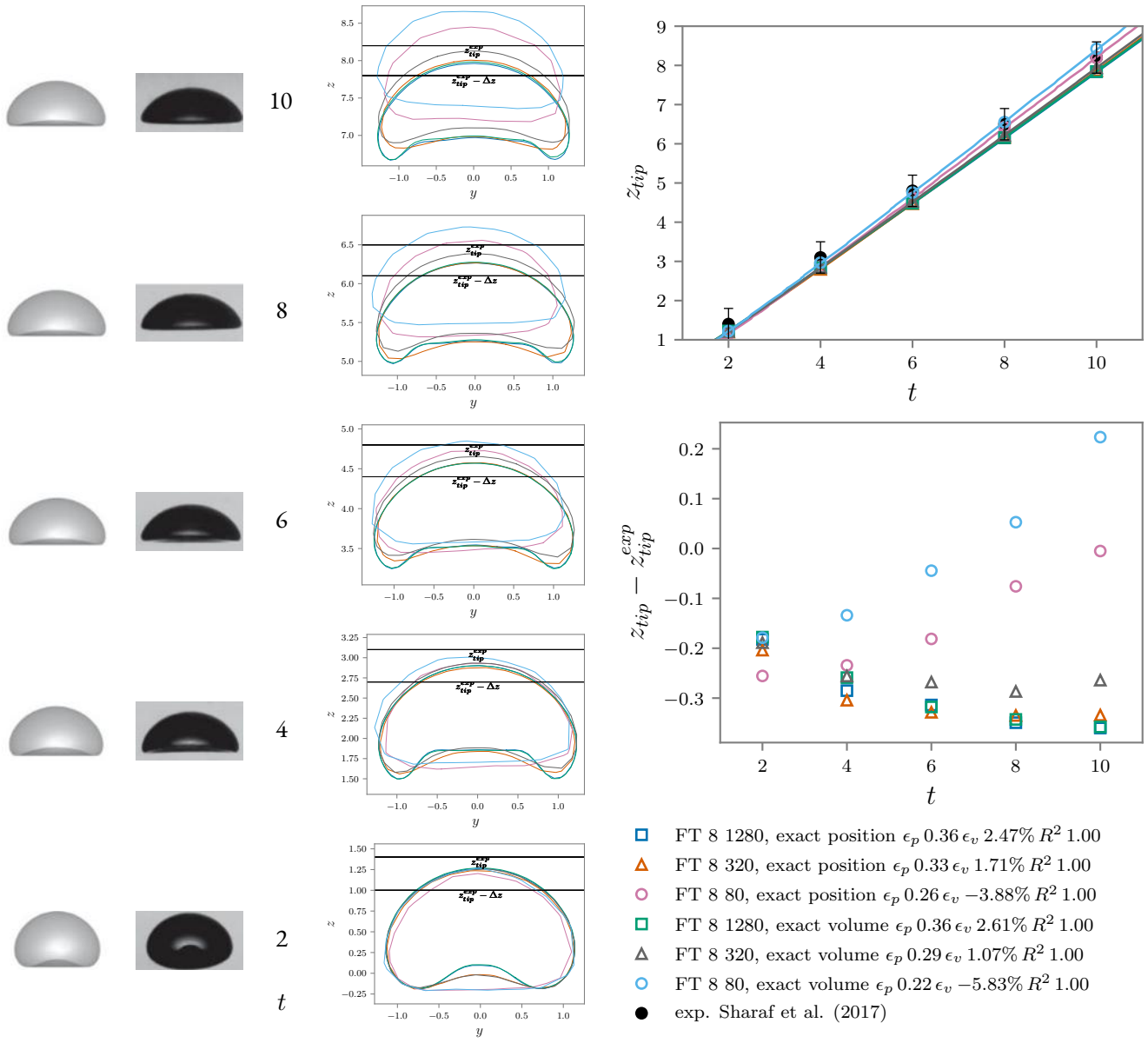


Figure 5.7: Influence of the resolution of the Lagrangian mesh with a fixed resolution, Morton 3.3×10^{-1}, Front-Tracking. The results from Sharaf et al. (2017) are indicated in black with their experimental errors. The first two rows are figures from a simulation and experiment from Sharaf et al. (2017). Reproduced from Sharaf et al. (2017), with the permission of AIP Publishing.

Influence of the surface tension formulation. With a Morton of 2.5×10^{-2}, the height functions and the Integral Formulation give similar results for the computations with the Front-Tracking method (fig. 5.8), the height functions being a little more precise for the velocity and tip height. In this test case, the surface tension does not seem to play an important role.

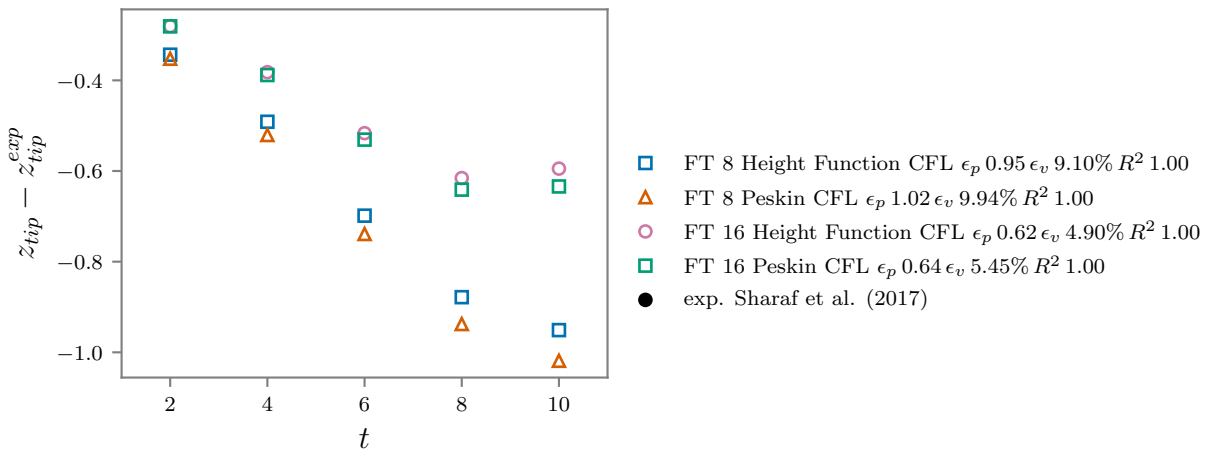


Figure 5.8: Influence of surface tension formulation, Morton 2.5×10^{-2}, Front-Tracking

The Front-Tracking is less precise for the velocity in fig. E.1. The 0.5 isocontour is also plotted for the Front-Tracking method.

SUMMARY. The surface tension formulation did not play an important role in these test cases. We have seen that the description of the viscosity is important, and so is the resolution of the Lagrangian mesh compared to the resolution of the Eulerian grid. The Lagrangian mesh cannot be chosen independently from the Eulerian grid.

Now that we have compared the Front-Tracking method to the VOF method for rising bubbles, we review methods for topological changes in the next part.

6 Topological changes: related work

We present a non-exhaustive list of topological methods used to simulate coalescence and breakup.

Not all interfaces that approach each other are due to coalesce, Qian and Law (1997) identified a bouncing regime for hydrocarbon droplets. Two-phase flows such as sprays are multi-scale, and all scales cannot be resolved. The Navier-Stokes equations do not include the effects of molecular scales, thus controlled topological changes will be required along with Sub-Grid Scale (SGS) modelling to reach mesh convergence. Classical interface capturing methods do not control coalescence and breakup, which can give a wrong outcome. Afanador et al. (2021) simulated the evolution periodic jets while activating topological changes or not. They concluded that some quantities such as phase and plane average velocities are little affected by the activation of the topological procedure. However, the surface area is noticeably larger without the topological procedure. The way topological changes are treated is thus important and we will present the main approaches subsequently.

6.1 Implicit interface tracking

In Front-Capturing methods (VOF,LS), the interface is represented by a scalar on the Eulerian grid. If two parts of the interface come closer than the grid size, coalescence or breakup happens automatically. With such methods, the topological changes are not controlled, unless special methods are used. In Herrmann (2005) a filament removal procedure replaces the cut part with a spherical drop. Jiang and James (2007) manipulates the volume-fraction boundary condition on the symmetry plane to control coalescence, yet this approach cannot be used in general. In Coyajee and Boersma (2009), coalescence is prevented by using one global marker function per droplet instead of one for all droplets for a CLSVOF method. Zhang and Law (2011) developed a film drainage model for the head-on collision of two identical droplets in a gas. They include the physics behind the droplet deformation, the internal flow and the attendant viscous loss, the rarefaction of the gas between the two drops, and the van der Waals force that tends to merge the two drops. Kwakkel et al. (2013) control coalescence and breakup by using one local marker function per droplet instead of one for all droplets in a CLSVOF method. Then coalescence and breakup are activated with film drainage model from Zhang and Law (2011), by respectively merging two marker functions should the contact time of two colliding droplets exceed the predicted film drainage time or splitting one marker function. In this method, the accuracy of the film drainage model is more important than the spatial grid resolution. Based on a time-varying graph-coloring problem, Naru (2021) proposed a more efficient multiple-marker method.

Chiodi (2020) developed a modified approach for the VOF, the Reconstruction with 2 Planes (R2P), where one or two planes are placed iteratively, thus enabling the representation of sub-grid films. Han and Desjardins (2021) developed a Connected-Component Labeling (CCL) approach to identify structures which are due to undergo topological changes with regards to a predefined criterion, such as a volume fraction threshold. Here the connected component is a set of connected Eulerian cells verifying the criterion. This enables to classify fluid structures according to their size. Chirco et al. (2022) detect structures thinner than a predefined criterion to perforate them before the grid resolution does.

Binary head-on collision of identical droplets at high Weber produce a thin lamella, which may artificially rupture for Front-Capturing methods, and the final outcome may be wrong. According to Focke and Bothe (2011), four cells in the lamella (in the normal direction) are required. Roisman et al. (2009) analyzed the head-on collision of identical drops at high impact Weber and Reynolds numbers, where a thin lamella is formed. One way to compute the flow inside the lamella would be to employ AMR. Focke and Bothe (2011) developed a lamella

stabilization method for simulations with a symmetry plane, the volume fraction in the ghost cells is modified to prevent the influence of the opposite side of the interface in the surface tension evaluation. Liu and Bothe (2019) employ a multi-scale approach for colliding drops: a pressure boundary condition, based on a SGS model is applied on the collision plane.

6.2 Explicit interface tracking

Explicit interface tracking methods possess two representation of the interface: the Lagrangian mesh and isosurfaces from a scalar on a grid such as the volume fraction (with its 0.5 isosurface) or a distance. By reconstructing the whole Lagrangian mesh from an isosurface, topological changes are automatic. There is no need for connectivities, vertices are created on the isosurface and triangles are created independently. The difficulty of implementing topological changes is often cited as the main disadvantage of the Front-Tracking method, while the ability to control them is also praised by others.

6.2.1 Topological changes without a supplementary grid

In Nobari and Tryggvason (1996), a Front-Tracking is presented where triangles are removed when drops are close and the surfaces are reconnected. In the Grid-Free (GF), the following steps are applied to the Lagrangian mesh only, without resorting to the grid: intersection detection, retriangulation of intersecting triangles, deletion of invalid triangles and reconnection of the triangles (Glimm et al., 2000). The intersection detection is accelerated by localizing the triangles on a topological grid (typically two or three times coarser than the Eulerian grid). If two triangles intersect, their intersection curve is stored. The intersecting triangles are cut along their intersecting curve and after removing invalid triangles, the newly formed triangles are connected. Nevertheless, the method may fail for complex interfaces. If the interface is invalid, the entire time-step is restarted with a smaller Δt. In Cristini et al. (2001), breakup consists in identifying a region and cutting the mesh with two sets of vertices. These vertices are moved to lie in two respective "splicing planes" The interface is then closed by half spheres. They also perform coalescence. Homma et al. (2006) perform breakup for an axisymmetric jet based on the distance to the axis of symmetry. Quan and Schmidt (2007) perform breakup with unstructured meshes with a predefined distance criterion. In the identified region, the phase of the cells is modified, some nodes are projected to half spheres and smoothing is applied. No volume correction is implemented. Quan et al. (2009) perform coalescence and breakup with unstructured meshes, no volume correction is implemented either. Razizadeh et al. (2018) first identifies triangles which are closer than a predefined distance criterion. Since the number of operations would be $O(N^2)$ with N the number of triangles of the Lagrangian mesh, the triangles are localized in the cells of a grid (the Eulerian grid for instance) in order to reduce the number of operations: the triangles localized in the neighboring cells are tested instead of the whole mesh. Their algorithm identifies pairs of close triangles, and then evaluates the dot product of the normals of the triangles of each pair. If it is positive, the pair is no longer considered as a candidate for topological change. If it is negative, the pair will take part in the topological change: coalescence if the triangles of the pair belong to two different drops, or breakup if they belong to the same drop. The two triangles are "collapsed" into one triangle at half distance, which they call "dual element". If the triangles of a pair do not have any common nodes, they are merged: their nodes are moved halfway and merged, starting by the two closest vertices and proceeding to the remaining four vertices based on the orientation of the triangles. With this procedure, some edges are shared by more than two triangles among the candidates for topological change, and these triangles are merged as well by merging the remaining two vertices. After successive merging operations, an area is formed by the "dual elements". In case of breakup, this area may turn into a thread, the interface is composed of two parts connected by only one node. For coalescence and breakup alike, the "dual elements" are deleted, and vertices that no longer belong to a triangle are deleted as well. Thus, there remains only edges which connect two triangles only. After removing the triangles sharing this vertex, the interface is separated in two parts at least for breakup and close interfaces are merged. This method is local, there is no need for a reconstruction of the whole Lagrangian mesh. No volume correction step is mentioned.

Topological changes are also employed in computer graphics, Computer Aided Design (CAD) and 3D biomedical image analysis. Numerous approaches are developed, such as "boolean operations" on volumes delimited by meshes, such as union, intersection and difference of two volumes, as well as mesh segmentation. Therefore, they may be used for coalescence and breakup, provided that they are robust enough to be used in a simulation, the difference here is that the result provided by the algorithm has to be valid to continue the simulation. Geometric libraries such as Computational Geometry Algorithms Library (CGAL) could be employed to perform the coalescence after the collision occurred, like in the GF method with the mesh editing procedures described in Fabri et al. (2000). They describe a "corefinement" procedure, which consists in: refining two intersecting meshes "so that their intersection polylines are a subset of edges in both refined meshes". Brochu and Bridson (2009) separates a mesh in two when an edge collapsing or swapping generates two triangles with the same vertices [1]. They merge meshes with a "zippering" method, by identifying two close edges. They delete the adjacent triangles, which forms a quadrilateral hole on both meshes. These holes are "zippered" together with a tube constituted of eight triangles. If the creation of the new triangles generates an intersetion, the topological change is cancelled. This method is extended to multiple materials in Da et al. (2014). According to them, the "zippering" approach is overly restrictive because of the intersection tests mainly. They propose an alternative method for nearby edges, which they call "snapping". In 2D, it consists in selecting an edge and the closest vertex from the opposite edge. The edge is split at the closest point to the selected vertex and the two vertices are "snapped" (moved halfway). The same operation is repeated once with the opposite edge and the separating edge is delete. Collisions are less likely during the "snapping" than the "zippering" because the modifications are less important. Schmidt and Brochu (2016) merge intersecting meshes. They refine meshes in the intersection regions, deletes intersecting triangles and fills the holes with an "adaptive mesh zippering" algorithm. Successive iterations where vertices from one "boundary loop" moves towards the nearest vertices from the other "boundary loop". With this displacement, edges may become too long or too short with regards to the predefined criteria, so they are split or collapsed. Eventually, after several iterations, the Euclidean distance defines a bijection between the sets of vertices. Mesh segmentation is a problem treated in many computer graphics applications. First, we need a criterion to determine where to cut the mesh. For that matter we need a distance criterion, but it could be computed based on: the minimal distance, an intersection along the triangle normal, the vertex normal. The Shape Diameter Function (SDF) was developed by Shapira et al. (2008). In Shapira et al. (2008), several rays are sent in a cone centered around the opposite direction of the normal. A mesh segmentation method is implemented in CGAL Yaz and Loriot (2023). A hole filling algorithm is presented in Liepa (2003), combined with a mesh segmentation algorithm, it could perform breakup for triangle meshes.

6.2.2 Topological changes with a supplementary grid

In the Grid-Based (GB) method, topological changes are handled by reconstructing the interface within each cell of a user specified grid, such as the Eulerian grid (Glimm et al., 2000). The intersections between the interface and the grid cell edges are computed (the triangles are localized on a grid for faster intersection tests). Unphysical crossings are deleted, and the interface is reconstructed using the remaining intersections. The Grid-Free (GF) method produces meshes of higher quality than the Grid-Based (GB) method, and conserves mass better but is is less robust. The GB method always produces a valid interface. The Locally Grid-Based (LGB) method is a combination of the GB and GF methods (Li, 2007). Bo et al. (2011) introduced a robust LGB method, aiming at reducing the use of the GB method. Torres and Brackbill (2000) developed the "point-set method", a Front-Tracking method without connectivity.

Shin and Juric (2002) developed the LCRM, which performs coalescence and breakup automatically, not unlike Front-Capturing methods. It is a global reconstruction, the whole mesh is modified. The LCRM can be easily parallelized. However, with linear interpolation, the reconstruction lacks in accuracy and smoothness, and the method tends to redistribute mass between separate interfaces or regions with large differences in curvatures. Shin and Juric (2007) alleviated this problem with high-order interpolation, based on the B-spline described in Torres and Brackbill (2000), thus improving mass conservation and parasitic currents. Singh and Shyy (2007) applies the LCRM only for the interfaces whose "probes" detected a topological event. Shin and Juric (2009)

[1] We prevent this situation instead of letting it happen (subsection 3.5.2, on page 28).

compute a signed distance function from the Lagrangian mesh, instead of using Poisson equation (eq. 2.22, on page 17), which requires more implementations for the boundary conditions when the interface is in contact with the boundaries. When the interface turns into thin structures, the high-order reconstruction from Shin and Juric (2007) suffers from ambiguities for rectangular shaped reconstruction cells: there can be multiple ways to reconstruct the Lagrangian mesh, leading to holes in the interface. Yoon and Shin (2010) circumvent this issue with a marching tetrahedra procedure. The direction of their marching tetrahedra procedure is alternated as it affects the topological outcome. Ceniceros (2010) use a hybrid LS / Front-Tracking method with AMR, and the Closest Point Transform (Mauch, 2000) to compute the signed distance function. Tolle et al. (2020) adapted the LCRM to unstructured meshes with AMR. One difference is that they use connected triangles and reconstruct the distance with a marching tetrahedra procedure Treece et al. (1999). Shin et al. (2011) developed the Level Front Reconstruction Method (LFRM), a geometrical method which reconstructs the Lagrangian mesh by computing intersections with a structured grid. It does not use an isosurface on a reconstruction grid, but only the Lagrangian mesh, and improves volume conservation. Like with the LCRM, connectivities are not required, the triangles can be reconstructed independently. The triangles are localized on the reconstruction grid. The cells which contain a portion of the interface are called reconstruction cells. The resolution of the LFRM can be adapted by subdividing the reconstruction cells since it does not depend on a scalar on the Eulerian grid. The adaptive LFRM is more efficient than employing the LCRM with a homogeneously refined grid (Shin et al., 2011). After the localization, the triangles are cut by the reconstruction cells. A local volume conservative reconstruction is then applied in the reconstruction cells. Rajkotwala (2020) coupled the LFRM with the film drainage model from Zhang and Law (2011). Chirco and Zaleski (2023) developed the Edge-Based Interface-Tracking (EBIT) method for 2D flows, where the vertices of the Lagrangian mesh lie on the edges of the Eulerian grid. A binary phase indicator function from Singh and Shyy (2007) is used in Pan et al. (2023) to perform topological changes automatically, which affects the volume enclosed by the interface. This function is updated during the transport of the interface.

A non-exhaustive list of topological methods based on Lagrangian meshes and applied to two-phase flows is presented in table 6.1.

Publication	Information	Coalescence	Breakup	Dimension
Glimm et al. (2000)	Lagrangian	✓	✗	3D
Glimm et al. (2000)	Eulerian	✓	✓	3D
Torres and Brackbill (2000)	Eulerian	✓	✓	3D
Cristini et al. (2001)	Lagrangian	✓	✓	3D
Shin and Juric (2002)	Eulerian	✓	✓	3D
Shin et al. (2005)	Eulerian	✓	✓	3D
Homma et al. (2006)	Lagrangian	✗	✓	Axisymmetric
Li (2007)	Mixed	✓	✓	3D
Quan et al. (2009)	Lagrangian	✓	✓	3D
Ceniceros (2010)	Eulerian	✓	✓	3D
Bo et al. (2011)	Mixed	✓	✓	3D
Shin et al. (2011)	Lagrangian	✓	✓	3D
Razizadeh et al. (2018)	Lagrangian	✓	✓	3D
Tolle et al. (2020)	Eulerian	✓	✓	3D
Pan et al. (2023)	Eulerian	✓	✓	2D

Table 6.1: Non-exhaustive list of topological methods employed with explicit interface tracking. The source of the information for topological change is indicated: Lagrangian mesh, Eulerian mesh (with an implicit representation of the interface), or mixed: Eulerian and Lagrangian. The check and cross marks indicate if coalescence and breakup were mentioned for the method.

6.3 Summary

Unlike Front-Capturing approaches, the classical Front-Tracking method does not handle topological changes and it is a well-known difficulty, which can be circumvented by resorting to an implicit description of the interface (an isosurface) or by projecting the mesh onto a topological grid. However, the whole mesh is modified, and we wish to preserve the mesh as much as possible to prevent the accumulation of position errors and in the hope to save up computational time. Alternatively, the Lagrangian mesh can be modified without a projection on a grid, and this is the approach we chose. The GF method and the "corefinement" methods are interesting, yet they cannot be employed to perform coalescence before two meshes intersect. The method from Razizadeh et al. (2018) is also interesting, yet we would like to present an alternative method. Our aim is to identify regions to be merged or separated and treat special cases, like when a region is small, we may not want to merge it. Our purpose is to develop a method for coalescence and breakup, where the changes to the existing meshes are limited and where the volume can be conserved globally.

In the next chapter, we present a method for coalescence and a method for breakup and apply them to numerical and experimental test cases.

7 Coalescence and breakup

In which we present a method to simulate coalescence and a method for breakup, The coalescence and breakup procedures are tested with numerical test cases and experiments.

We present here two geometrical methods designed to handle coalescence and breakup for simple topological cases, using the vertices and connectivities of the Lagrangian mesh. Each bubble/drop is contained in a numbered object called *bubble* and possessing its own list of triangles, vertices (markers) and the connectivities to identify the neighbors of a triangle. The coalescence is performed by identifying vertices of one bubble within a predefined range of other bubbles. Then these points are projected on the corresponding bubble and the mesh of each bubble is adapted to have a common polygon which is used to link both bubbles. This method is accelerated by a localization of the triangles on the Eulerian mesh.

Hypothesis. The algorithm presented below cannot handle multiple structures merging in the same region. Here we suppose that when two *bubbles* are about to coalesce, we can identify two regions on each *bubble* involved in the process, and that no other *bubble* is involved in the vicinity of these regions. In other words, multiple *bubbles* may coalesce with one *bubble* if only two *bubbles* are involved locally.

7.1 Coalescence between distinct bubbles/drops

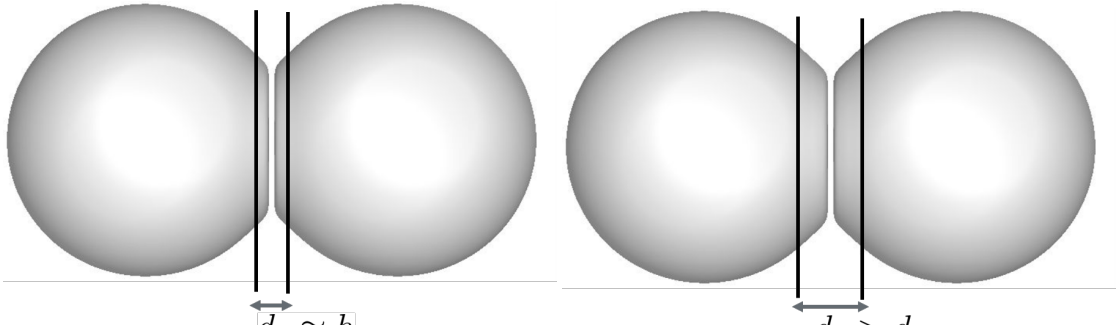
The triangles are located on the structured Eulerian scalar mesh to accelerate the distance computations between the vertices or centroids of the triangles of different bubbles. The distance computations later mentioned are reduced to examining triangles in a few cubic volumes given by the Eulerian mesh. The main steps of the coalescence algorithm are summarized in algorithm 4.

Algorithm 4: Coalescence algorithm

1. Identification of the bubbles to be merged
2. Identification of vertices at a minimal distance inferior to d_2
3. Construction of a temporary region of coalescence on the first *bubble* and of a projection vector $\mathbf{p_{moy}}$
4. Creating vertices at nearly d_2
5. Construction of the coalescence region of the first *bubble*
6. Construction of the image of the tagged vertices on the polygon of the coalescence region
7. Adapting the boundaries of the coalescence region for a bijection between the two *bubbles*
8. Connection of the *bubbles*
9. Remeshing after the coalescence

Identification of the bubbles to be merged. The coalescence method between *bubbles* is activated when the distance between the centroid of some triangles of two *bubbles* or more are at a distance inferior to d_1 ($d_1 = h$ the Eulerian mesh spacing, for example), as depicted in fig. 7.1. In that case, the *bubble* groups to be merged are identified. For the sake of simplicity, we will consider next only one pair of *bubbles*.

Identification of vertices at a minimal distance inferior to d_2. For the considered pair of *bubbles*, the smallest Euclidean distance of the vertices of one *bubble* to the other is computed (figs. 7.5a and 7.5b). If this distance is inferior to d_2, which is chosen superior to d_1 ($d_2 = 1.5d_1$ for example), the vertex is tagged by

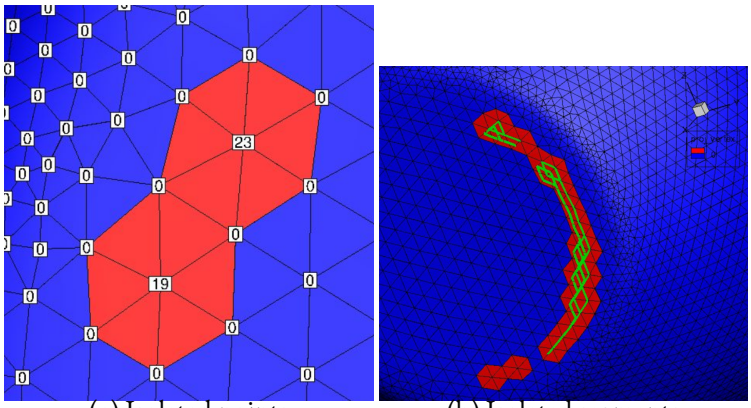


(a) Activation (b) Modifications

Figure 7.1: Distance criteria for coalescence

an integer value. The projection vectors $\mathbf{p}$ of the vertices on the interface of the second *bubble* are kept after normalization if their norm is inferior to d_2.

CONSTRUCTION OF A TEMPORARY REGION OF COALESCENCE ON THE FIRST *BUBBLE* AND OF A PROJECTION VECTOR $\mathbf{p_{moy}}$. A temporary region is constructed with the adjacent triangles whose vertices are all tagged, illustrated by fig. 7.5a (the vertices whose normalized distance is inferior to 1). Isolated vertices are untagged and will not intervene in the coalescence (fig. 7.2).



(a) Isolated points (b) Isolated segments

Figure 7.2: Special cases

We do not project edges or isolated points, it has to have a thickness of one triangle at least. Sometimes regions are connected by a projected vertex. So we project vertices in the N_1 neighborhood before adapting the mesh on the other side as long as there remains connected regions (fig. 7.3). Similarly, if isolated points or edges are not projected inside a coalescence region, we force their projection. Coalescence regions are merged if they share a vertex.

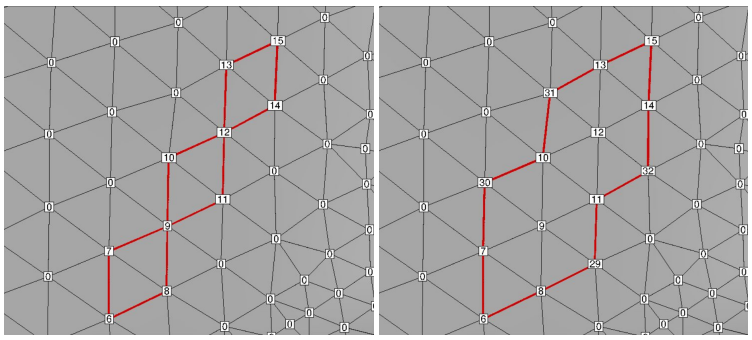
[!hbt] 

Figure 7.3: Special cases: connected regions (left), solution: the region is widened (right)

We do not use the minimal distance to adapt the meshes in the last stages. Our objective is to identify two equivalent regions, delimited by equivalent polygons. The projections vectors of two *bubbles* are not necessarily colinear and thus the polygons are different (fig. 7.4). The two regions would be asymmetric: the minimal distance is not bijective. We compute the minimal distance in a first step to determine a temporary region on the first *bubble*. We then determine a vector for each region by averaging the directions, which is bijective.

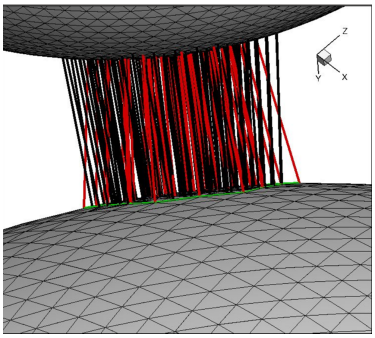


Figure 7.4: Illustration of a few projections with minimal distance between two close *bubbles*, the minimal distance vectors from the top *bubble* to the bottom *bubble* are illustrated in black, the minimal distance vectors from the bottom *bubble* to the top *bubble* are illustrated in red

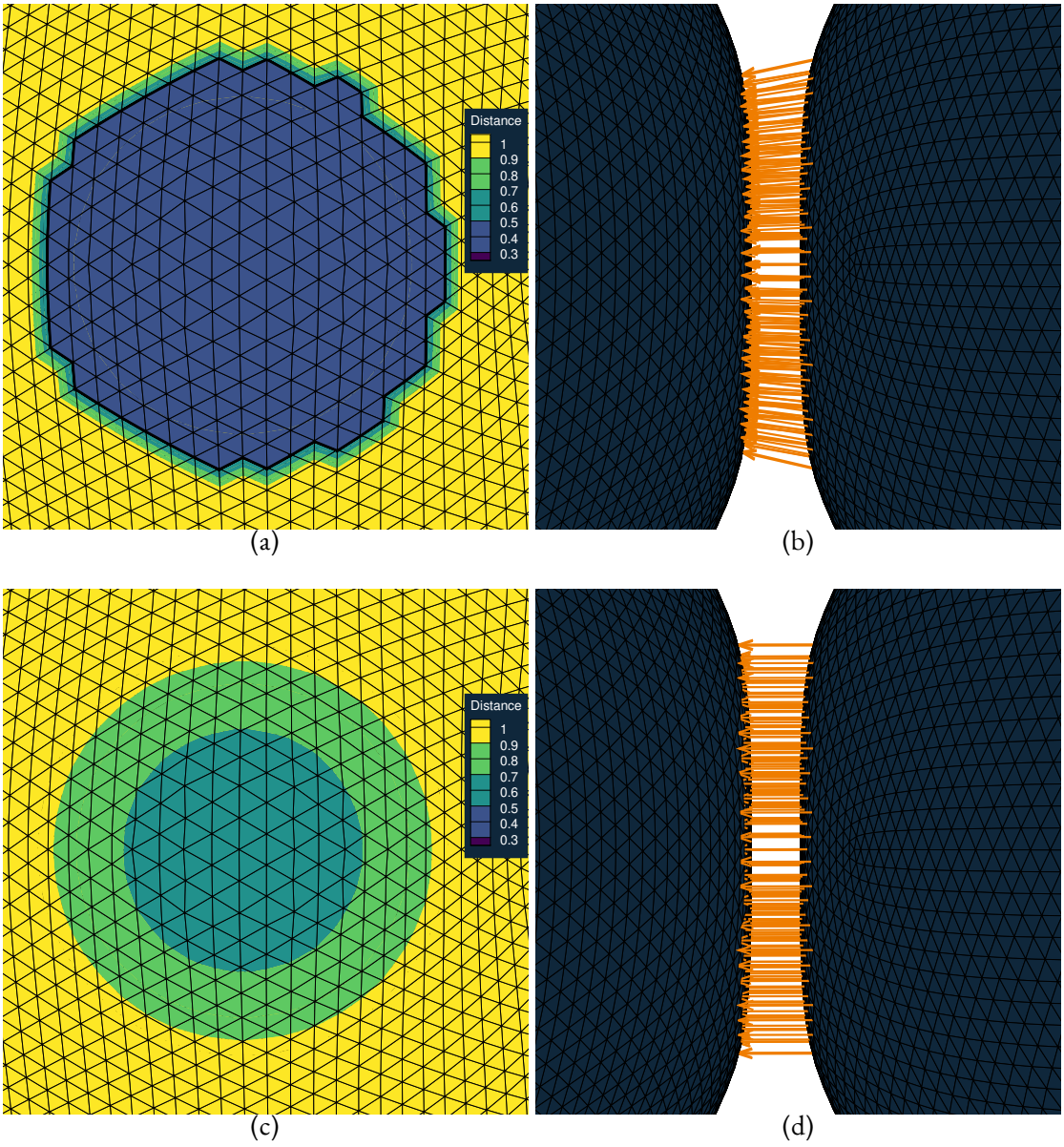


Figure 7.5: Minimal distance (a) and distance along the projection vector $\mathbf{p_{moy}}$ (b) (normalized by d_2)

A unique projection vector $\mathbf{p_{moy}}$ is defined on the temporary region by averaging the normalized projection vectors $\mathbf{p}$ for all tagged vertices belonging to the region. From this point forward, distance computations for this region are performed with an intersection test with a segment colinear to $\mathbf{p_{moy}}$ (figs. 7.5d and 7.6). This new distance is more regular (fig. 7.5c). In order to accelerate the distance computations, only the triangles localized in the scalar control volumes traversed by the ray are examined. The ray-traversal algorithm examines the intersection of the faces of a control volume to proceed to the next (fig. 7.6).

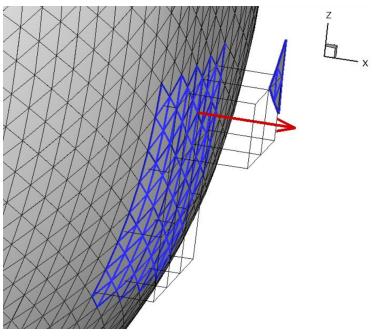


Figure 7.6: Intersection test along $\mathbf{p_{moy}}$ for the distance computation of the coalescence method, the ray is illustrated in red, the cubic cells are the scalar control volumes used to accelerate the distance computations

Creating vertices at nearly d_2. In order to improve the shape of the region between the two *bubbles*, the edges with a unique tagged vertex are examined. By denoting $d > d_2$ and $D < d_2$ the distances along $\mathbf{p_{moy}}$ to the second *bubble* of the vertices s (non tagged vertex) and S (tagged vertex), a new vertex N is introduced on the edge so that its projection on the other *bubble* is nearly at d_2. The position $\vec{x}_N$ of the new point N is obtained by linear interpolation :

$$\vec{x}_N = \frac{d_2 - D}{d - D}\vec{x}_s + \frac{d - d_2}{d - D}\vec{x}_S$$

This new point is tagged as well. Adding a new vertex implies creating new triangles and updating connectivities (fig. 7.7a).

Construction of the coalescence region of the first *bubble*. A coalescence region is then constructed the same way the temporary region was but with the newly created vertices. The triangles inside the coalescence region are tagged by an integer value.

Construction of the image of the tagged vertices on the polygon of the coalescence region. In order to construct the image on the second *bubble* of the coalescence region, the projection vector $\mathbf{p_{moy}}$ is used. This unique vector guarantees the bijection between the vertices of the coalescence regions of the two *bubbles* and gives a more regular shape of the interface after the coalescence.

The tagged vertices at the polygon of the coalescence region are projected on the second *bubble* in the direction of the projection vector and generate new vertices and triangles on the second *bubble*. These new vertices are tagged with the same number than the projected vertices. With the intersection algorithm from Woop et al. (2013) we know the signed distance to the edges of the examined triangle. If two signed distances are null, there is no need to create a new point for the projection hits a vertex. If one signed distance is null, an edge is hit so the adjacent triangles are split in two. If all signed distances are non null, the triangle is split in three.

Adapting the boundaries of the coalescence region for a bijection between the two *bubbles*. The next step consists in adapting the meshes so that there is bijection between the edges of the polygons of the coalescence regions. First, the polygon of the coalescence region of the first *bubble* is extracted. Let S'_1 et S'_2 be the images of S_1 and S_2 by the projection defined previously. If S'_1 and S'_2 do not share an edge, a set of edges must be created to link them.

A unit vector $\mathbf{n} = (\vec{x}_{S'_2} - \vec{x}_{S'_1})/\|\vec{x}_{S'_2} - \vec{x}_{S'_1}\|$ is defined. From S'_1, the intersection of the ray from that point and directed by the unit vector, projection of $\mathbf{n}$ in the plane of one of the triangles having S'_1 as one of its vertices, and the opposite edge is constructed : a new vertex A'_1 is created (and the associated triangles). This procedure is reiterated from this new vertex to create a series A'_i (of increasing index) of vertices until S'_2 is reached (fig. 7.7b). Since new vertices are generated on the second *bubble* between S'_1 and S'_2, the same work has to be done on the

first *bubble*, on the edge formed by S_1 et S_2. By denoting $A'_0 = S'_1$, $A'_n = S'_2$ and $A_0 = S_1$, $A_n = S_2$, the new vertices A_i on the edge are positioned to verify the relation:

$$\frac{d(A_{i+1}A_i)}{\sum_{i=0}^{n-1} d(A_{i+1}A_i)} = \frac{d(A'_{i+1}A'_i)}{\sum_{i=0}^{n-1} d(A'_{i+1}A'_i)} \tag{7.1}$$

with $d(A_{i+1}A_i)$ the Euclidean distance between A_{i+1} and A_i.

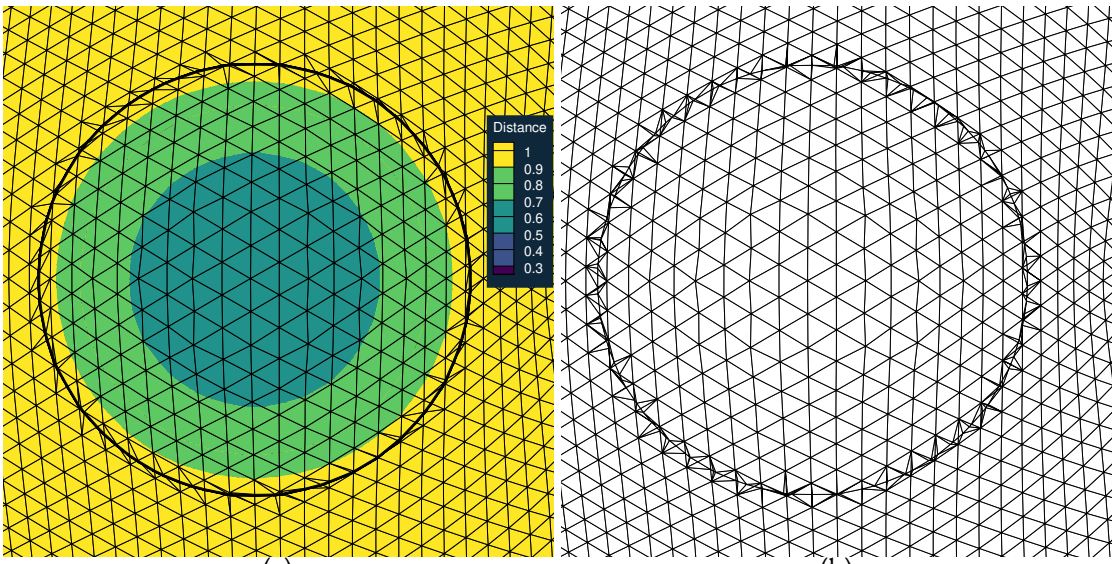


Figure 7.7: Mesh refinement of the first *bubble* (a) and mesh adaptation of the second *bubble* (b)

Connection of the *bubbles*. Once the polygons are defined on each *bubble*, the triangles and vertices inside the polygon are deleted (fig. 7.8a). The pairs of vertices S and S' are merged and moved halfway between their original position, $(\vec{x}_S + \vec{x}_{S'})/2$. The connectivities of the new *bubble* are updated by creating two triangles for each common edge[1], as depicted in fig. 7.8a.

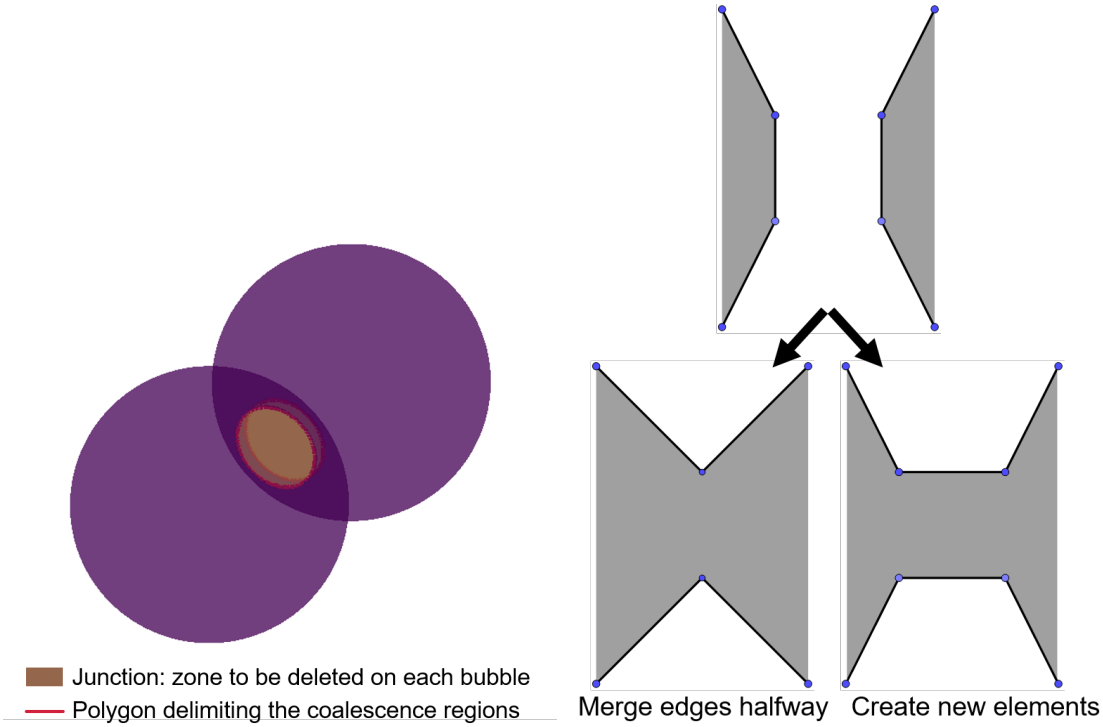
Remeshing after the coalescence. The mesh at the junction between the two *bubbles* has generally a poor quality. This junction is smoothed with an edge-based volume conserving smoothing (Kuprat et al., 2001). Even if the shape of the junction is satisfactory, its triangles are too large with regards to the curvature generated by the coalescence. It is necessary to refine the mesh in the neighborhood of the junction. Each triangle sharing a vertex on the junction is split without resorting to surface reconstruction. The quality of the reconstruction would be bad because of the lack of resolution. Badly shaped triangles are deleted. The transport by the velocity field in the next time steps will help improve the quality of the mesh with the surface tension.

7.2 Breakup of a ligament

A method for the breakup of a ligament is now presented. Some of the steps are similar, but the distance is computed with the vertex normals.

Identification of the bubbles which will break. The breakup method is activated for a *bubble* when the distance along the vertex normal is inferior to d_3 ($d_3 = h$ the Eulerian mesh spacing, for example). The mesh is separated based on a distance criterion d_4 which is chosen superior to d_3. Regions of breakup are identified,

[1] Another way to connect the two *bubbles* would be to reconnect the triangles with equivalent edges.



(a) Deletion (b) Merging

Figure 7.8: Connecting the two *bubbles*

delimited by segments. Isolated segments are ignored. Linear interpolation is also used to improve the polygons delimiting the regions. Connected regions are treated like in coalescence. We may refer to these regions as clusters, and they are delimited by polygons. These clusters are then due to be separated. We may merge clusters based on several criteria. For example if a cluster has a small area compared to the area of the *bubble*, we may merge it with one of its neighboring clusters, thus cancelling a separation. We also compute an average normal for the cluster. If the minimal dot product with this average normal and the normals of the cluster is positive, we may merge this cluster with one of its neighbors. This can prevent identifying clusters at the end of a ligament.

Mesh segmentation. The final clusters are separated in different *bubbles*. The holes caused by the mesh segmentation are filled with a patch of triangles along the polygons separating the clusters. The patches are smoothed with the edge-based volume conserving smoothing (Kuprat et al., 2001). *Bubbles* which are smaller than the distance criterion can either be deleted or maintained. If the small objects are deleted, their volume is redistributed equally to the structures that were in contact with it. The advantage is that the newly formed *bubbles* are distant, this prevents intersections. If the small *bubbles* are maintained, they have to be displaced with a SGS model to resume the computation without provoking intersections.

Entrapment. Air entrapment may be seen as a breakup event for the other phase, so the breakup procedure is adapted and a ray is generated on both sides of the interface to detect both breakup and entrapment events (fig. 7.9). The difference with air entrapment is that it does not generate a new *bubble* of the same phase but a *bubble* of the other phase, so the newly formed Lagrangian mesh must be added to the current Lagrangian mesh. Entrapment is detected when the newly formed part has not the same orientation: the volume is negative. During air entrapment, we may prefer to keep the subgrid *bubbles* which are trapped inside the original *bubble*.

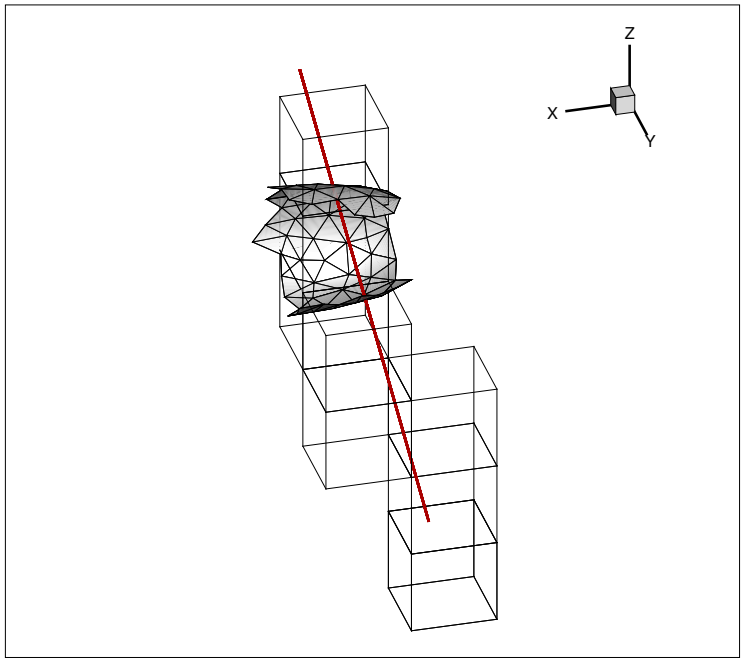


Figure 7.9: Air entrapment, the ray is illustrated in red, the cubic cells are the scalar control volumes used to accelerate the distance computations, the part of the interface that was examined for the current vertex is illustrated in gray.

7.3 Simple coalescence and breakup test cases

We employ the coalescence and breakup methods in four test cases with simple topological events: breakup of a drop due to variable tension, breakup of a drop in a shear flow, binary collision of identical droplets (followed by coalescence and reflexive separation) and binary collision of unequal droplets (followed by coalescence and air entrapment).

7.3.1 Breakup generated by a variation in surface tension

We consider the case presented in Abu-Al-Saud et al. (2018), where a bubble breaks due to a variation in surface tension. A drop of diameter $D = 2$ is placed at the center of a domain of dimensions $7D \times 3D \times 3D$ with Dirichlet boundary conditions. The Laplace number (La) is thus defined:

$$\mathrm{La} = \frac{\rho_1 \sigma_0 D}{\mu_1} \tag{7.2}$$

where ρ_1 and μ_1 are the density and viscosity of the bubble. The fluid properties are summarized in (table 7.1).

ρ_1	ρ_2	μ_1	μ_2	g	σ_0	La
$(kg.m^{-3})$	$(kg.m^{-3})$	$(Pa.s)$	$(Pa.s)$	$(m.s^{-2})$	$(kg.s^{-2})$	$-$
1	25	1	25	0	1.72	3.44

Table 7.1: Fluid and physical properties.

The surface tension coefficient varies in the x-direction fig. 7.10, it is non-null in the whole domain:

$$\sigma(x) = \sigma_0 \max(1 - 1.25|x/R|, 0.1) \tag{7.3}$$

We use the Integral Formulation (IF) with Peskin distributions since it takes into account variable surface tension coefficients. Peskin's functions are also used for the velocity interpolation. Symmetry boundary conditions are used. The breakup method is activated with an activation criterion $d_3 = 4h$ with h the Eulerian mesh spacing and a breakup criterion $d_4 = 6h$ as depicted in fig. 7.11. The structure smaller than the breakup criterion is deleted and its volume is distributed equally to the adjacent *bubbles*.

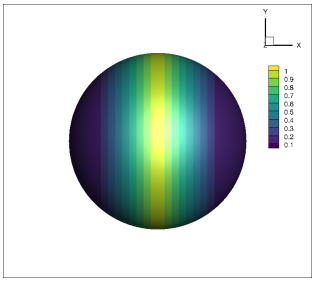


Figure 7.10: Distribution of the non-dimensionalized surface tension σ/σ_0 on the bubble at $t = 0$, test case from Abu-Al-Saud et al. (2018)

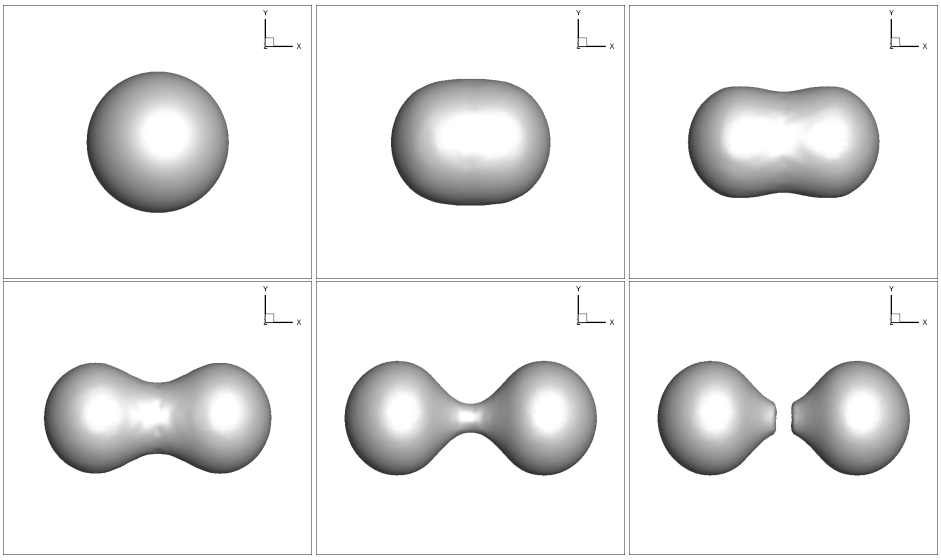


Figure 7.11: Breakup due to variable surface tension

7.3.2 Shear flow

We consider a drop sheared by two opposite velocities (Li et al., 2000; Razizadeh et al., 2018). The domain dimensions are $3 \times 1 \times 1$. A coarse mesh is used: $dx = dy = dz = 1/32$, there are 16 cells per drop diameter. The breakup criterion is $d_4 \approx 0.5$. No-slip boundary conditions are used in the $\mathbf{Z}$ direction, and periodic in the $\mathbf{X}$ and $\mathbf{Y}$ directions. The liquids have the same density and unit viscosities. The drop has a radius R and viscosity μ. Two plates are placed at a distance of d. A shear rate is imposed with u the magnitude of the velocities of the plates:

$$\dot{\gamma} = \frac{2u}{H} \tag{7.4}$$

The Reynolds and capillary numbers are thus defined:

$$Re = \frac{\dot{\gamma} a^2}{\nu} \tag{7.5}$$

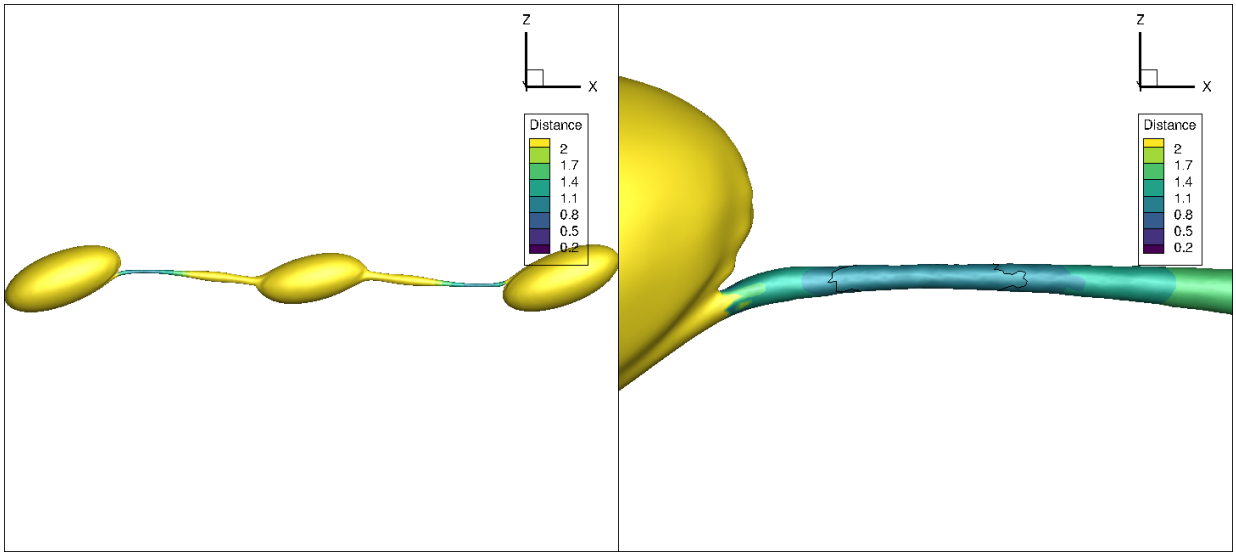


Figure 7.12: Breakup of a drop in a shear flow: distance contour

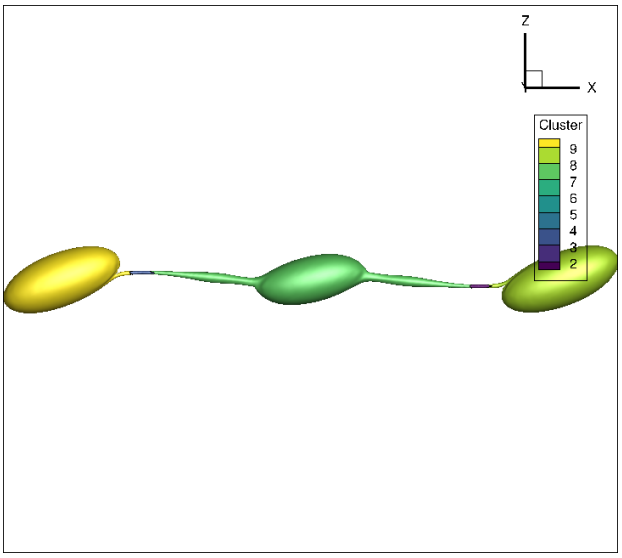


Figure 7.13: Breakup of a drop in a shear flow: cluster identification

$$Ca = \frac{a\dot{\gamma}\mu_2}{\sigma} \tag{7.6}$$

We perform a simulation with $\mathrm{Re} = 0.2$ and $\mathrm{Ca} = 0.42$. Two parts of the sheared drop are smaller than the breakup criterion as depicted in fig. 7.12. The breakup algorithm identifies five parts to be separated (fig. 7.13).

7.4 Binary drop collision

We attempt to reproduce the experiments made by Ashgriz and Poo (1990). Two drops of water collide in the air. The collision outcomes are often plotted with the Weber number and the non-dimensionalized impact parameter x which measures the offset from head-on collision:

$$x = \frac{X}{D_1 + D_2} \tag{7.7}$$

with X the distance between the straight lines crossing the drop centers and colinear to the relative velocity vectors, D_1 is the diameter of the smaller drop and D_2 the other diameter. The Weber number here is defined as:

$$\mathrm{We} = \frac{\rho_1 D_1 U}{\sigma} \tag{7.8}$$

with U the relative velocity, ρ_1 the liquid density and σ the surface tension coefficient. At low We, the drops are slow enough to coalesce. In some binary drop coalescence regimes, thin films of air or water form, and this can prove difficult to simulate. We use the Momentum Conserving and coupled solver developed in (Elouafa, 2022; El Ouafa et al., 2023). The computations are not as stable without the Momentum Conserving. We use the Integral Formulation (IF) and Peskin's interpolation.

7.4.1 Initialization of the velocity

For a given Weber number, we need to initialize the simulation so that the drops reach a velocity of $(U/2, 0, 0)$ and $(-U/2, 0, 0)$. Coalescence can be initialized with a force to accelerate the drops like in Nobari and Tryggvason (1996). In Rajkotwala et al. (2020), a divergence-free velocity field is initialized, with four vortices. This avoids a pressure jump at the beginning of their simulations. In our implementation, a constant velocity is initialized in and around the drops. The number of cells outside is determined by the stencil of the transport schemes (2 for Peskin's interpolation), so that the vertices are initially transported with the exact velocity. However, with a constant velocity around the drops, the divergence is not null. In Nangia et al. (2019), a projection step is used for a droplet splashing on a film. In our implementation, the velocity field is made divergence-free with a resolution with the coupled solver (Ouafa, 2021), equivalent to a projection step.

In the i scalar control volume V_i, we integrate the divergence d_i:

$$d_i = \sum_f \mathbf{v} \cdot \mathbf{n} S_f \tag{7.9}$$

where S_f is the surface of the face f of the scalar control volume i.

We define the relative error on divergence in l_∞ and l_2 norms as:

$$l_\infty = \frac{\max|d_i|}{\max(\sum_f |\mathbf{v} \cdot \mathbf{n} S_f|)} \tag{7.10}$$

$$l_2 = \sqrt{\frac{\sum_i V_i (d_i)^2}{\frac{1}{2}\sum_i V_i \sum_f (\mathbf{v} \cdot \mathbf{n} S_f)^2}} \tag{7.11}$$

The projection step reduces the divergence error in l_2 and l_∞ norm, as we can see in table 7.2 for the case $\mathrm{We} = 30$.

	l_2	l_∞
Before projection	1.0	3.1×10^{-1}
After projection	1.5×10^{-1}	1.1×10^{-3}

Table 7.2: Example of divergence errors before and after projection with a preconditioned residual of 10^{-4}

The parameters are summarized in table 7.3 for both cases.

We	U	D_1	D_2	Δ	x	D_1/h
23	1.44	8.0×10^{-4}	8.0×10^{-4}	1.00	0.05	25.6
30	2.05	5.2×10^{-4}	8.0×10^{-4}	0.65	0.05	16.6

Table 7.3: Fluid and physical properties for the binary drop collision cases, D_1 is the diameter of the smaller drop and D_2 the other diameter, h is the Eulerian mesh spacing

7.4.2 Reflexive separation, We $= 23$

Two equal drops collide, coalesce and form a thin film, after that a reflexive separation occurs. The film breakup is not implemented, so numerical breakup does not happen. The thin liquid film is ill-resolved and this delays the reflexive separation. We would very fine a fine Eulerian resolution or a Sub-Grid Scale model to simulate the film correctly. The following criteria are used: coalescence criterion $d_2 = 3h$, coalescence activation criterion $d_1 = 2h$, breakup criterion $d_4 = 3h$ for a mesh $128 \times 64 \times 64$. We use a CFL of 0.5, based on convection.

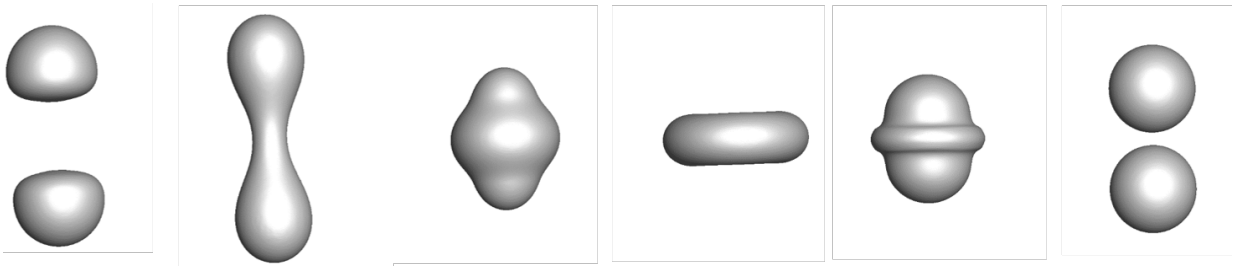


Figure 7.14: Binary drop collision, We $= 23$

7.4.3 Air entrapment, We $= 30$

Two unequal droplets collide and coalesce with We $= 30$. A bubble is trapped by the newly formed drop. We use a mesh $96 \times 64 \times 64$, with a CFL of 0.5. The following criteria are used: coalescence criterion $d_2 = 3h/2$, coalescence activation criterion $d_1 = h$, breakup activation criterion $d_3 = h$ and breakup criterion $d_4 = 3h/2$. The method for the coalescence between two *bubbles* is activated between the two first frames of fig. 7.15. The breakup method identifies a "ligament of air", which is cut as shown in fig. 7.16. The subgrid *bubble*, i.e. the trapped air bubble, is not removed and thus conserved in the same data structure as the drop. A Sub-Grid Scale model is required to represent the dynamics of air entrapment. The two structures are side by side and with the smoothing step they intersect each other and it is not possible to resume the computation.

7.5 Summary

The proposed methods for coalescence and breakup can be used for simple topological changes, and the simulations come to a qualitative agreement with numerical and experimental test cases. Not all topological changes are handled yet, like the breakup of a liquid sheet. For now, the criteria are geometric, we did not include a coalescence or breakup model. For the breakup method, the distance may be computed with several rays like in Shapira et al. (2008) to obtain a more regular distance. For complex topological changes, resorting to an implicit representation of the interface seems necessary.

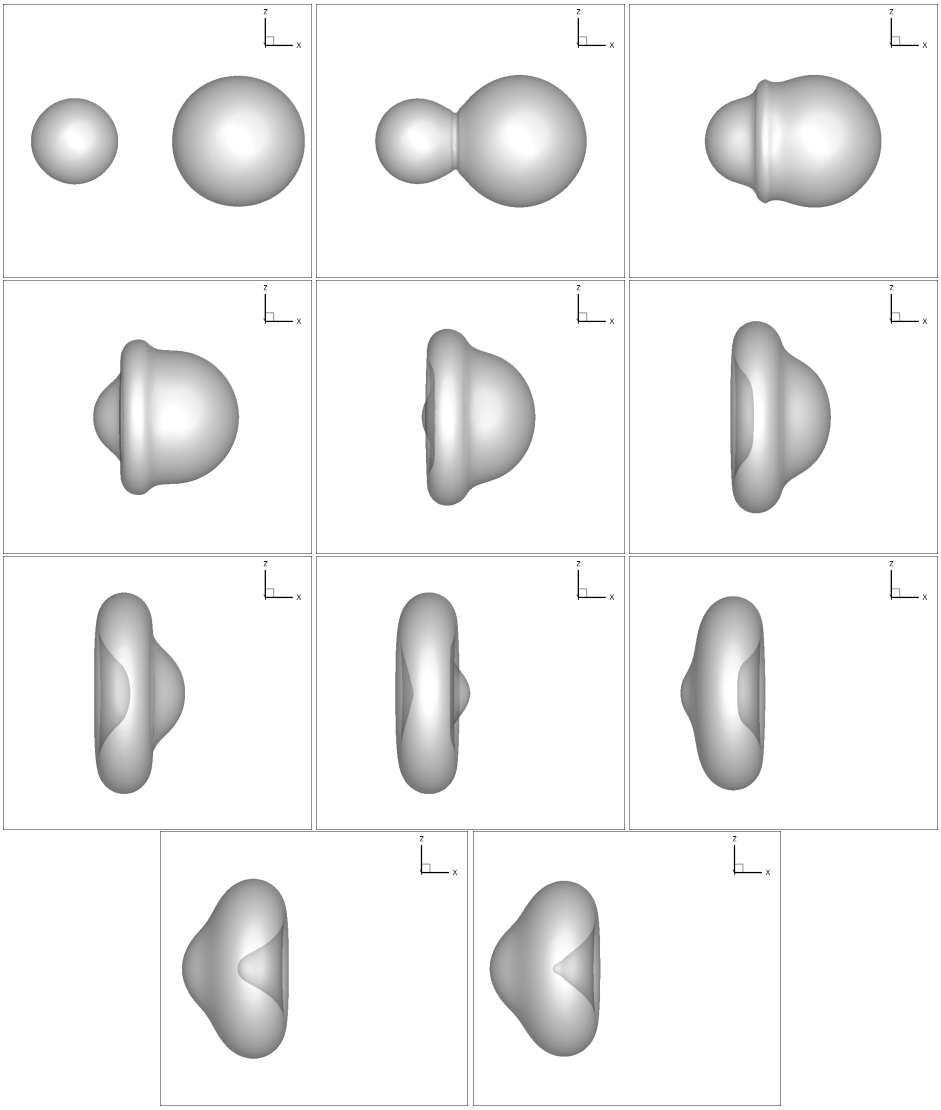


Figure 7.15: Before air entrapment, simulation of the collision We $= 30$ from the experiment of Ashgriz and Poo (1990)

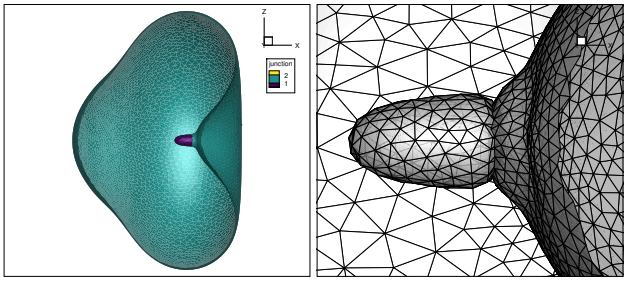


Figure 7.16: Air entrapment, simulation of the collision We $= 30$ from the experiment of Ashgriz and Poo (1990). The two pictures are obtained by clipping the Lagrangian mesh in the $\mathbf{Y}$ direction. On the left, the identified regions are coloured. On the right, the trapped bubble intersects the drop after the smoothing step, as no SGS model is employed.

Part IV

Conclusions and Perspectives

Conclusions and perspectives

A Front-Tracking method was presented. The influence of the remeshing methods and criteria on the approximation of the curvature, position and volume was appreciated with two unsteady analytical test cases. The expanding and shrinking test case is simple: with a spherical shape, the exact curvature is constant at each timestep, and the mesh is almost homogeneous. The second test case, with a surface deformed in a radial velocity field, allowed us to test the adaptive remeshing criteria along with the three edge splitting methods on a surface with spatially varying curvature. The use of polynomial approximation for the edge splitting and collapsing generally improves the approximation of curvature, position and volume. We observed that even though the adaptation to curvature was interesting with exact velocities, it was less beneficial with interpolated velocities when the velocity field presents strong variations. This is understandable since the numerical curvatures are used in the remeshing criterion and position errors accumulate.

The developed Front-Tracking method was then compared with a VOF method to appreciate the influence of the coupling of the Lagrangian mesh with the Eulerian grid. It was observed that the formulation of the average viscosity has a significant impact on the shape of the bubble. Likewise, the shape of the bubble changes noticeably depending on the resolution of the Lagrangian mesh.

A method for coalescence and a method for breakup were proposed, based on geometrical criteria. These two methods are activated by distance criteria and rely only on the interface mesh, without resorting to a supplementary grid. These methods were employed on numerical and experimental configurations from the literature to appreciate their robustness and performances.

When the velocity field is not accurate enough, the adaptive remeshing to the curvature cannot be employed, otherwise unphysical features are tracked and the quality of the simulation is not improved.

Concerning the resolution of the Navier-Stokes equations, using a Sub-Grid Scale model, Adaptive Mesh Refinement, a GFM or a two-fluid formulation like in Tavares et al. (2021) would provide the Front-Tracking method with a more physical velocity field. The computation of the surface tension may be improved, for example with an Integral Formulation without Peskin's distributions, i.e. by computing the intersections of the Lagrangian mesh with the Eulerian grid, like in Popinet and Zaleski (1999). The parallelization of the Front-Tracking method will be necessary to go further in convergence studies for the rising bubble case.

As for topological changes, the proposed methods only work for simple cases. The classification of the liquid structures needs to be improved to decide which structures should be merged or separated. Robustness is an issue when only using the explicit representation of the interface. To the best of our knowledge, other Front-Tracking methods in the literature resort to a supplementary grid for complex topological changes.

Part V

Appendices

A Comparison of four Runge-Kutta schemes with a three-dimensional deformation of a sphere

LeVeque (1996) proposed a test case where a sphere deforms in an analytical velocity field described in eq. A.1 where $T = 3$, and then returns to its initial position and shape. The sphere of radius $R = 0.15$ is initially centered at $(0.35, 0.35, 0.35)$ in a unit domain: $[0, 1]^3$. We consider the position error in l_∞ norm, with $r_i^e = R = 0.15$. First, we study the time integration schemes (subsection 2.5.2, on page 15) with exact velocities, without remeshing, with a mesh of 32^3, and a Lagrangian mesh of 1280 faces (fig. A.1). For this velocity field, the RK1, RK2, SSP-RK3 and RK4 have respectively 1st, 3rd, 3rd and 5th orders in time, as long as machine error does not come into play. The position errors are then evaluated for the transport with spatial interpolation (PERM, fig. A.2). Here the intermediate velocities are not known, so 2nd order Adams-Bashforth extrapolations are used, which degrades the order of convergence of the RK2, SSP-RK3 and RK4 methods.

$$\begin{aligned} u(x,y,z,t) &= 2sin^2(\pi x)sin(2\pi y)sin(2\pi z)cos(\pi t/T) \\ v(x,y,z,t) &= -sin(2\pi x)sin^2(\pi y)sin(2\pi z)cos(\pi t/T) \\ w(x,y,z,t) &= -sin(2\pi x)sin(2\pi y)sin^2(\pi z)cos(\pi t/T) \end{aligned} \tag{A.1}$$

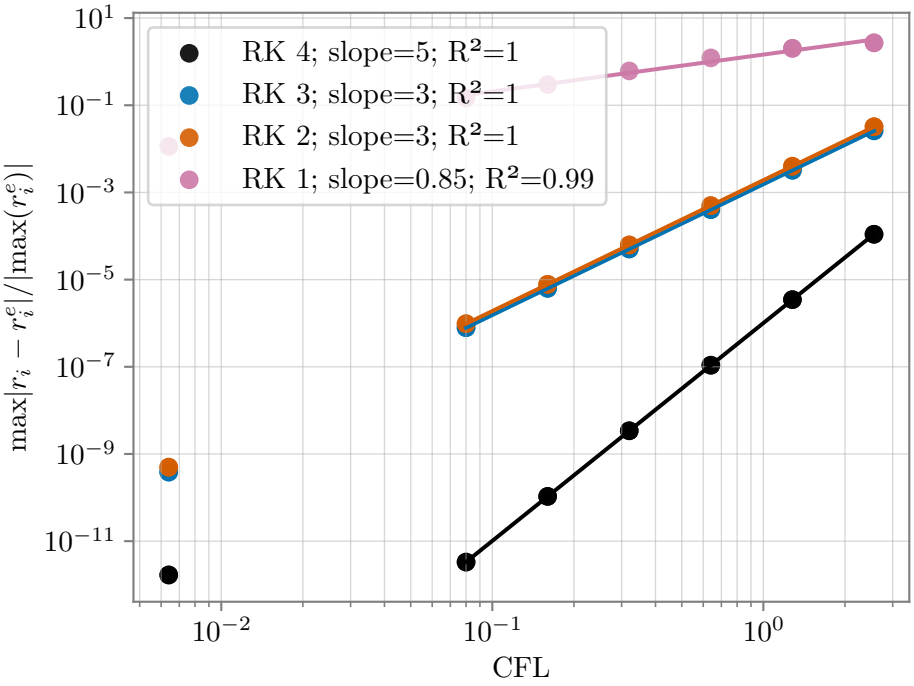


Figure A.1: Comparison of Runge-Kutta schemes, exact velocity, CFL based on a mesh of 32^3, and a reference velocity $U = 2$

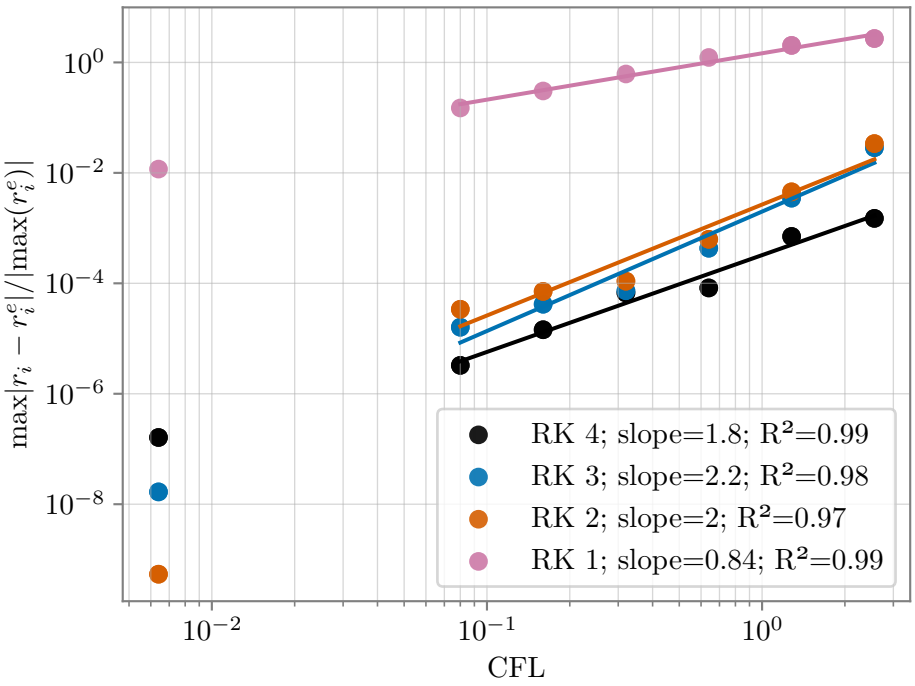


Figure A.2: Comparison of Runge-Kutta schemes, PERM interpolation, mesh of 32^3, CFL based on a reference velocity $U = 2$

B Computation of the volume fraction: Ray-Casting method

We describe the method employed to compute the volume fraction and the phase indicator.

The Ray-Casting method is used to compute the phase indicator χ and the volume fraction C in the scalar control volume (i, j, k) of the Eulerian grid, denoted $\chi(i, j, k)$ and $C(i, j, k)$. For every scalar node (i, j, k), we wish to know if it belongs to the interior of he region enclosed by the Lagrangian mesh or its exterior. This problem is known as the "Point-in-Polygon" problem, an overview of the methods employed to solve it can be found in Ogayar et al. (2005). The Ray-Casting method is based on the Jordan curve theorem. According to the Jordan Curve theorem, a plane simple (which does not intersect itself) closed curve divides the points of the plane into two distinct domains. By considering two points A and B, a ray is cast from A to B. If the number of intersections of segment $[AB]$ is even, points A and B are in the same phase, A and B are in different phases otherwise.

In our case, the triangle mesh separates the computation domain into two phases. For a given node (i, j, k), which we denote A, we choose a point B outside of the domain (whose phase is set according to the boundary conditions). By computing the number of intersections between the ray $[AB]$ and the triangle mesh, we can tell if the points A and B are in the same phase or not. It is however not necessary to examine all triangles for each node.

Localization of the triangles. The triangles of the Lagrangian mesh can be located in each scalar control volumes, and we may test only the triangles belonging to the scalar control volumes crossed by the ray. With a cartesian Eulerian mesh, the algorithm can be further improved by casting rays in a fixed direction, such as $\mathbf{X}$ (a fixed direction in the Cartesian coordinate system, aligned with the Eulerian grid) (Sarthou et al., 2011). Instead of computing the node values independently, the nodes are examined in rows, aligned in the $\mathbf{X}$ direction, so that the number of intersections n_{int}can be simply incremented in successive cells. This reduces the number of examined triangles, because a node $(i + 1, j, k)$ would require to examine the same triangles as (i, j, k), with the triangles located between them in addition.

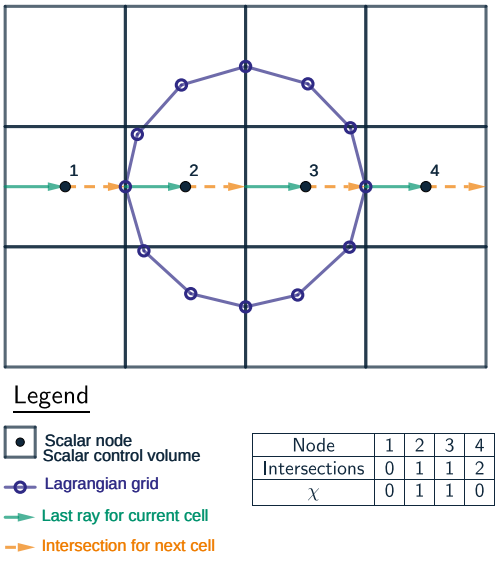
Computation of the phase indicator. Algorithm 5 advances in a fixed direction aligned with one of the grid axes such as $\mathbf{X}$. The function *Intersection* returns true if the segment $[AB]$ intersects the triangle. Figure B.1a illustrates the Ray-Casting method for a row of scalar control volumes in the direction of the ray. The triangles are located in the scalar control volumes, so the intersection tests are performed in two steps. The first test examines a segment from the left YZ face (in the $\mathbf{X}$ direction) of the scalar control volume to the scalar node. A second ray is cast from the scalar node to the right face. The intersections of this second ray are counted for the next cells.

Computation of the volume fraction with a scalar control volume refinement. The Ray-Casting method gives a binary value: the phase indicator χ. By subdividing the scalar control volume and applying the Ray-Casting method to the subdivisions, the volume fraction can be approximated (fig. B.1b). The volume fraction is approximated by a weighting with V_c the volume of the scalar control volume and V_i and χ_i the volume and phase indicator of its ith subdivision:

$$C = \frac{1}{V_c} \sum_{i \in subdivisions} V_i \chi_i \tag{B.1}$$

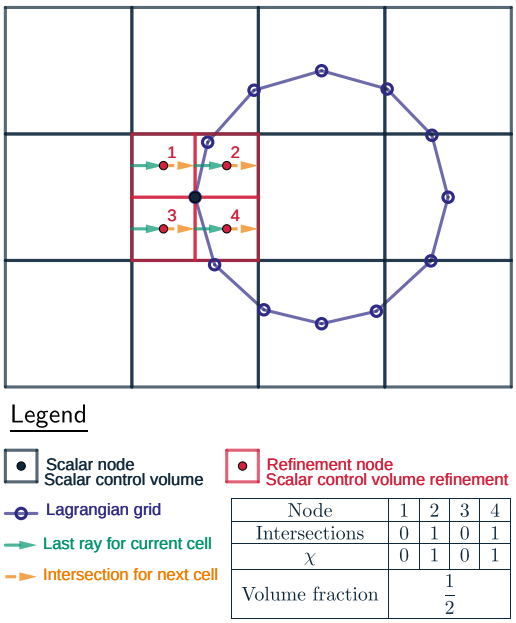
Algorithm 5: Ray-Casting

```
for k = startz,endz do
    for j = starty,endy do
        n_int = 0
        for i = startx,endx do
            for triangle in scalar control volume(i,j,k) do
                if Intersection then
                    n_int = n_int + 1
            end
            if n_int is even then
                χ(i, j, k) = 0
            else
                χ(i, j, k) = 1
            end
        end
    end
end
```



Node	1	2	3	4
Intersections	0	1	1	2
χ	0	1	1	0

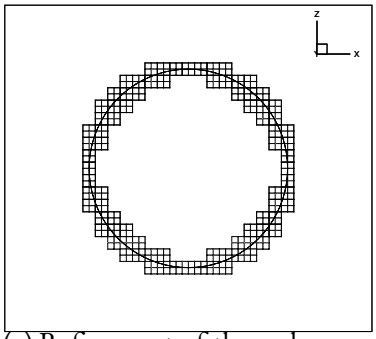
(a) Computation of the phase indicator



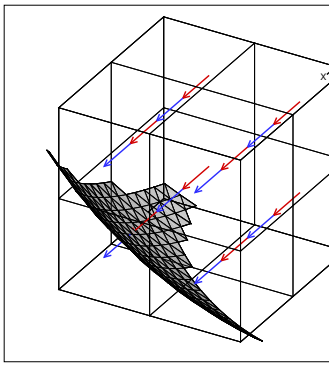
Node	1	2	3	4
Intersections	0	1	0	1
χ	0	1	0	1
Volume fraction	$\frac{1}{2}$			

(b) Computation of the volume fraction

Figure B.1: Ray-Casting



(a) Refinement of the scalar control volumes, slice view



(b) One level of refinement, the rays illustrated in the figure are used for a more precise evaluation of the volume fraction

Figure B.2: Refinement of the scalar control volume around the interface

The scalar control volumes are subdivided when triangles are located in the vicinity, as illustrated by the slice view in fig. B.2a where one level of refinement was employed. Subdivisions are only used in the vicinity of the interface, where the count of located triangles is not null. The triangles are then located on a finer level : on the subdivisions, starting from the triangles already located in the scalar control volume. Figure B.2b illustrates the refinement of the scalar control volume in 3D. The Lagrangian mesh represented in gray is the part that was located on the scalar control volume to accelerate the computation of the intersections. The red ray in each subdivision leaves from the left face to the scalar node and the blue ray leaves from the scalar node to the right face. The volume fraction is only computed in the cells where the interface is located. Otherwise it has the same value as the phase indicator. In the example, there is one level of refinement so the scalar control volume is divided in 2^3 subdivisions and intersections are tested in each subdivision.

Intersection between a ray and a triangle. If the ray hits an edge or a vertex, the same intersection is counted more than once, and this produces an error in the volume fraction for the current cell and the following cells in the ray direction. We use the "watertight" intersection test from Woop et al. (2013) instead of other methods such as the well-known procedure presented in Möller and Trumbore (1997). The latter does not ensure consistency at edges and vertices because it chooses one vertex among the three forming one triangle and then compute the intersection based on this vertex. The problem is that a shared edge or vertex can be computed differently depending on the triangle. The method proposed by Woop et al. (2013) computes the three signed distances to the edges, which guarantees consistency. In our case, we not only require a watertight algorithm but we also need to count only once an edge and once a vertex if the ray hits exactly at an edge/vertex within the machine error. If such a case is encountered, our algorithm recasts a ray with an origin shifted in the $\mathbf{Y}$ and $\mathbf{Z}$ direction with a small distance ϵ.

Ray-Casting at the limits of the computation domain. Some changes are necessary when the interface is near the limits of the computation domain. The first ray in a scalar control volume starts at the limits of the scalar control volume. If some triangles were right at the limit of the computation domain, the Ray-Casting algorithm could miss some intersections. We need to count the intersections before the wall and we move the markers slightly outside the domain to detect the intersections. At the last scalar control volume we count intersections up to the

last subdivision, that is to say : we do not detect intersections between the center of the last subdivision and the wall. In our implementation, when an element is in range of a wall, defined by a tolerance, the marker is shifted outside of the domain so that the Ray-Casting algorithm detects the element. The marker is not shifted at the exact limit of the domain but slightly outside. For instance, when an element is near a wall, say the left wall $x = x_{min}$: when the X coordinates of the vertices of an element are inferior to a threshold, these X coordinates are modified to $x_{min} - \epsilon$ with $\epsilon = h \times 10^{-10}$ (h Eulerian grid spacing) for instance. The rays are cast at the center of the scalar control volume: for instance at $(x_{min}, y_{min}, z_{min})$. By shifting the markers slightly outside of the domain, no intersection is missed. When computing the volume fraction at the wall with the refined Ray-Casting, only the contributions of the subdivisions inside the computation domain are taken into account.

Influence of the number of subdivisions in the raycasting algorithm. The Vofi library (Bnà et al., 2016) uses a variable number of nodes in each direction from 4 to 20 to integrate an analytical expression of the interface. We compare our results with the Vofi library for the volume fraction of a drop. Among the different sources of errors, we may cite the approximation of the surface by triangles, the limited number of rays and their position with regards to the interface.

The number of refinements influences the error on the volume fraction. With n levels of refinement there are 2^n subdivisions in each direction. The error on the volume fraction for a bubble of radius $R = 0.25$ centered at $(0.5, 0.5, 0.5)$ in a cubic domain of dimensions $1 \times 1 \times 1$ is computed with the Vofi library (Bnà et al., 2016). The order of the error is nearly one for a 32^3 mesh (fig. B.3).

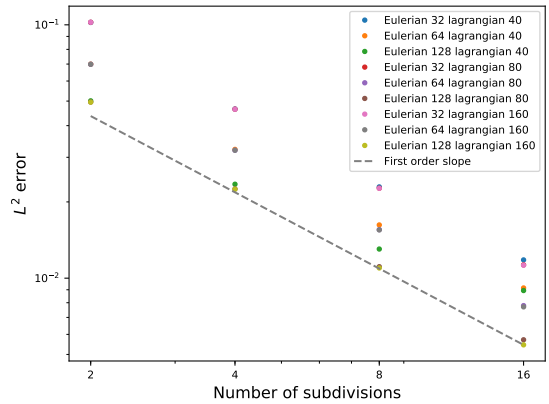


Figure B.3: l^2 error of the volume fraction compared to Vofi, in function of the number of subdivisions, for different Eulerian grids and Lagrangian meshes

C Sphere

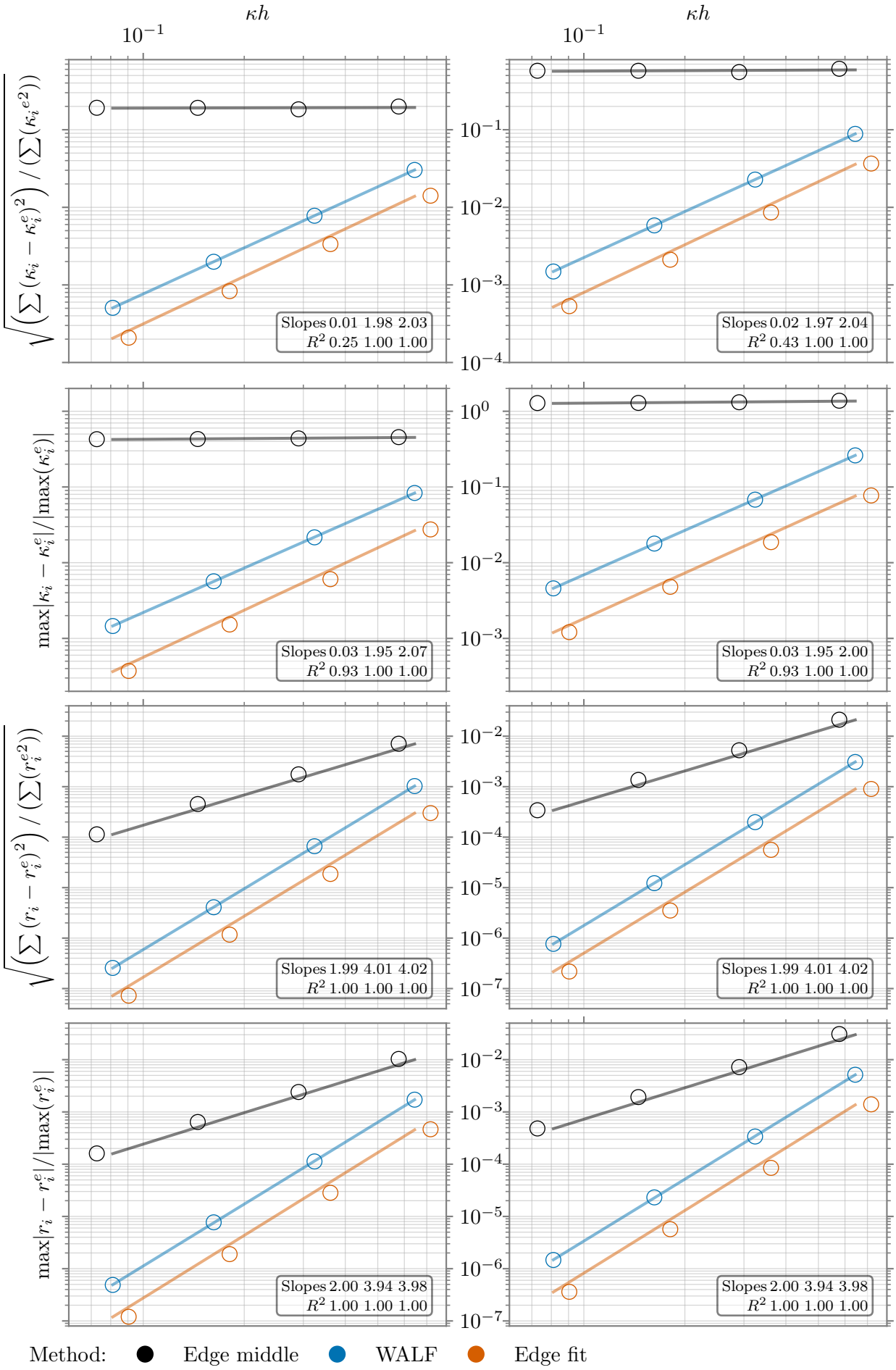


Figure C.1: Sphere case, no edge collapsing, no volume correction, errors at the middle (left) and end of the simulation (right), l_2 and l_∞ norms

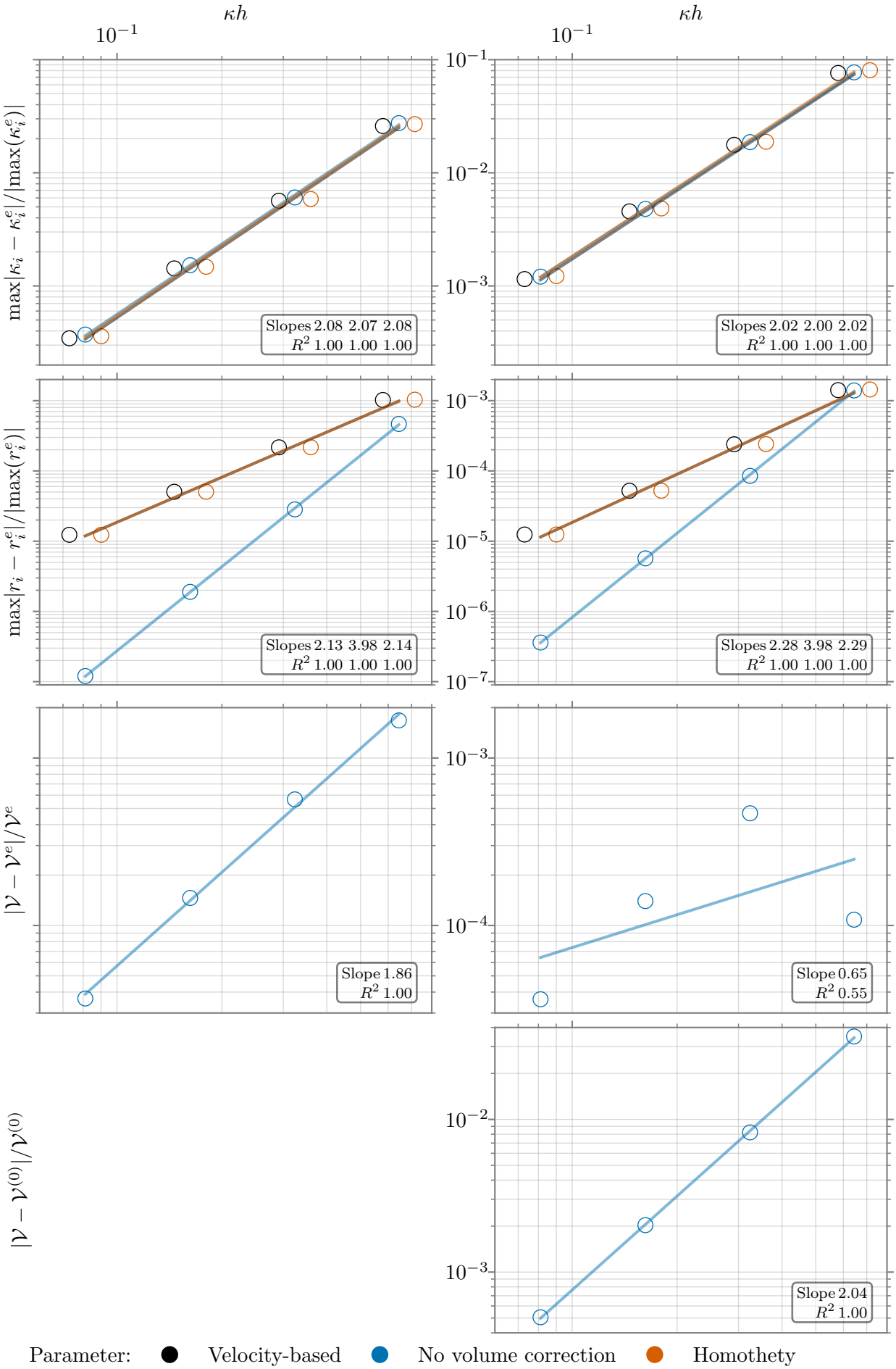


Figure C.2: Sphere case, without edge collapsing, with volume correction, errors at the middle (left) and end of the simulation (right), velocity-based volume correction (VBC), no volume correction and homothety (HR) see section 2.10, on page 18

D Radial

D.1 Distance computation with a discretized reference surface

In Garland and Heckbert (1997), the approximation error E_i sums a distance error from the vertices of the reference mesh $\mathcal{X}_n$ to the examined discretized surface $\mathcal{T}_i$ plus the error from the vertices of the examined mesh $\mathcal{X}_i$ to the reference discretized surface $\mathcal{T}_n$ with the minimal Euclidean distance dist (eq. D.1). In Hu et al. (2017), the one-sided Hausdorff distance is approximated (maximum of Euclidean distance).

$$E_i = \frac{1}{\operatorname{card}\mathcal{X}_n + \operatorname{card}\mathcal{X}_i}\left(\sum_{x\in\mathcal{X}_n}\operatorname{dist}^2(x,\mathcal{T}_i) + \sum_{x\in\mathcal{X}_i}\operatorname{dist}^2(x,\mathcal{T}_n)\right) \tag{D.1}$$

We could define a fine reference mesh with exact positions. Yet, if we want to compute a curvature error with the fine mesh, we would need to interpolate curvature on the mesh under study, based on the curvature computed at the surrounding vertices or employ a more general method, able to compute the curvature at any point on the discretized surface. For a vertex on the reference mesh, the closest point on the mesh under study does indeed not necessarily coincide with a vertex of the mesh. Supplementary figures are presented, with l_2 and l_∞ norms and in zones $\mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \mathcal{Z}_3$, $\mathcal{Z}_2$ and $\mathcal{Z}_3$.

D.2 Exact position

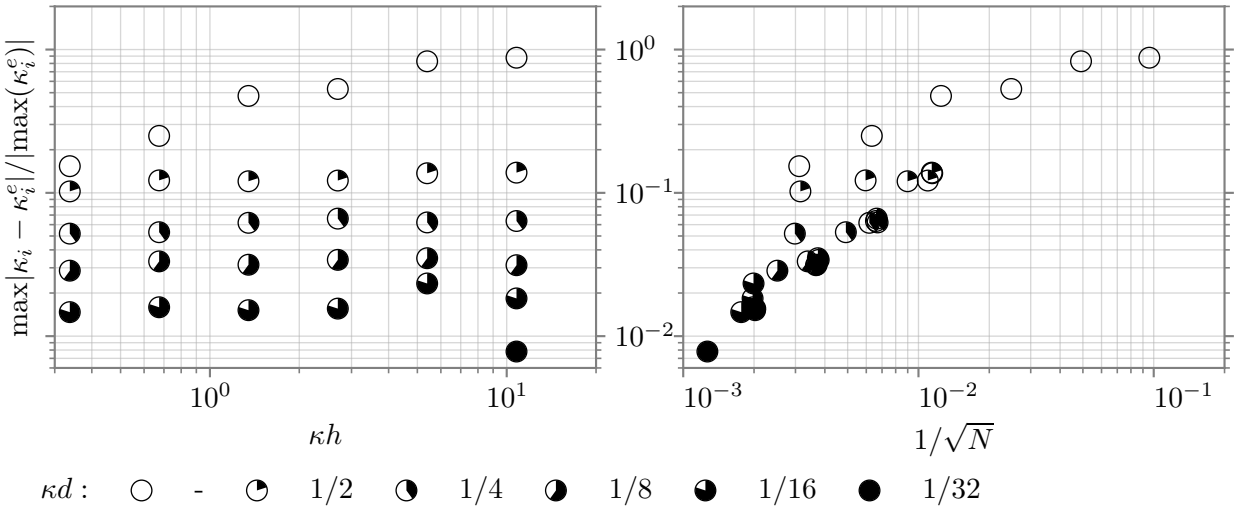


Figure D.1: Exact position, whole domain, l_2 norm in fig. 4.13

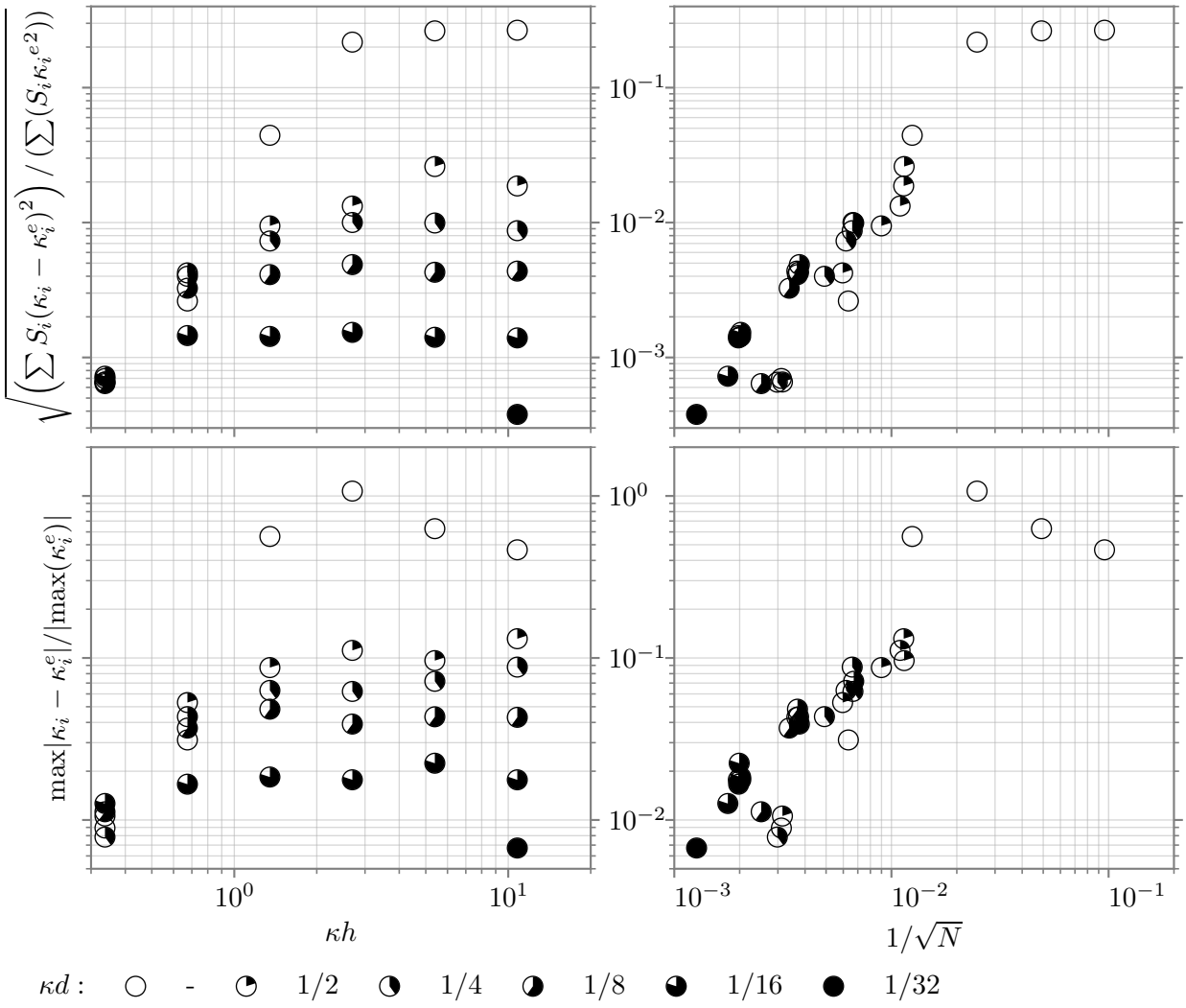


Figure D.2: Exact position, zone 3, whole domain in fig. 4.13

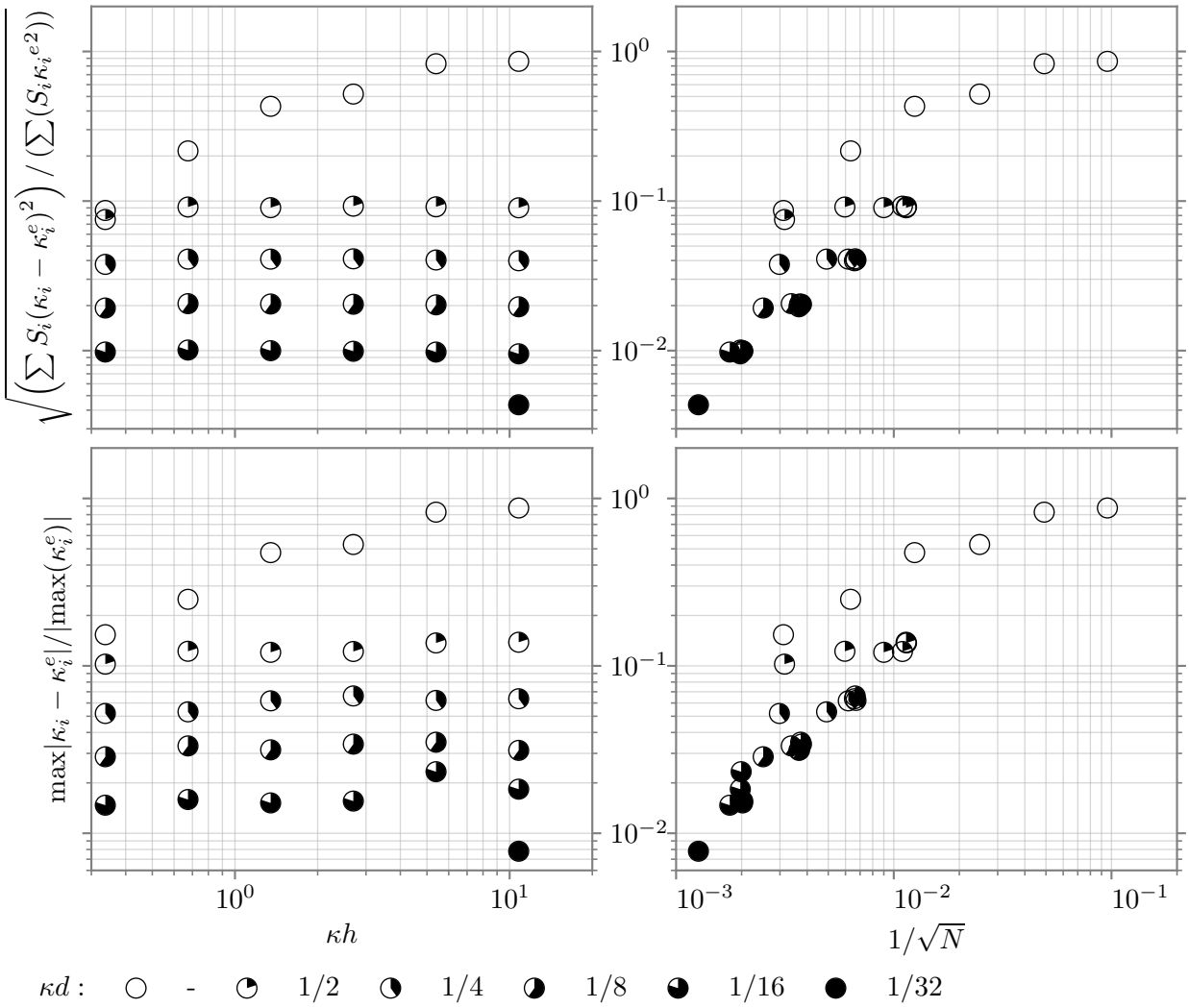


Figure D.3: Exact position, zone 2, whole domain in fig. 4.13

D.3 Influence of refinement diffusion

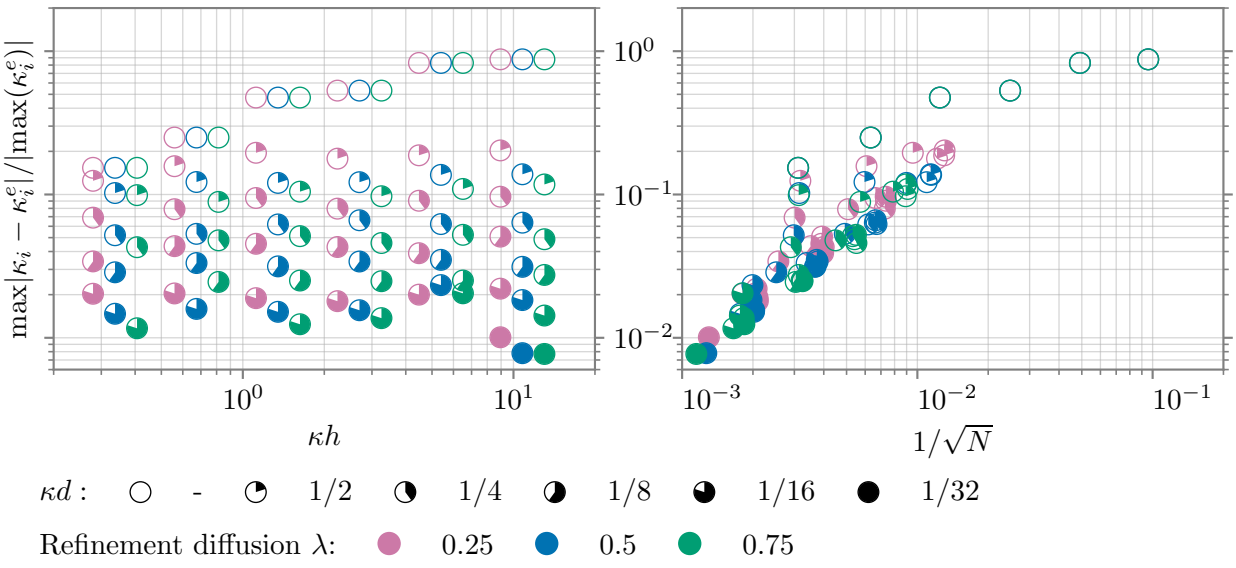


Figure D.4: Exact position, zone 4, influence of refinement diffusion, l_2 norm in fig. 4.15

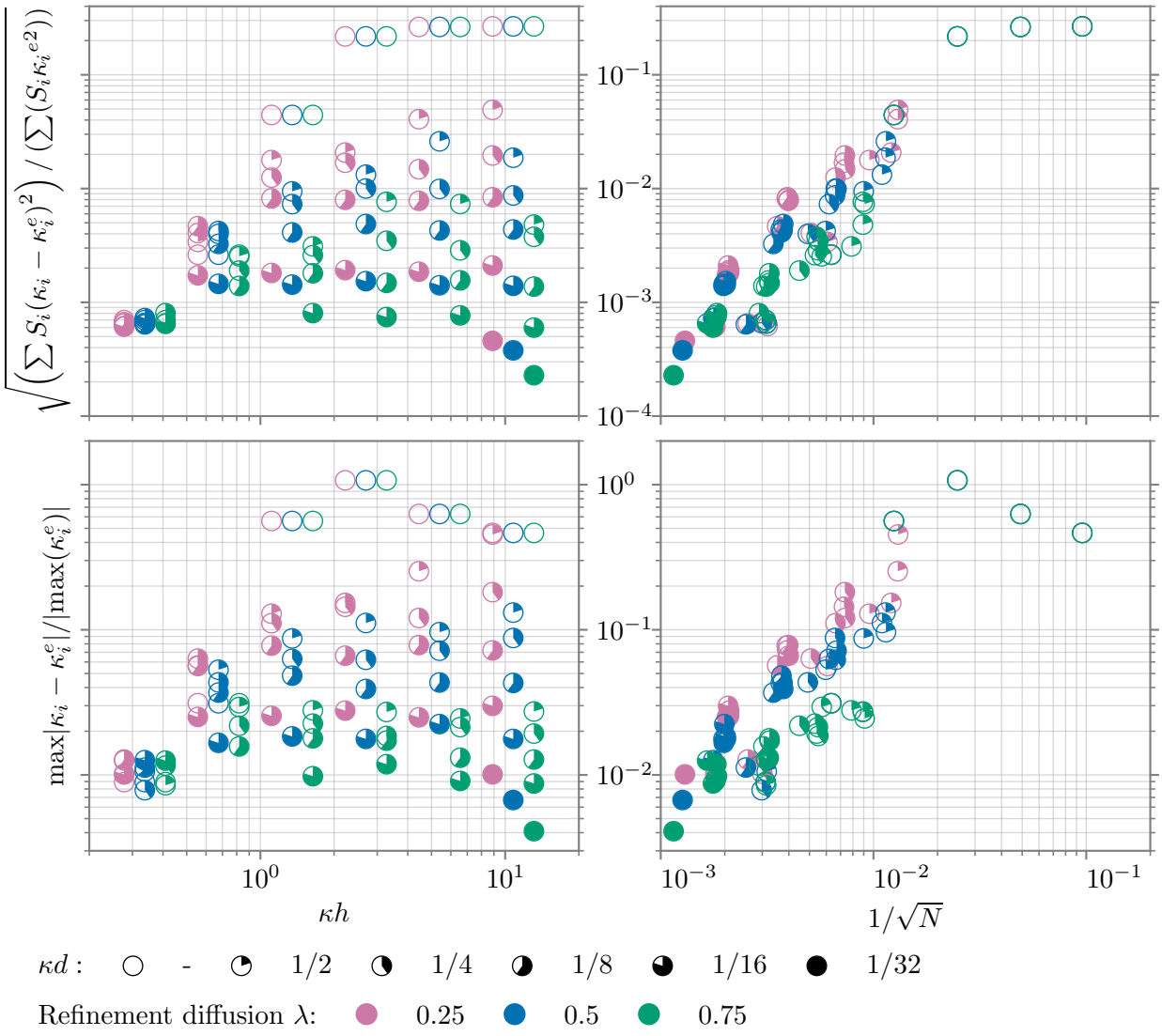


Figure D.5: Exact position, zone 3, influence of refinement diffusion, whole domain in fig. 4.15

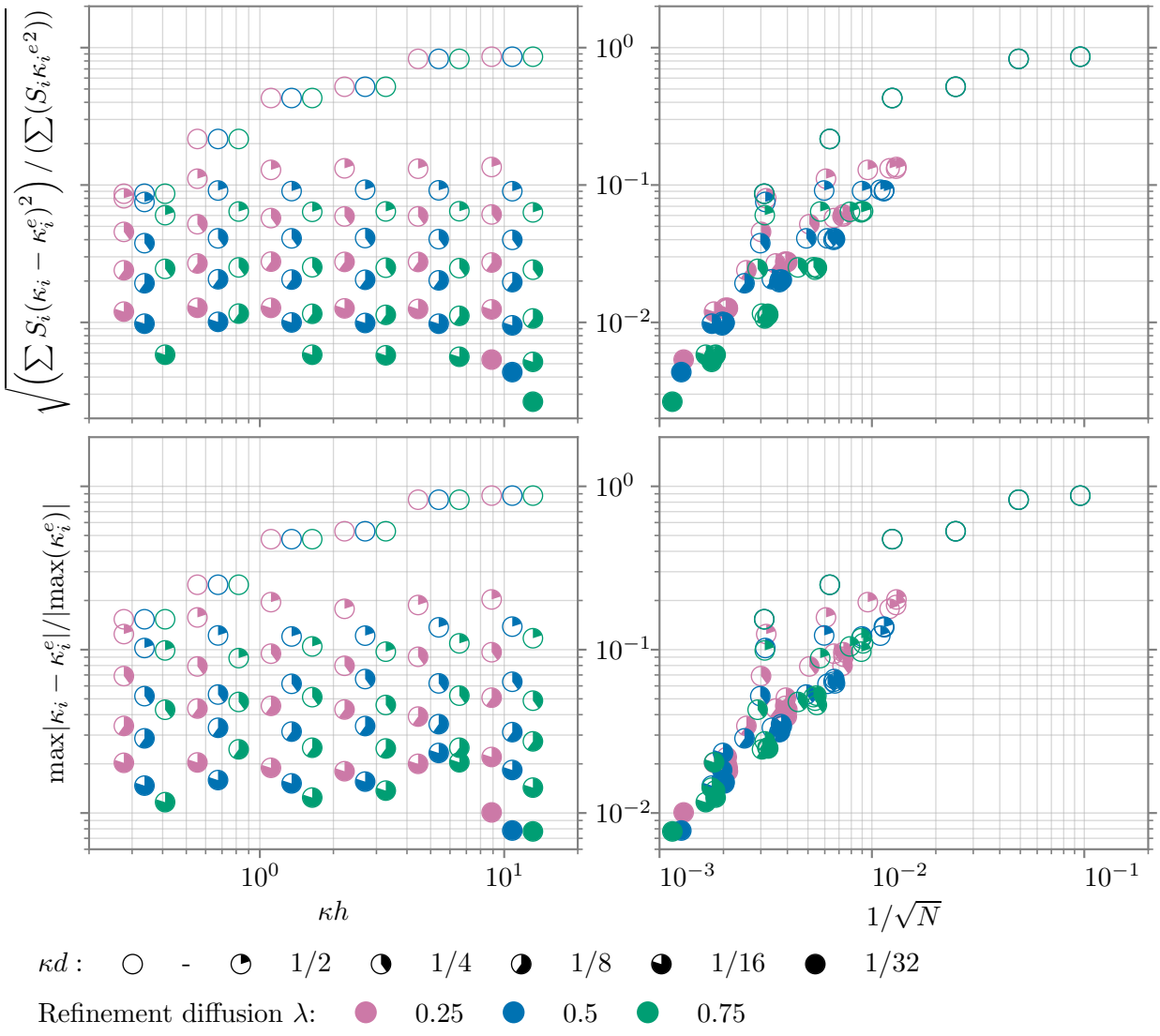


Figure D.6: Exact position, zone 2, influence of refinement diffusion, whole domain in fig. 4.15

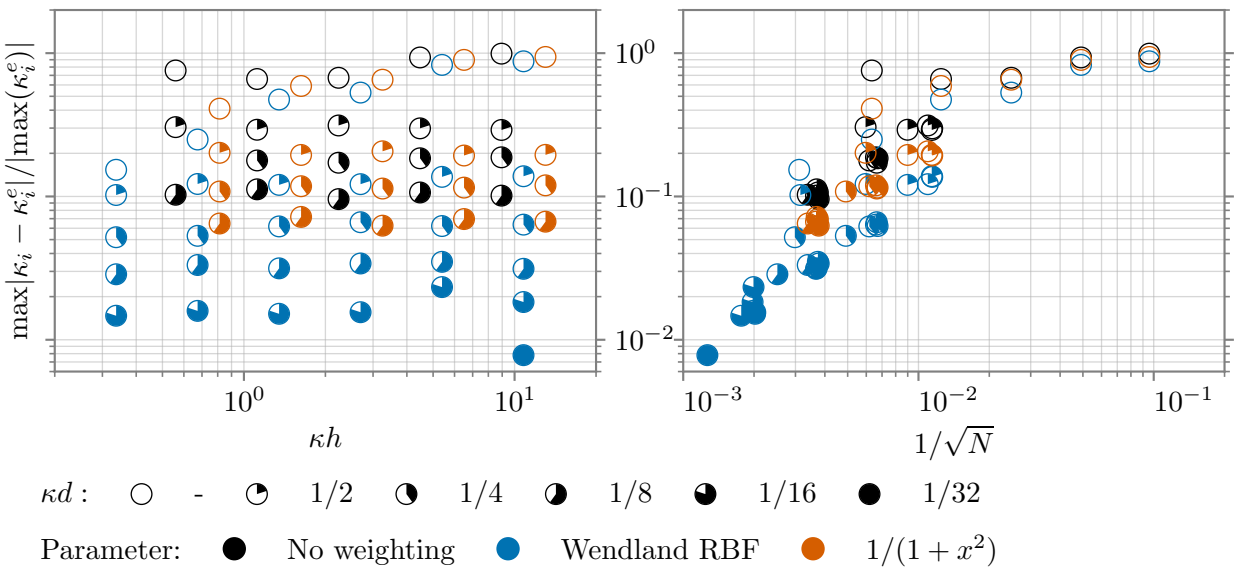


Figure D.7: Exact position, whole domain, influence of weighting, Wendland's RBF (Wendland, 1995), $R = 3R_{mean}/2$, l_2 norm in fig. 4.16 (on page 57)

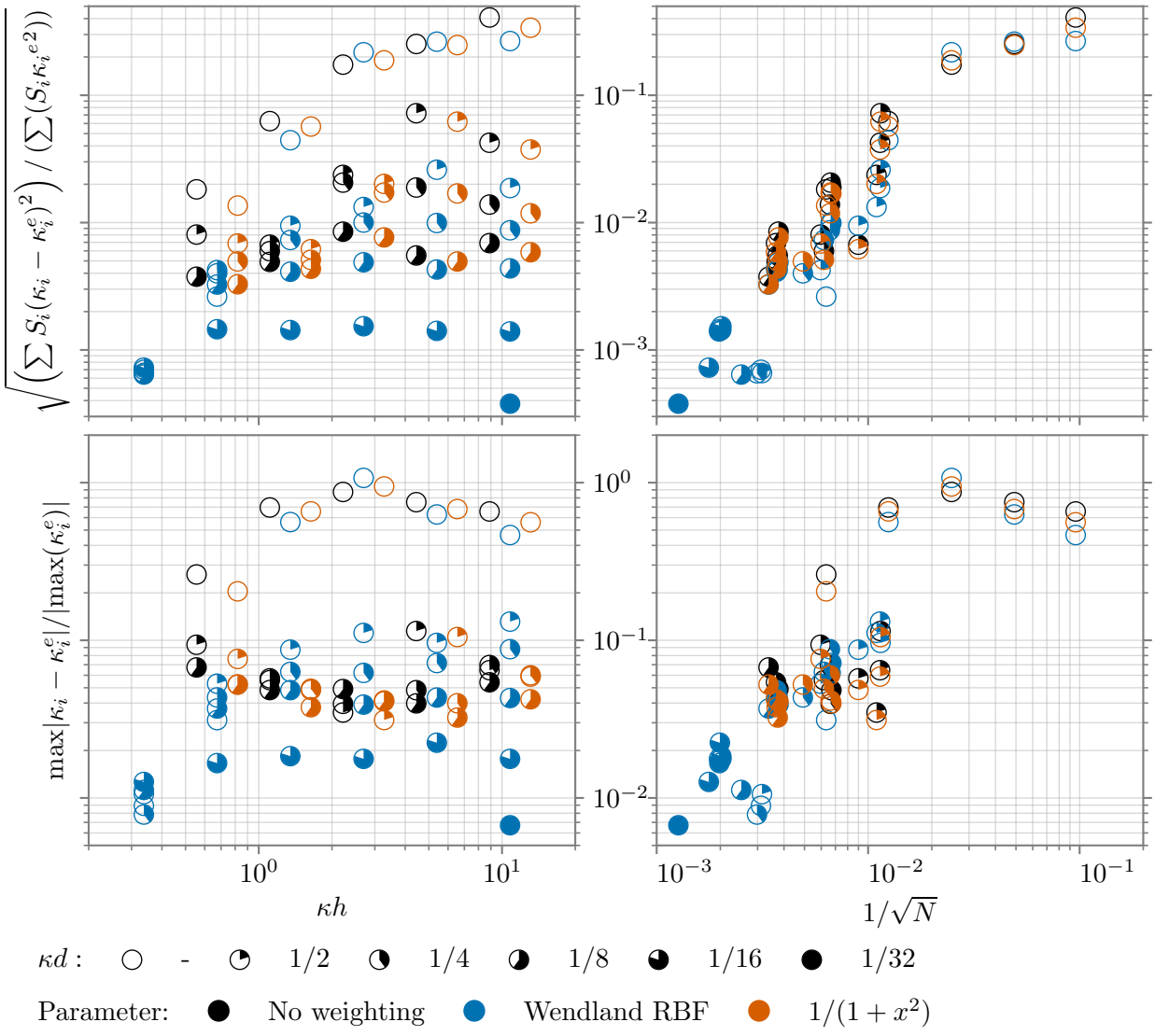


Figure D.8: Exact position, zone 3, influence of weighting, Wendland's RBF (Wendland, 1995), $R = 3R_{mean}/2$, whole domain in fig. 4.16 (on page 57)

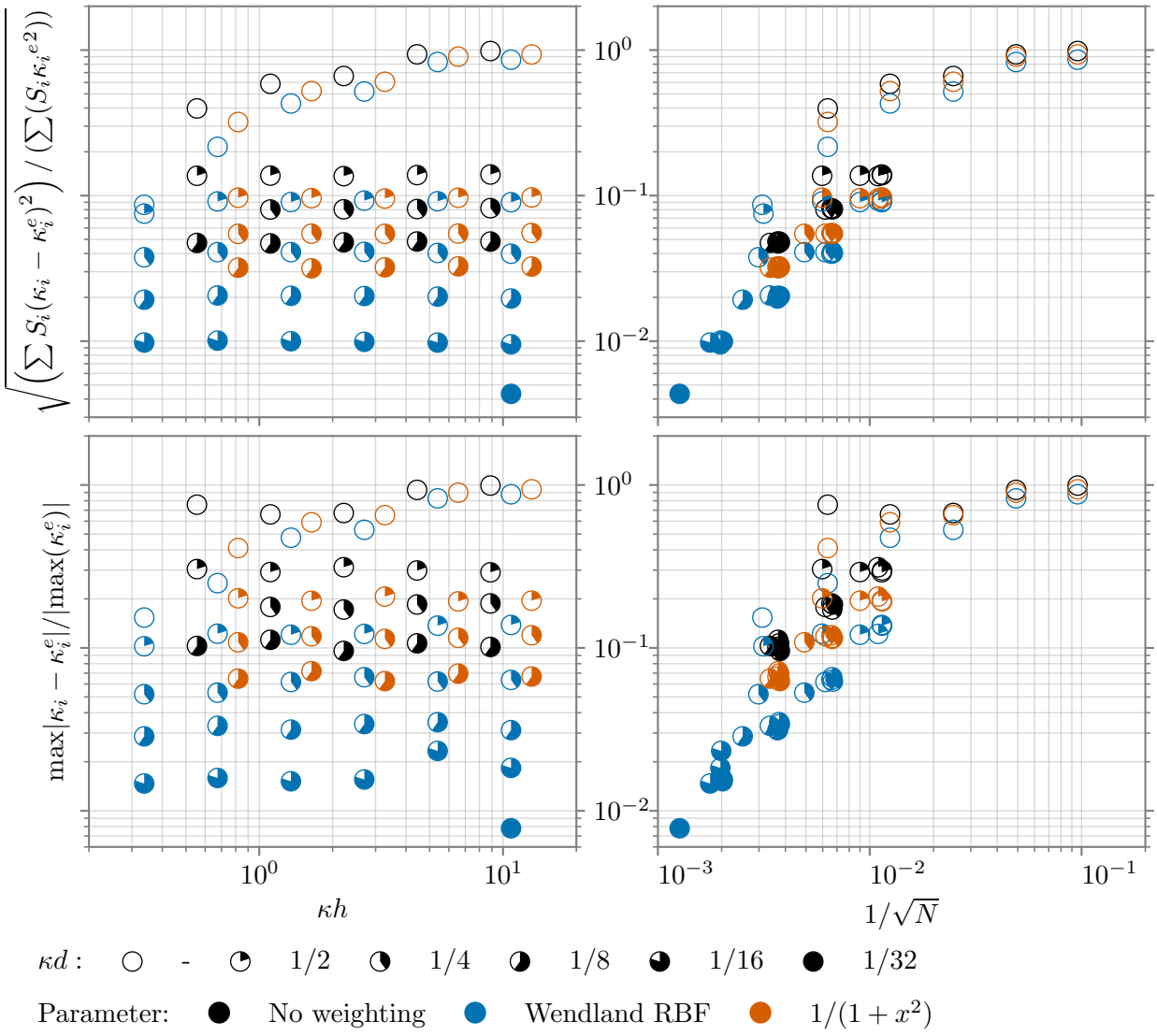


Figure D.9: Exact position, zone 2, influence of weighting, Wendland's RBF (Wendland, 1995), $R = 3R_{mean}/2$, whole domain in fig. 4.16 (on page 57)

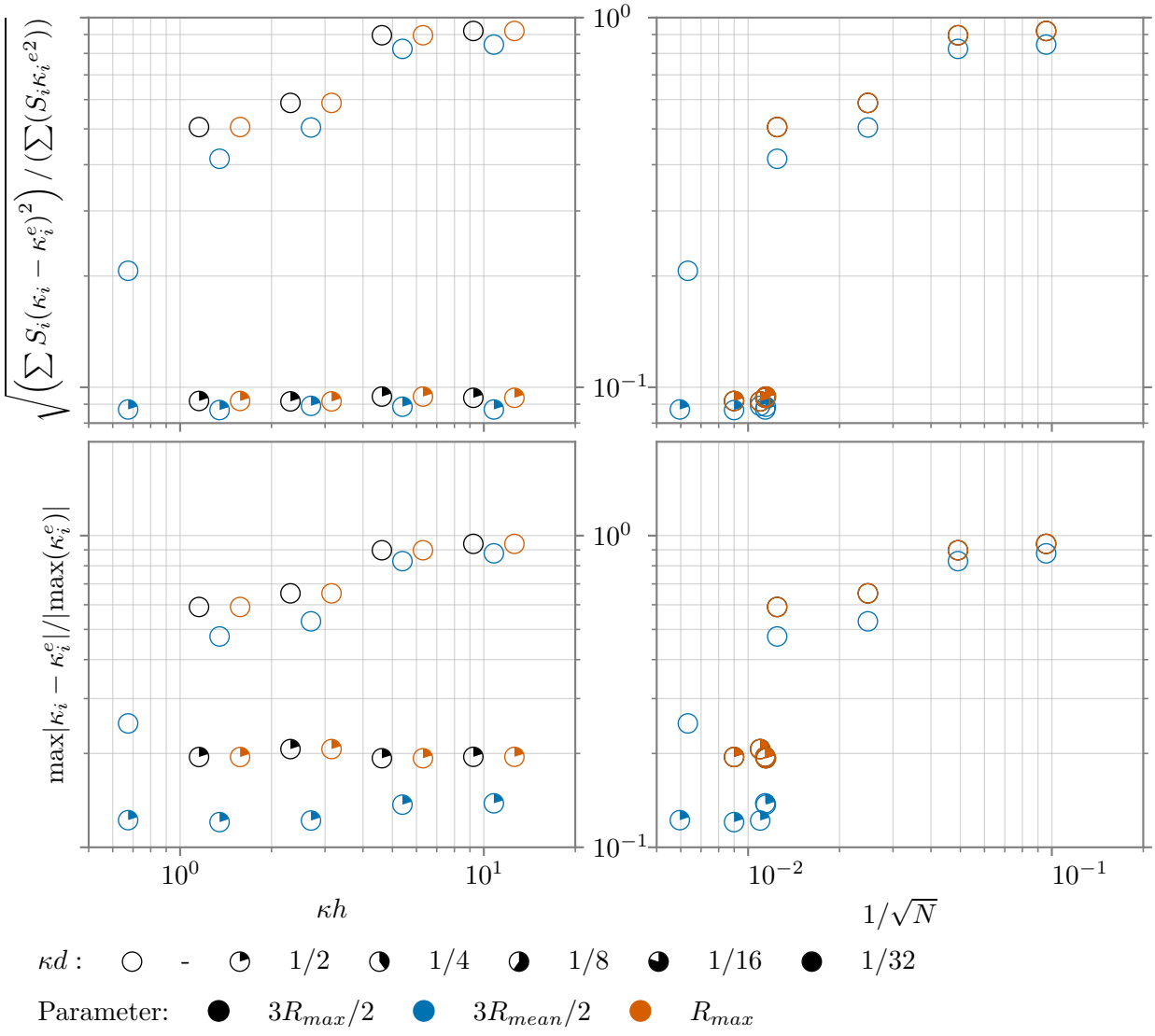


Figure D.10: Exact position, whole domain, influence of weighting, Wendland's RBF (Wendland, 1995), $R = 3R_{max}/2$, $R = 3R_{mean}/2$, $R = R_{max}$

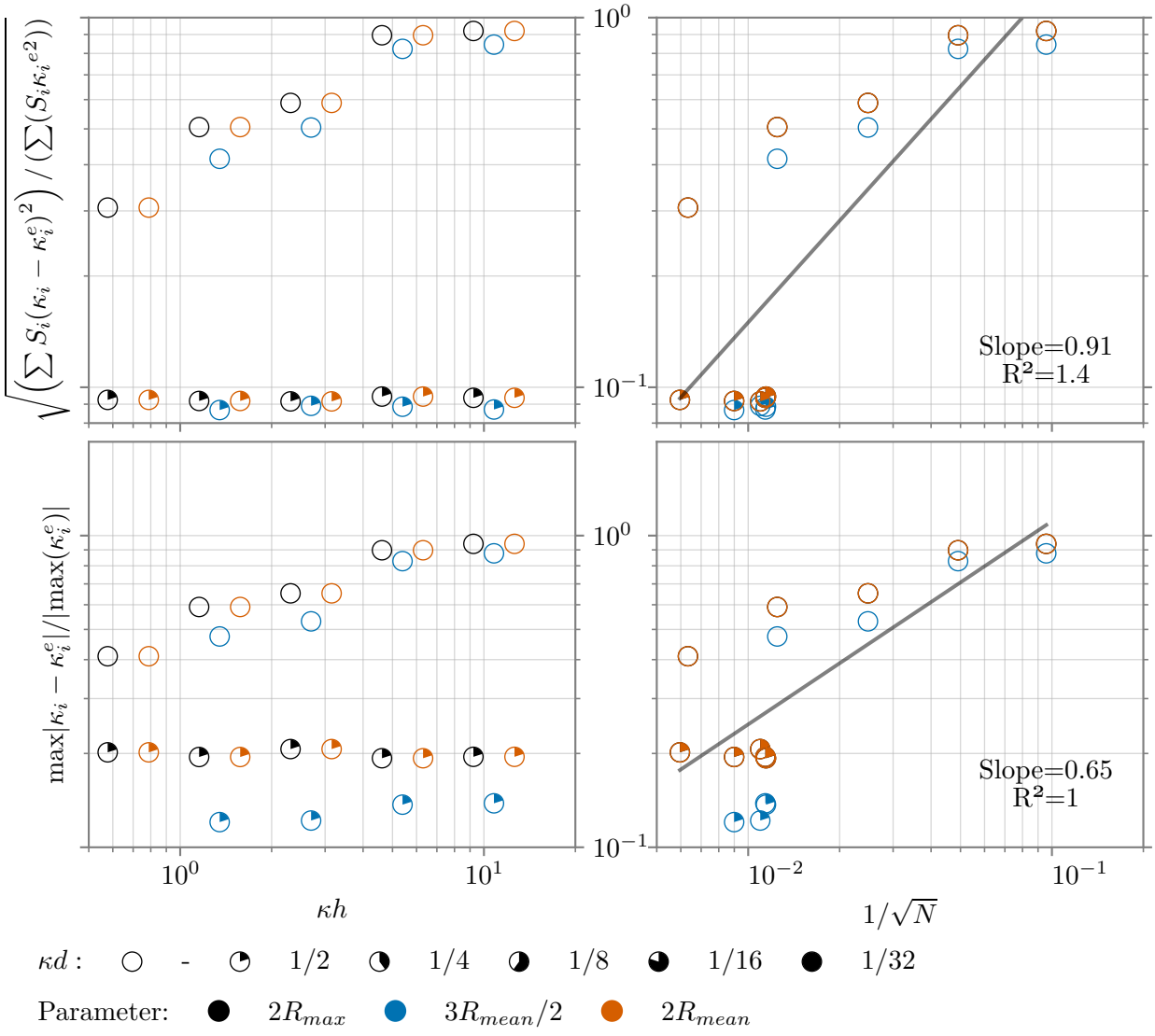


Figure D.11: Exact position, whole domain, influence of weighting, Wendland's RBF (Wendland, 1995), $R = 2R_{max}$, $R = 3R_{mean}/2$, $R = 2R_{mean}$

D.5 Influence of the definition of the reference position

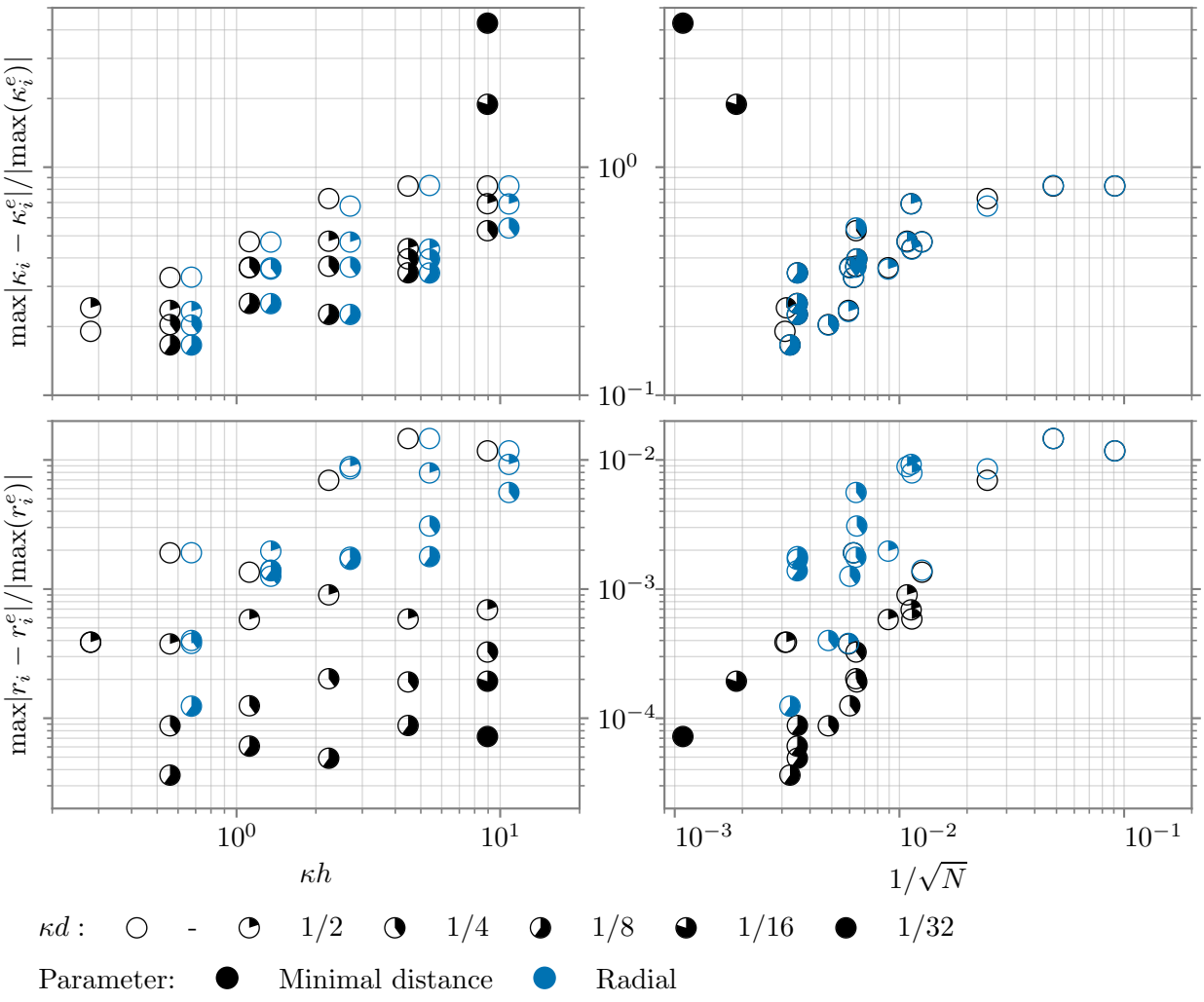


Figure D.12: Exact transport, whole domain, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), influence of the definition of the reference position, l_2 norm in fig. 4.17 (on page 59)

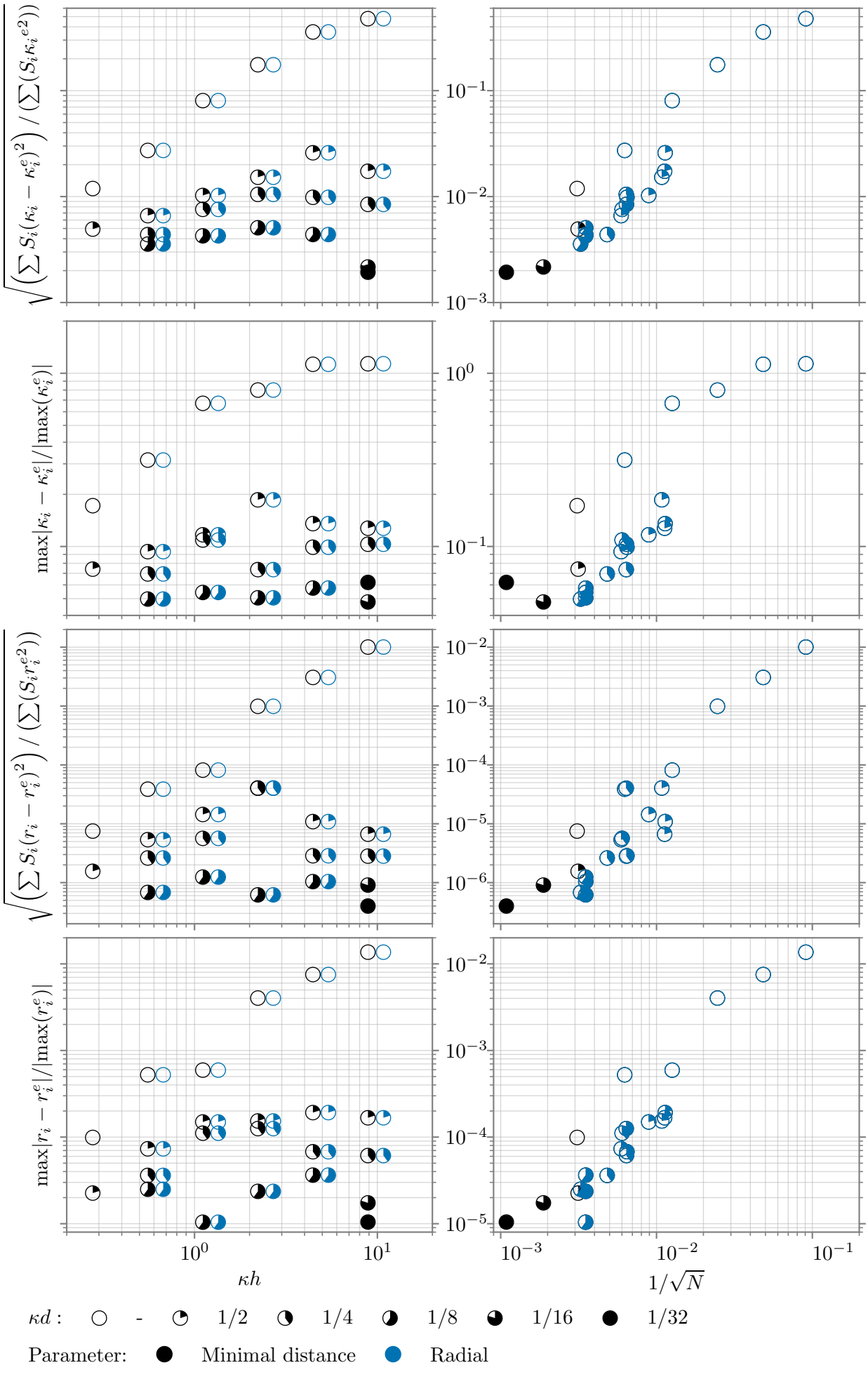


Figure D.13: Exact transport, zone 3, edge fit (Gorges et al., 2022), influence of the definition of the reference position, whole domain in fig. 4.17 (on page 59)

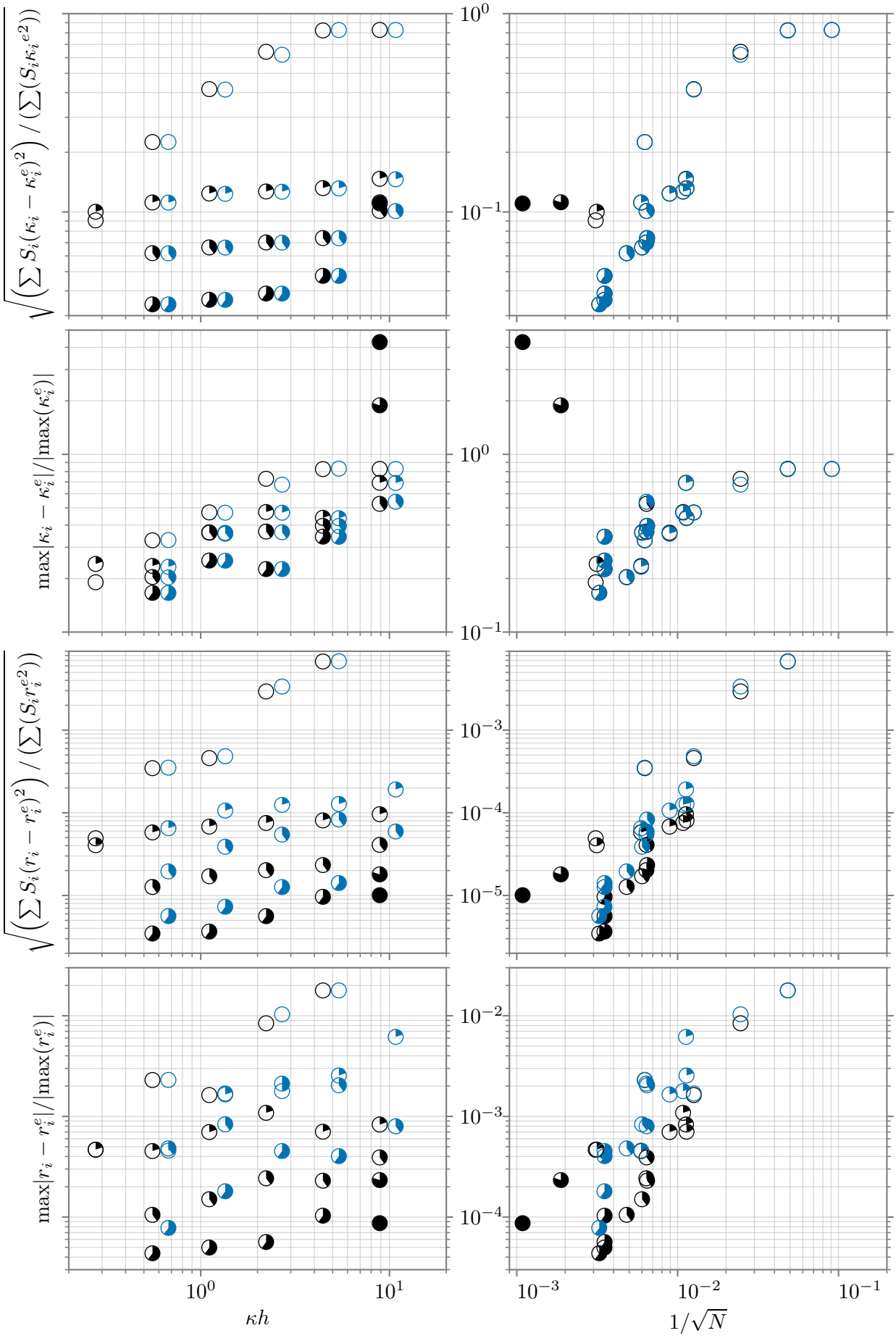


Figure D.14: Exact transport, zone 2, edge fit (Gorges et al., 2022), influence of the definition of the reference position, first two points ($\kappa d = 0$) removed (position error lower than 10^{-14}, because of the small number of points in zone 2 and those points and those points were presumably in the initial mesh)

D.6 Influence of weighting for the new vertex position

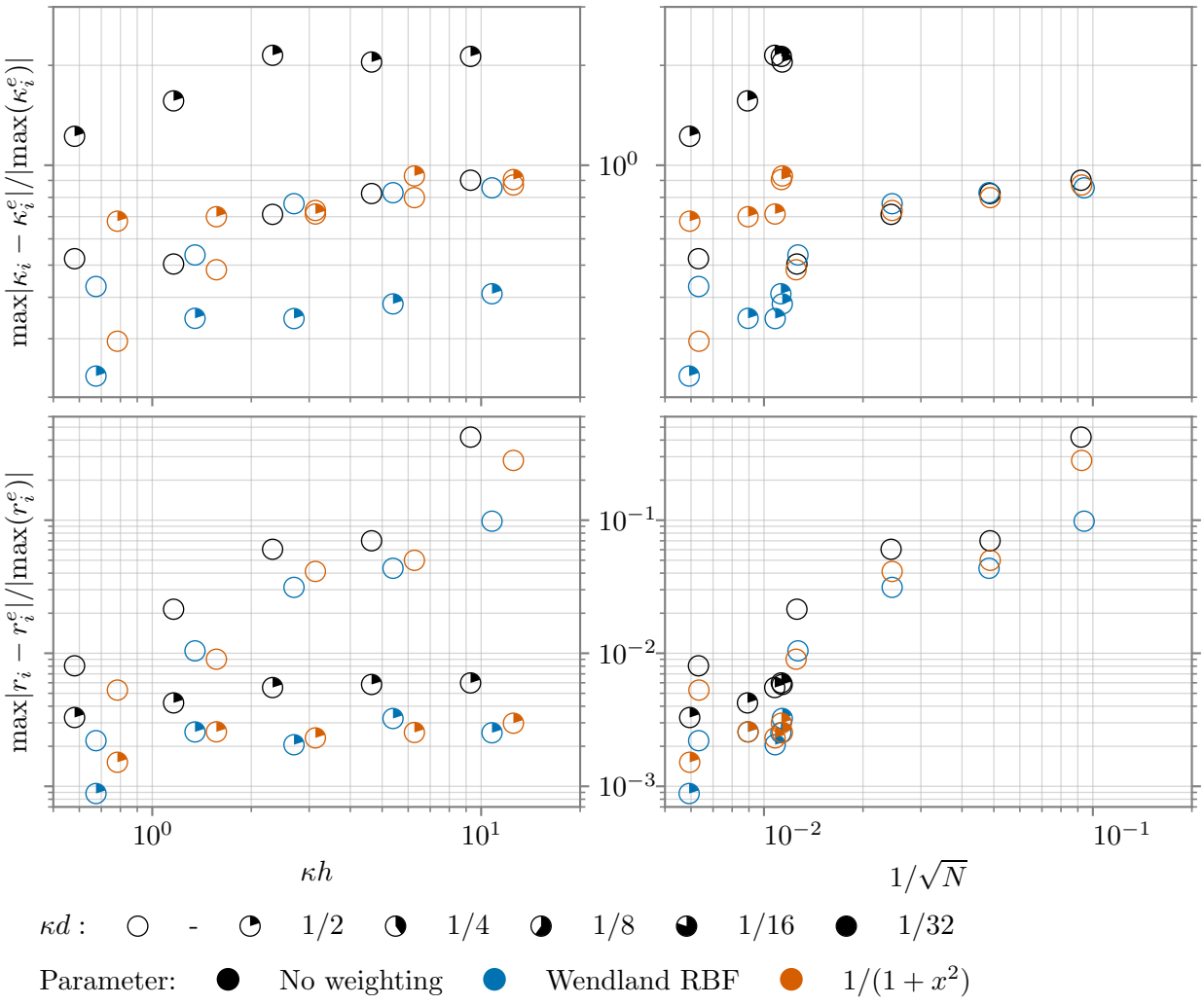


Figure D.15: Exact transport, whole domain, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), influence of the definition of the reference position, l_2 norm in fig. 4.18 (on page 60)

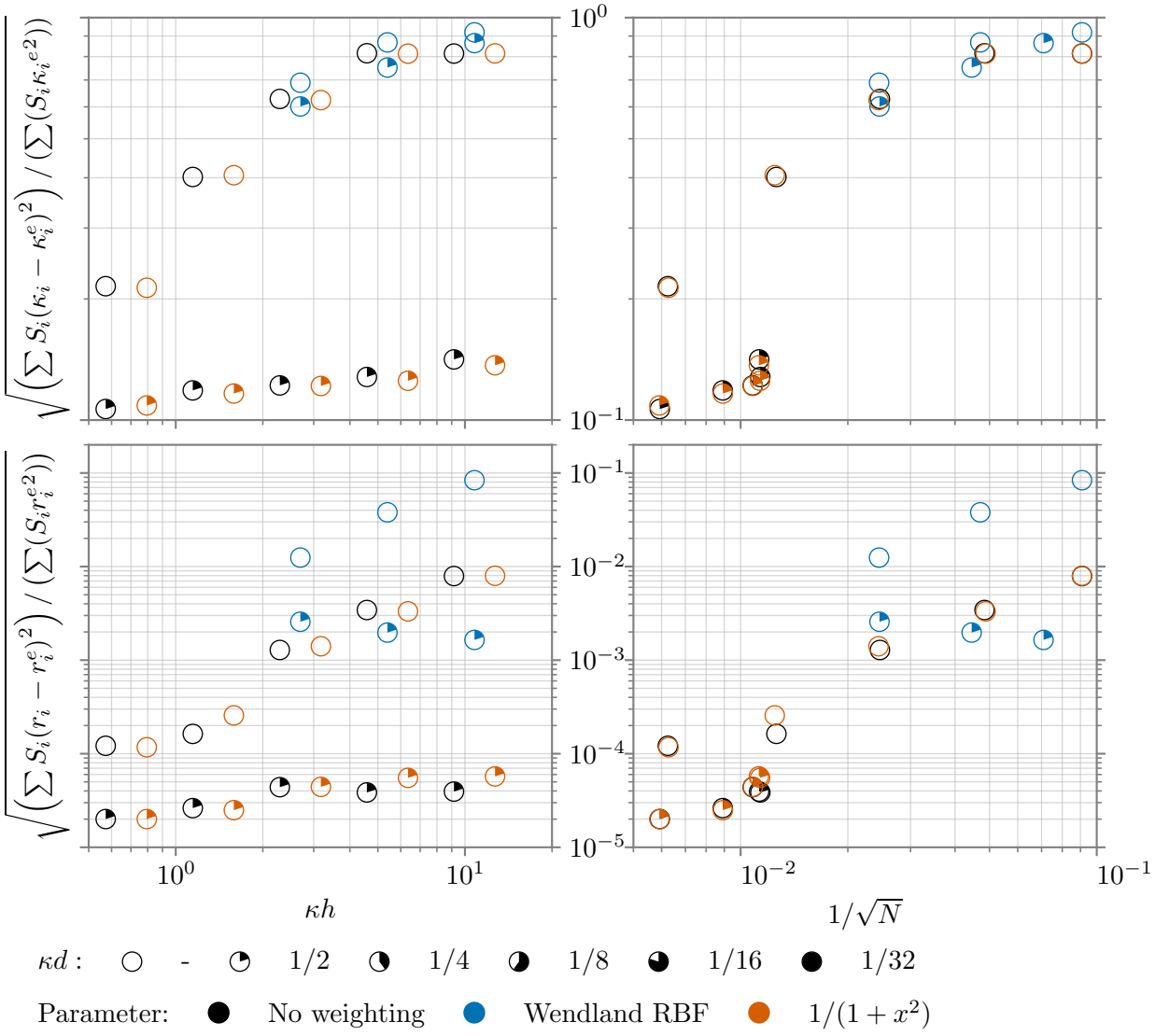


Figure D.16: Exact transport, whole domain, reference position with minimal distance, edge fit (Gorges et al., 2022), influence of weighting for the new vertex position

D.7 Comparison of edge refinement methods with radial reference position

Here we denote the reference values for the i-th vertex $r^e = r^{e,1}$ and $\kappa^e = \kappa^{e,1}$.

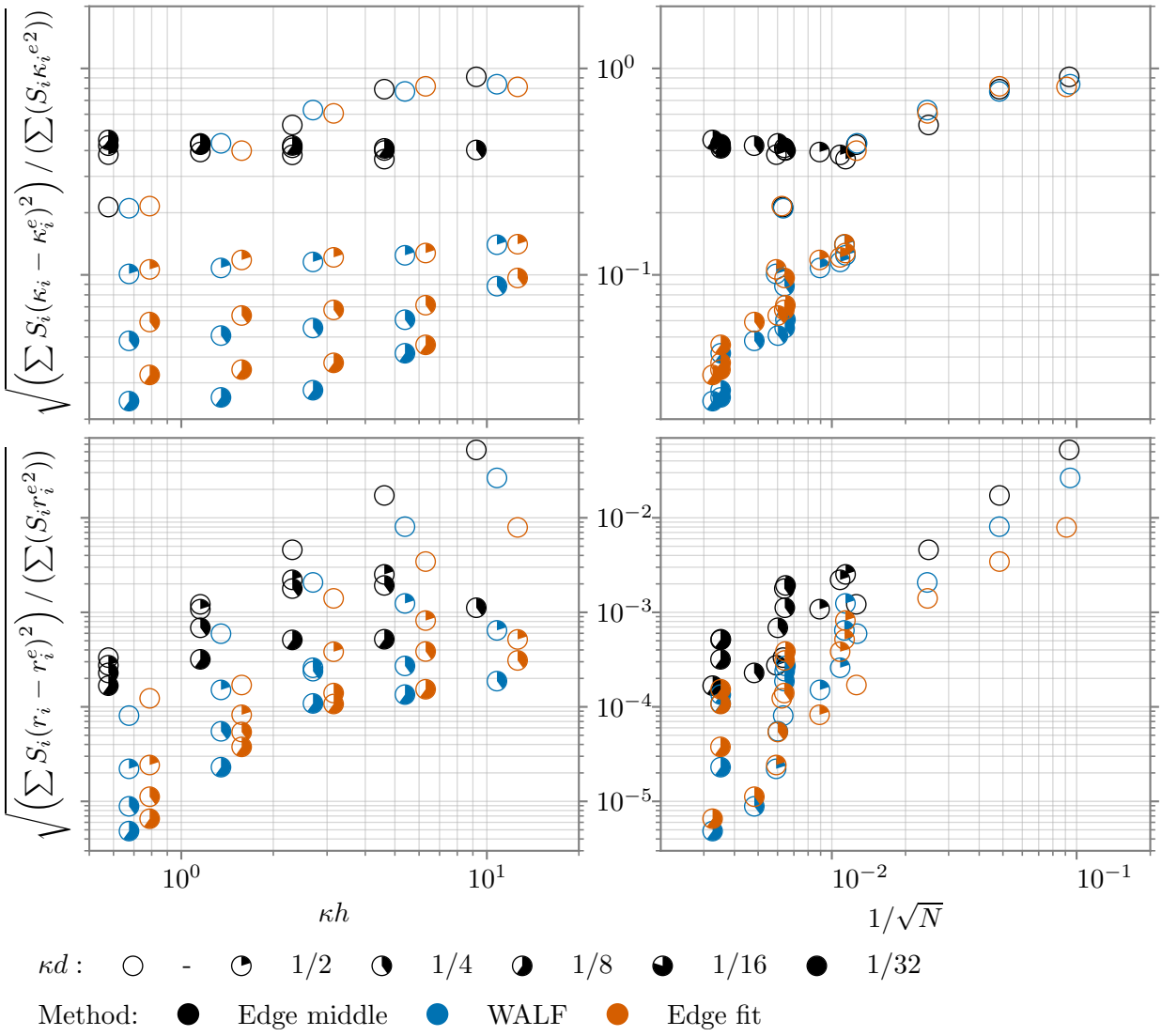


Figure D.17: Exact transport, whole domain, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), radial reference position, l_∞ norm in fig. D.18 (on page 136)

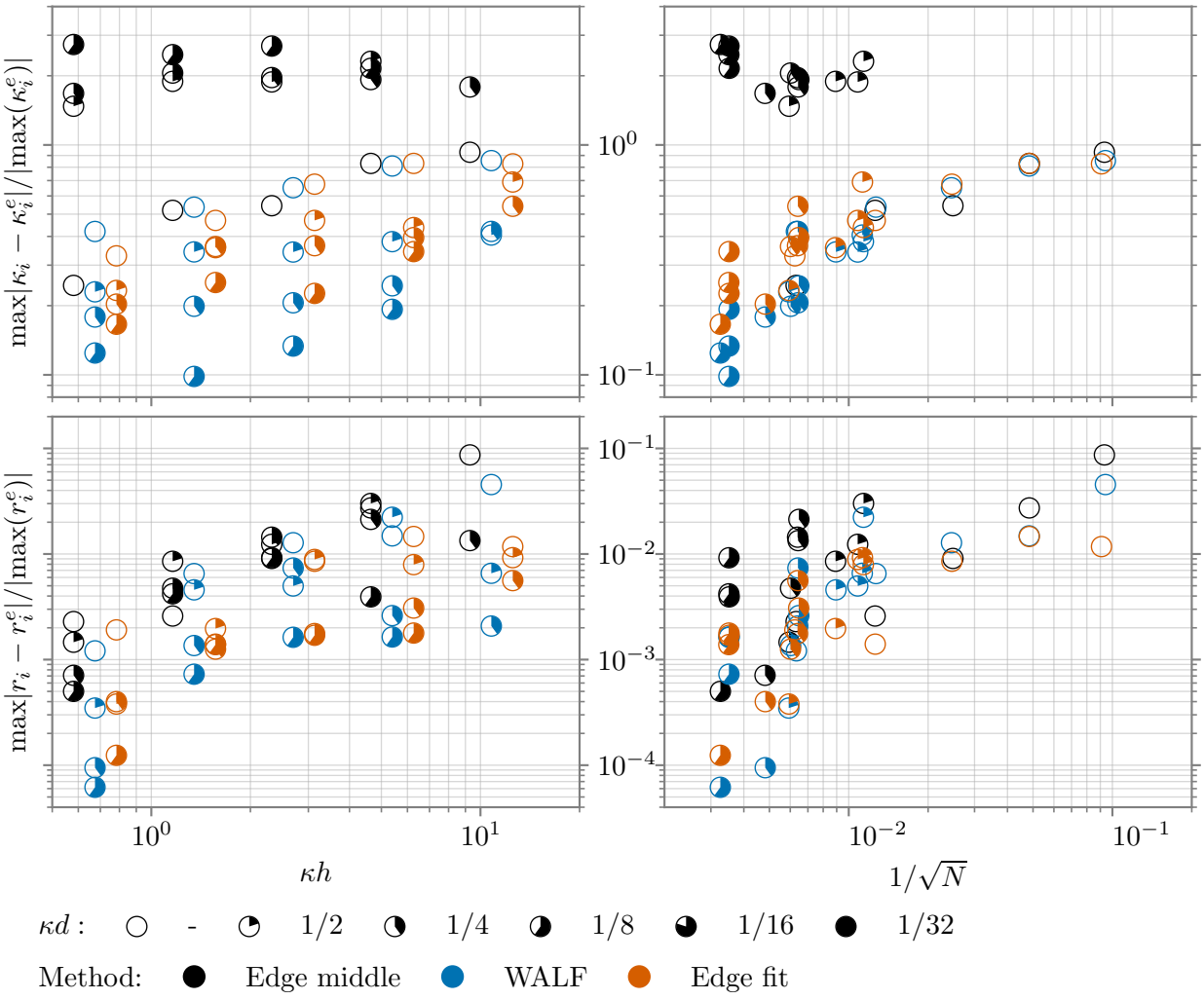


Figure D.18: Exact transport, whole domain, radial reference position, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), l_2 norm in fig. D.17 (on page 135)

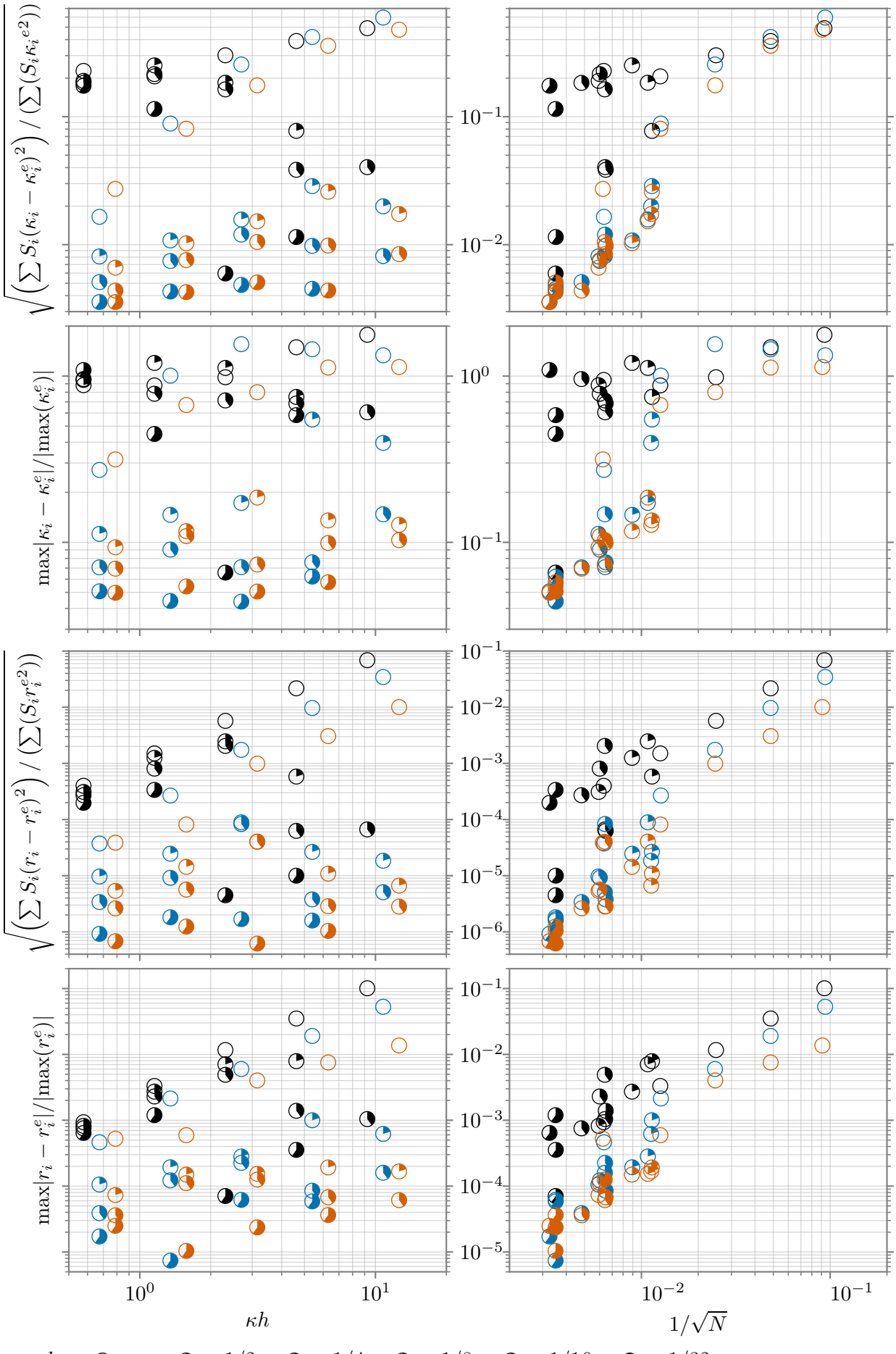


Figure D.19: Exact transport, zone 3, radial reference position, whole domain in fig. D.17 (on page 135)

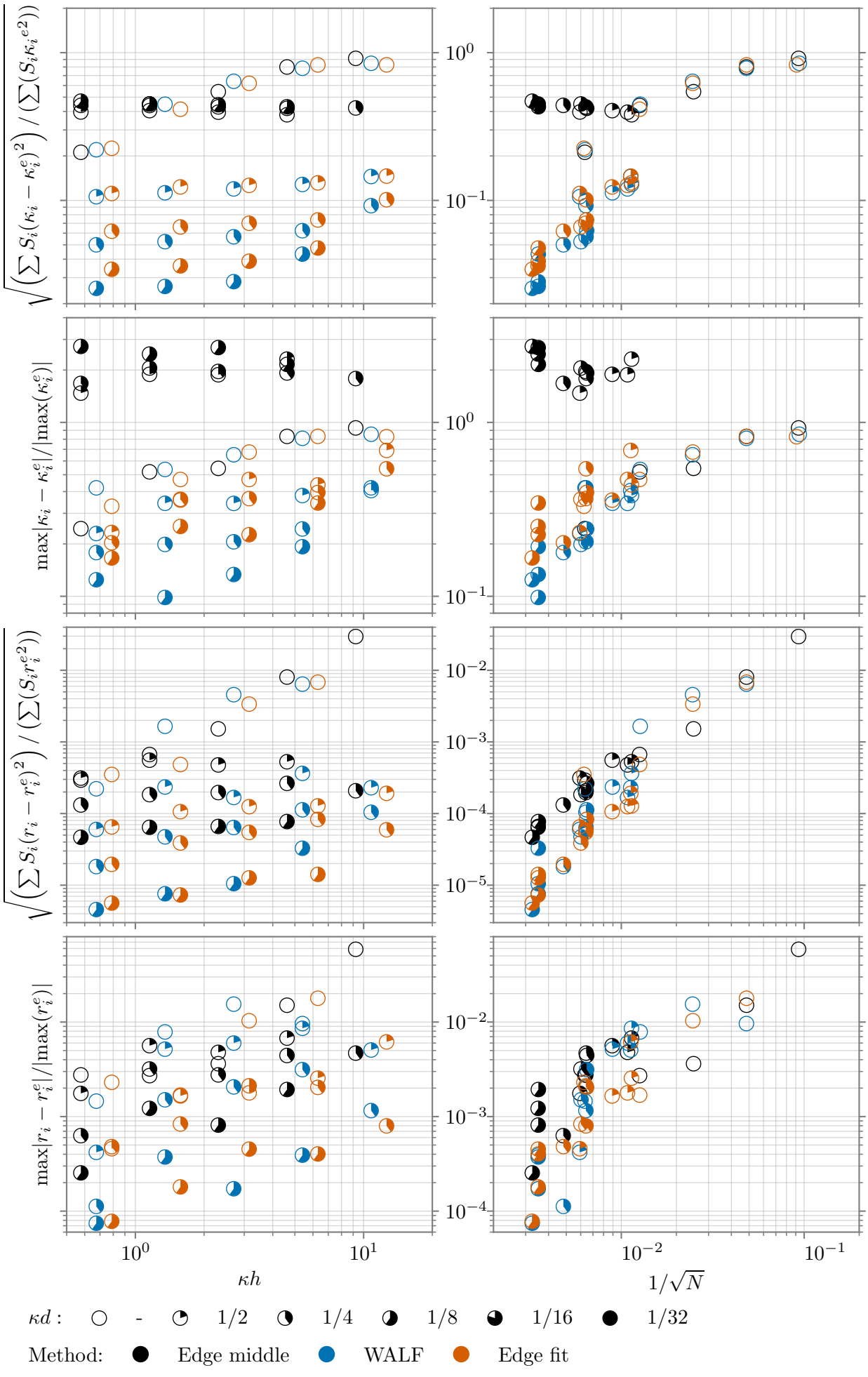


Figure D.20: Exact transport, zone 2, radial reference position, whole domain in fig. D.17 (on page 135)

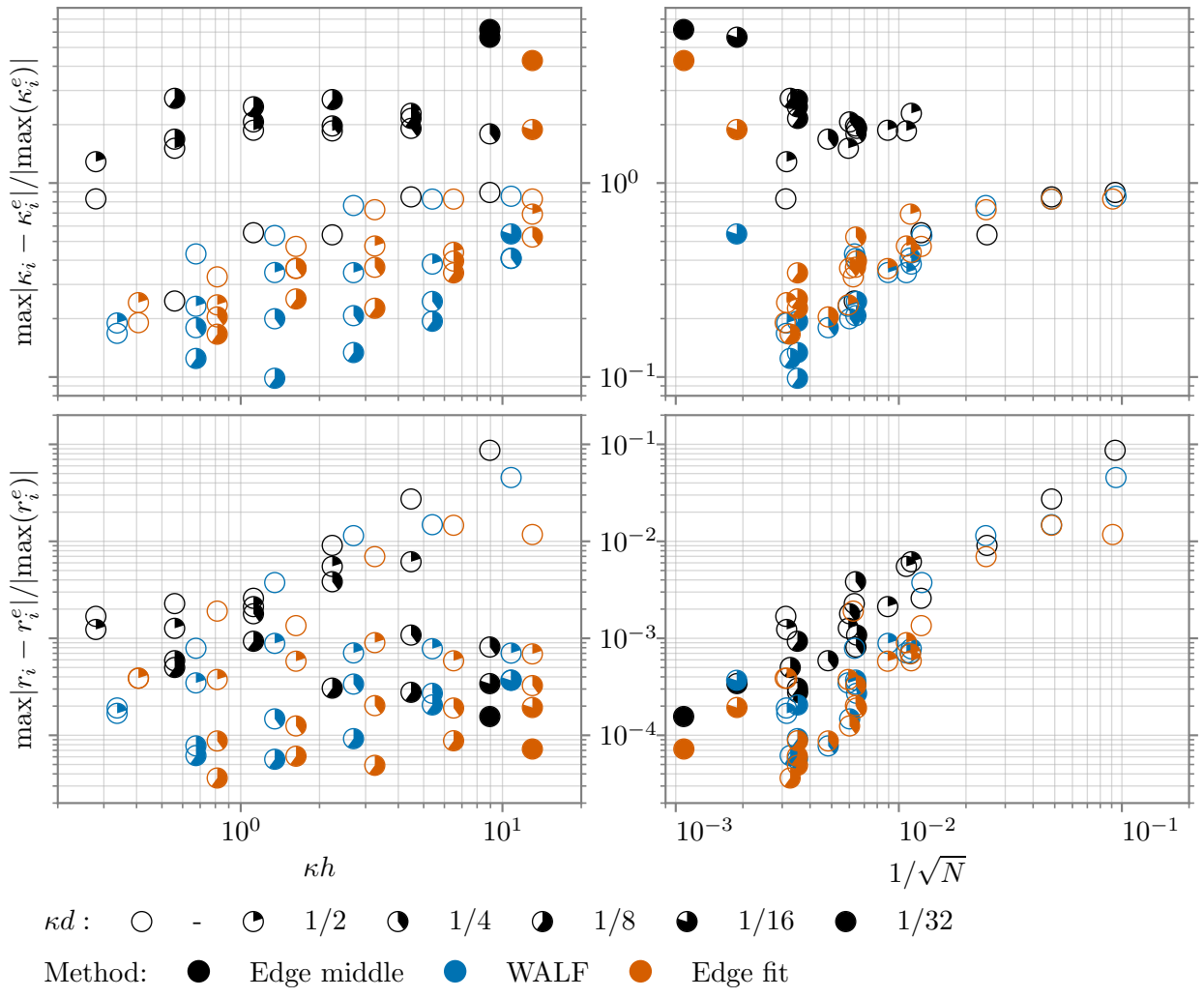


Figure D.21: Exact transport, whole domain, reference with minimal distance, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), l_2 norm in fig. 4.19 (on page 61)

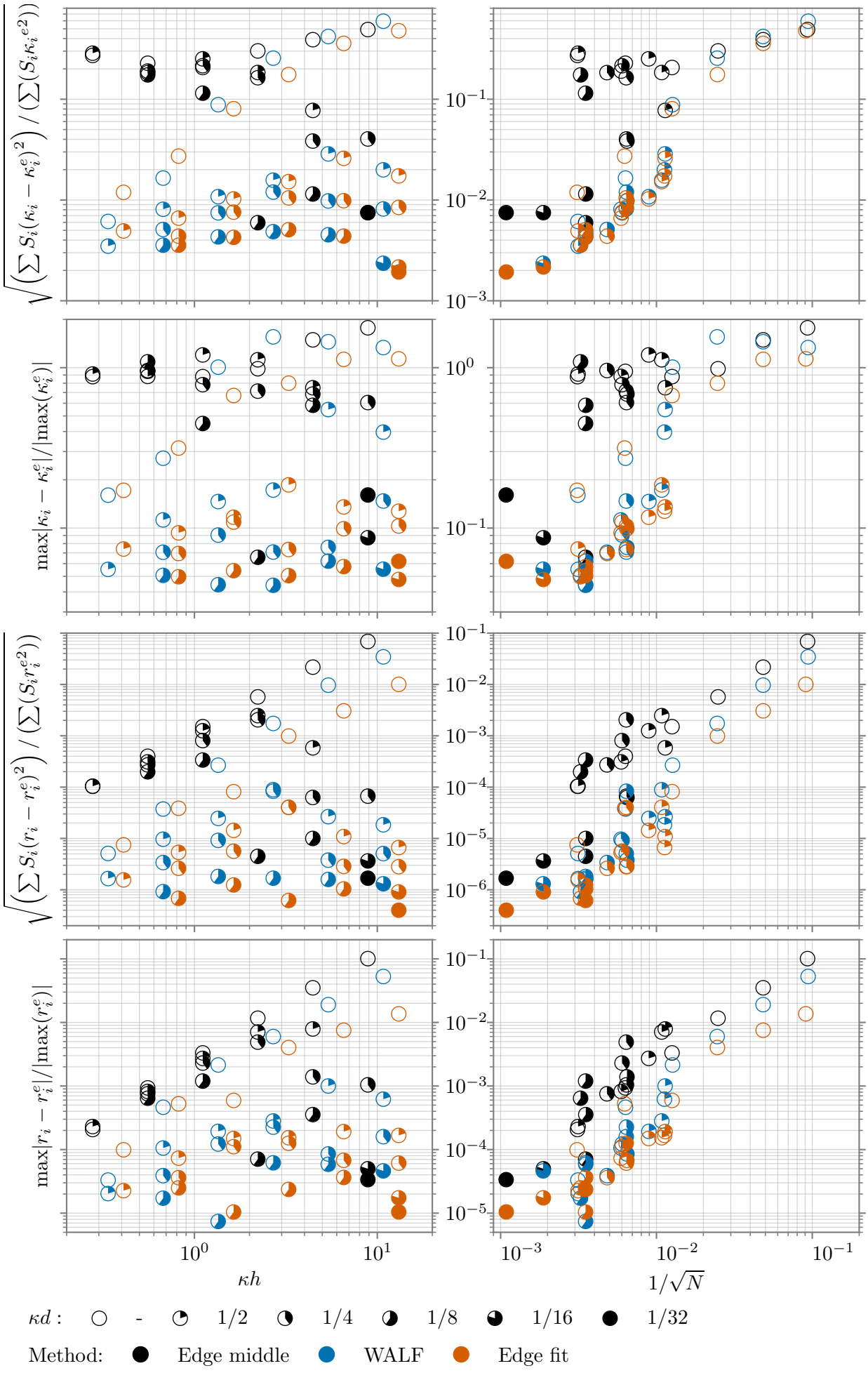


Figure D.22: Exact transport, zone 3, reference with minimal distance, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), whole domain in l_2 norm in fig. 4.19 (on page 61)

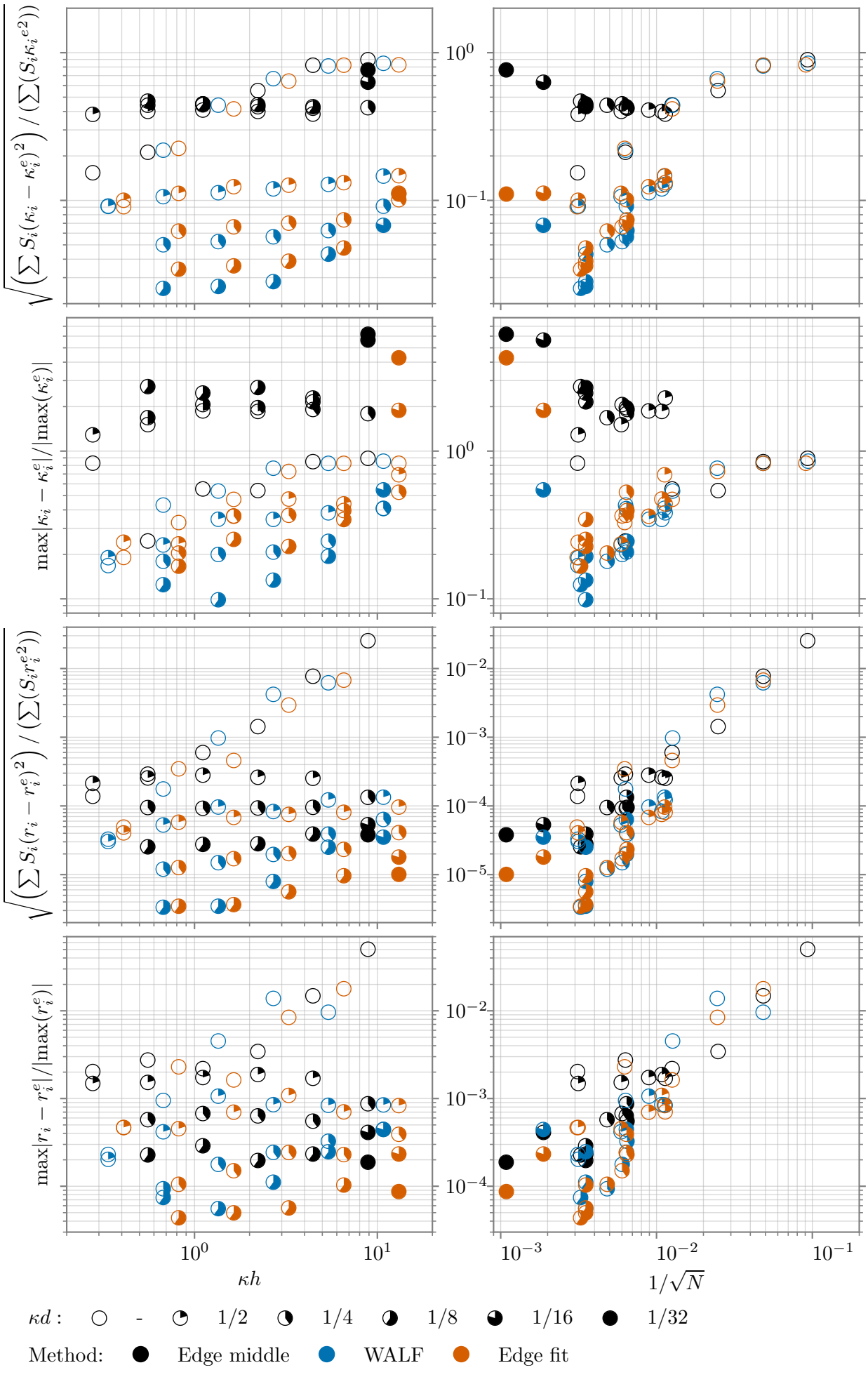


Figure D.23: Exact transport, zone 2, reference with minimal distance, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), whole domain in l_2 norm in fig. 4.19 (on page 61)

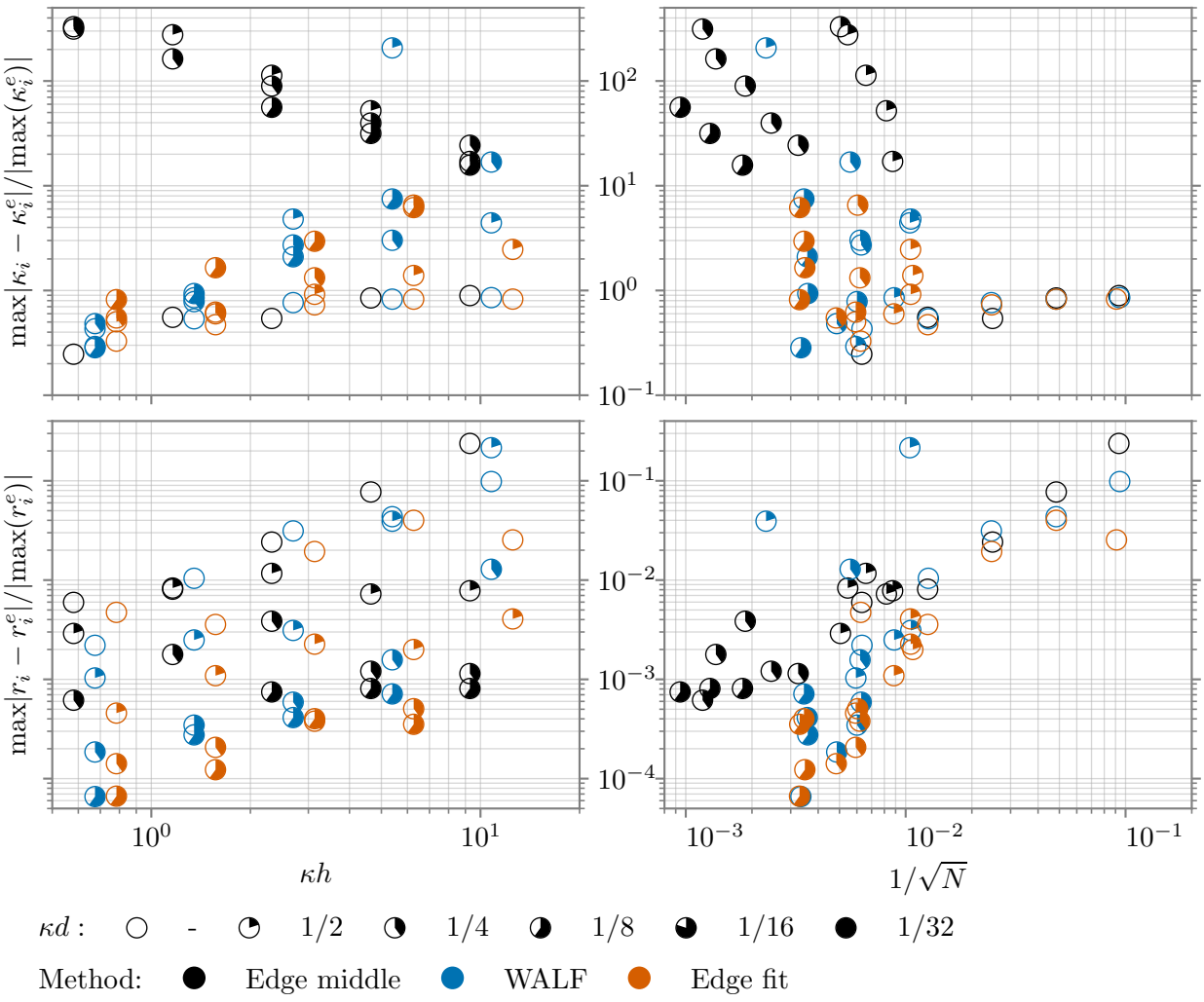


Figure D.24: Exact transport, whole domain, reference with minimal distance, adaptation to numerical curvature, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), l_2 norm in fig. 4.20 (on page 63)

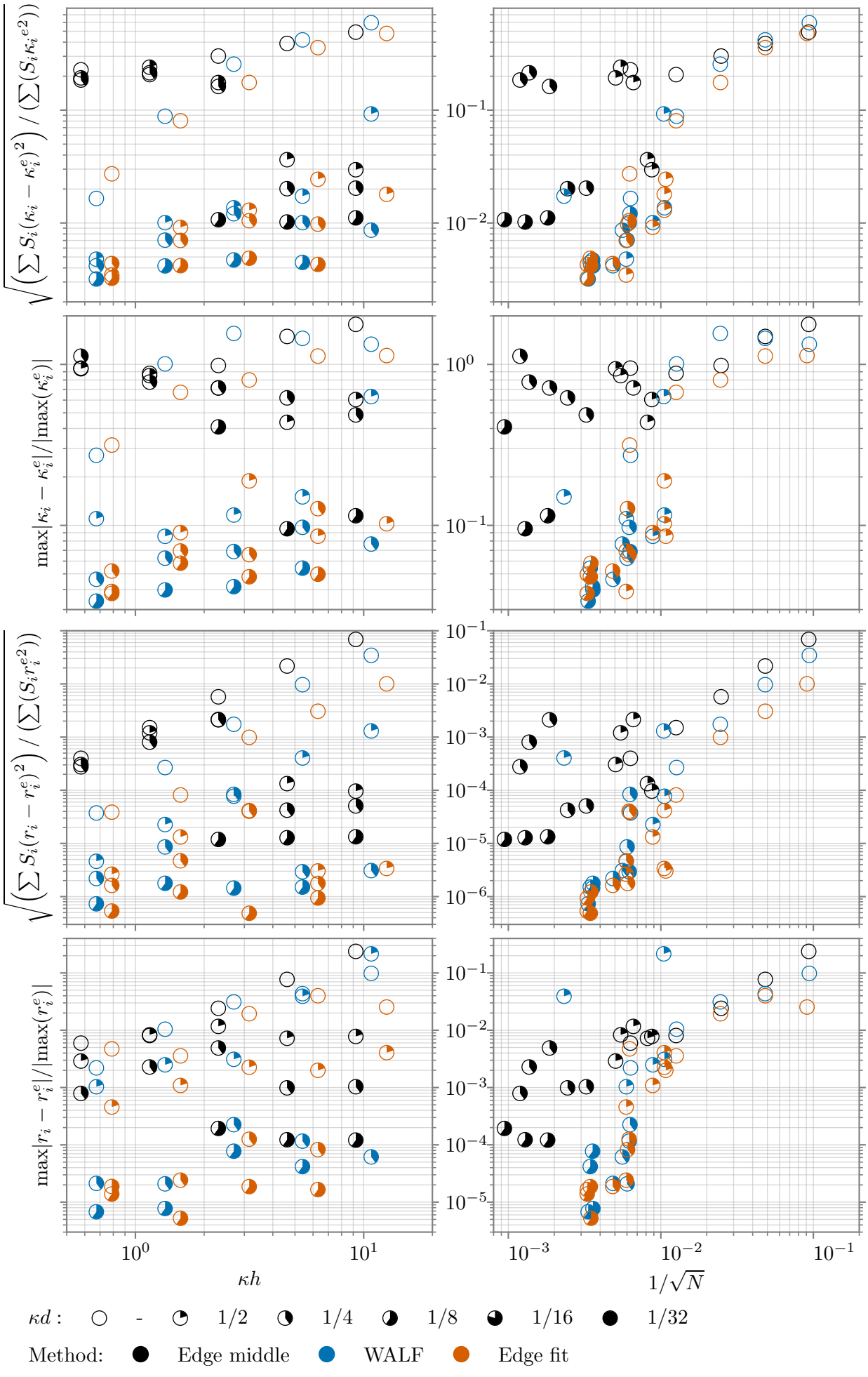


Figure D.25: Exact transport, zone 3, reference with minimal distance, adaptation to numerical curvature, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), whole domain in l_2 norm in fig. 4.20 (on page 63)

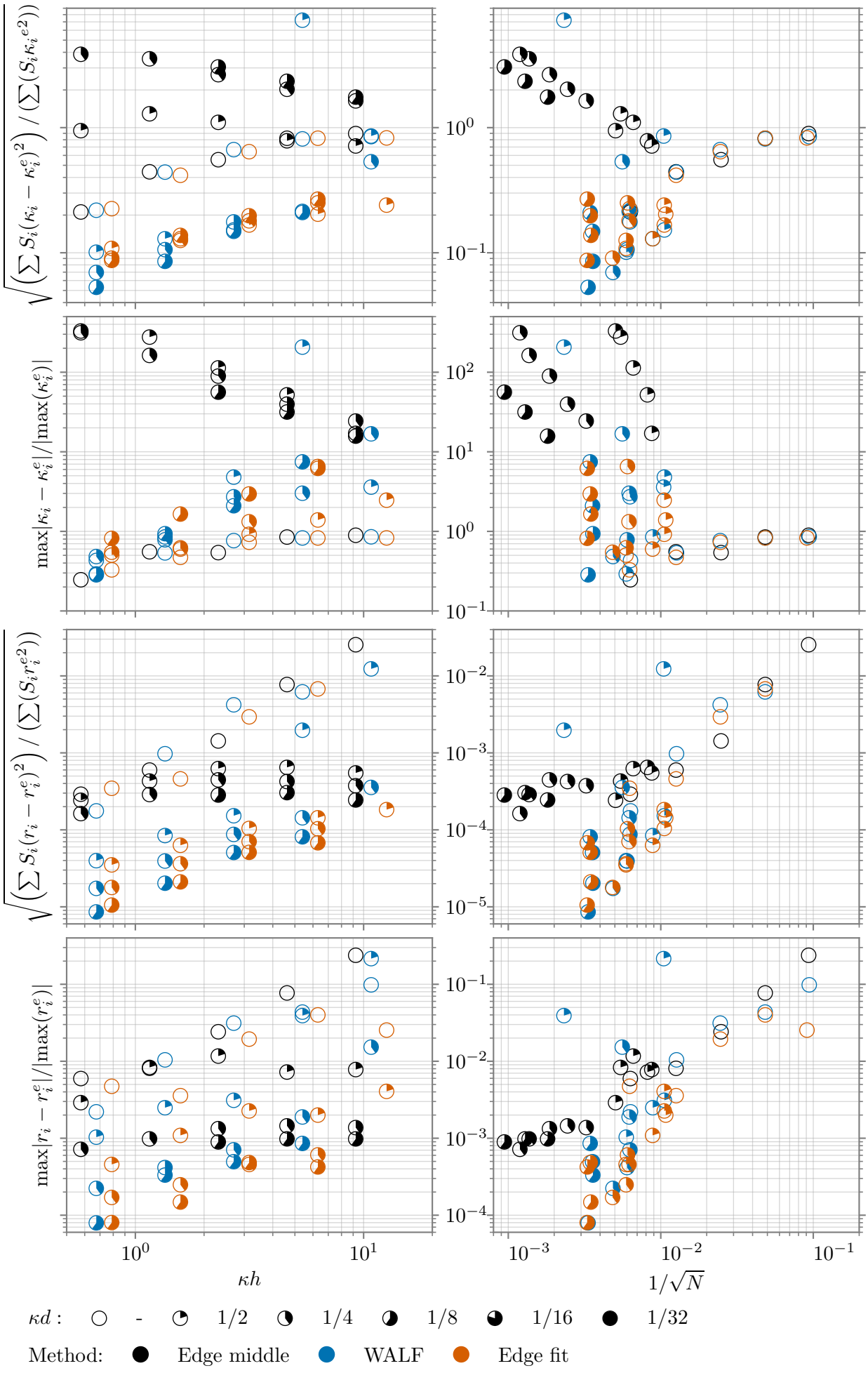


Figure D.26: Exact transport, zone 2, reference with minimal distance, adaptation to numerical curvature, edge middle, WALF (Jiao and Wang, 2012), edge fit (Gorges et al., 2022), table 3.1 (on page 30), whole domain in l_2 norm in fig. 4.20 (on page 63)

D.10 Influence of the time-step with numerical curvatures for the remeshing criteria

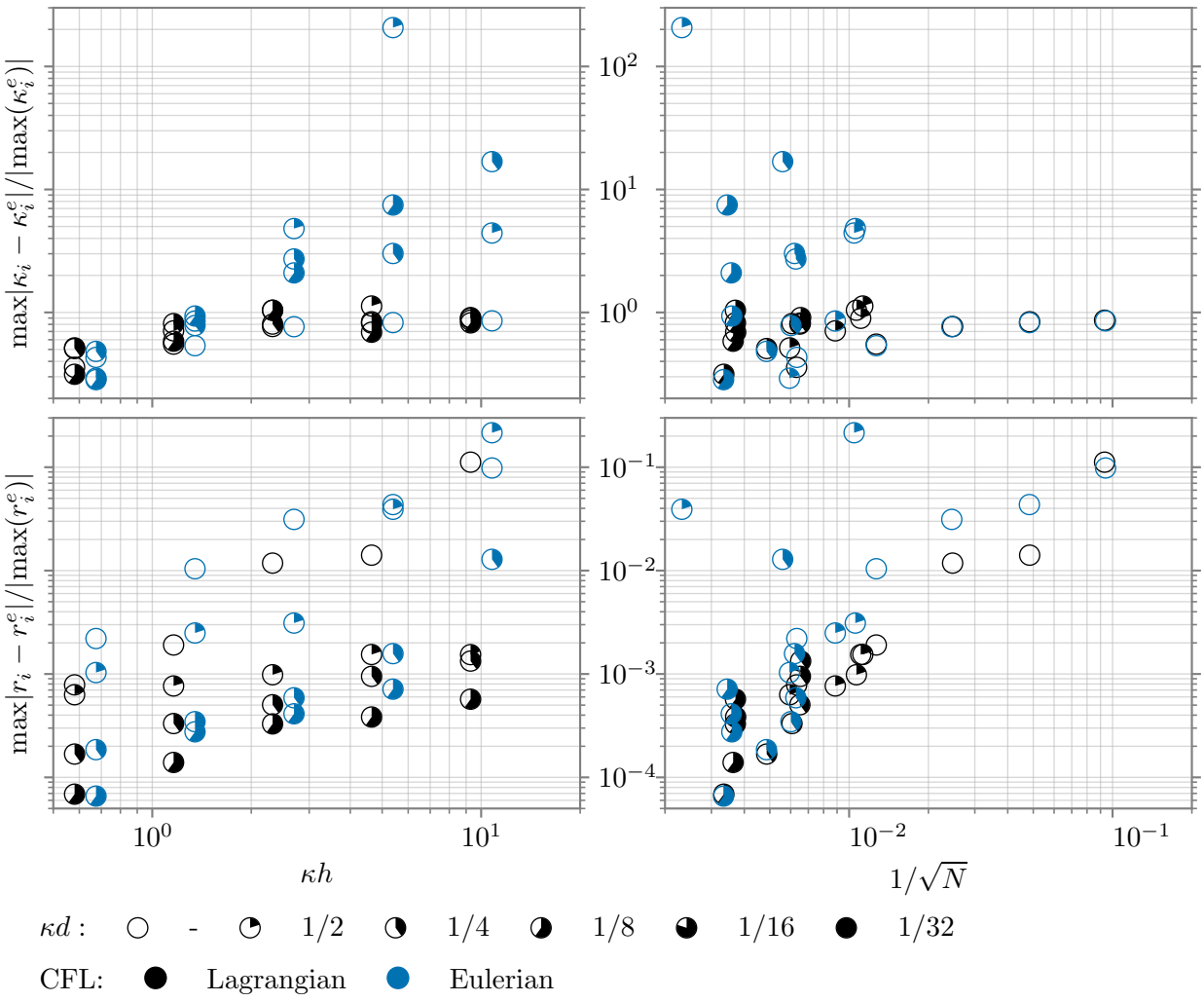


Figure D.27: Exact transport, whole domain, reference position with minimal distance, adaptation to numerical curvature edge middle, Lagrangian and Eulerian CFL, WALF (Jiao and Wang, 2012), l_2 norm in fig. 4.21 (on page 64)

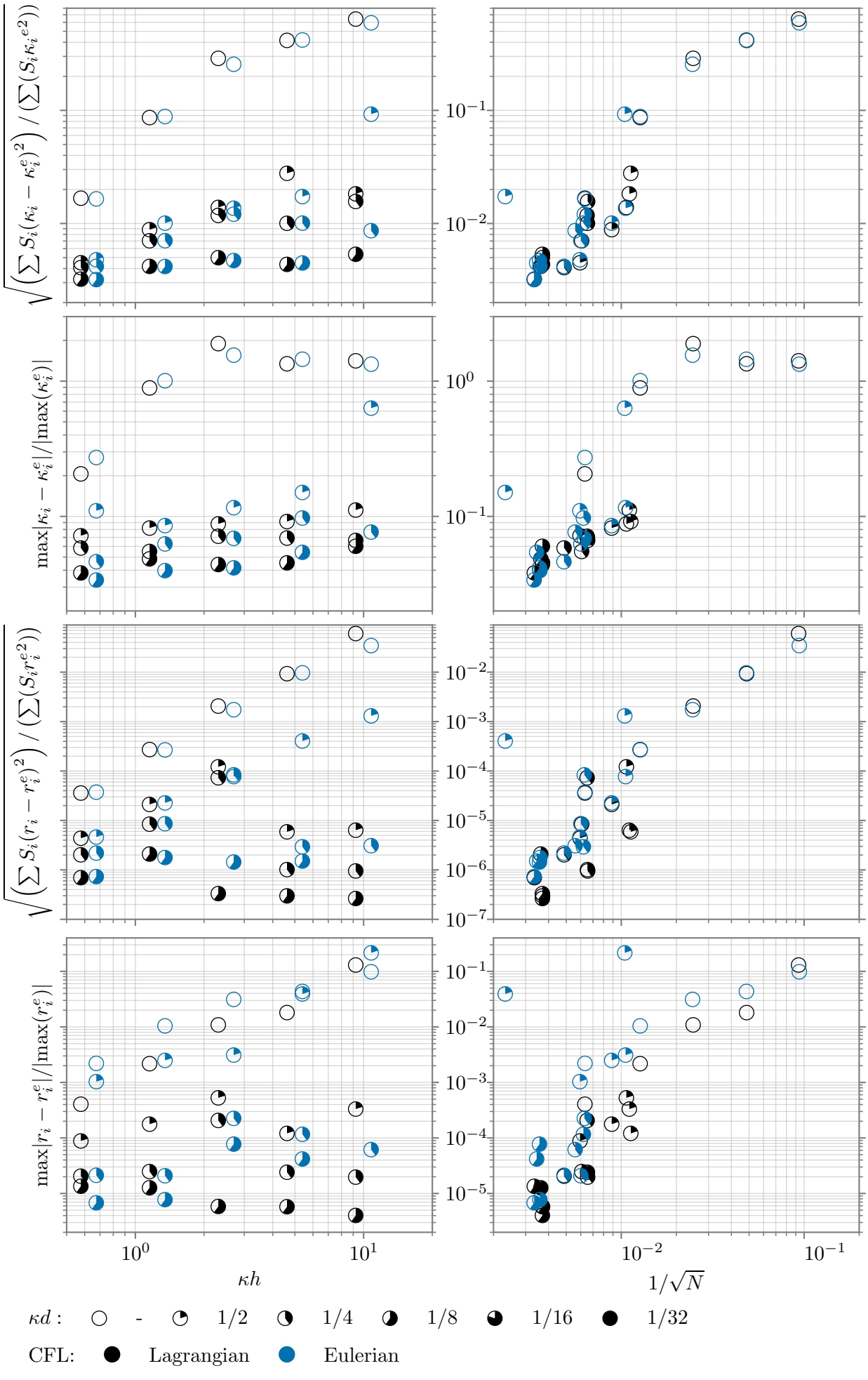


Figure D.28: Exact transport, whole domain, reference position with minimal distance, adaptation to numerical curvature edge middle, Lagrangian and Eulerian CFL, WALF (Jiao and Wang, 2012), whole domain in l_2 norm in fig. 4.21 (on page 64)

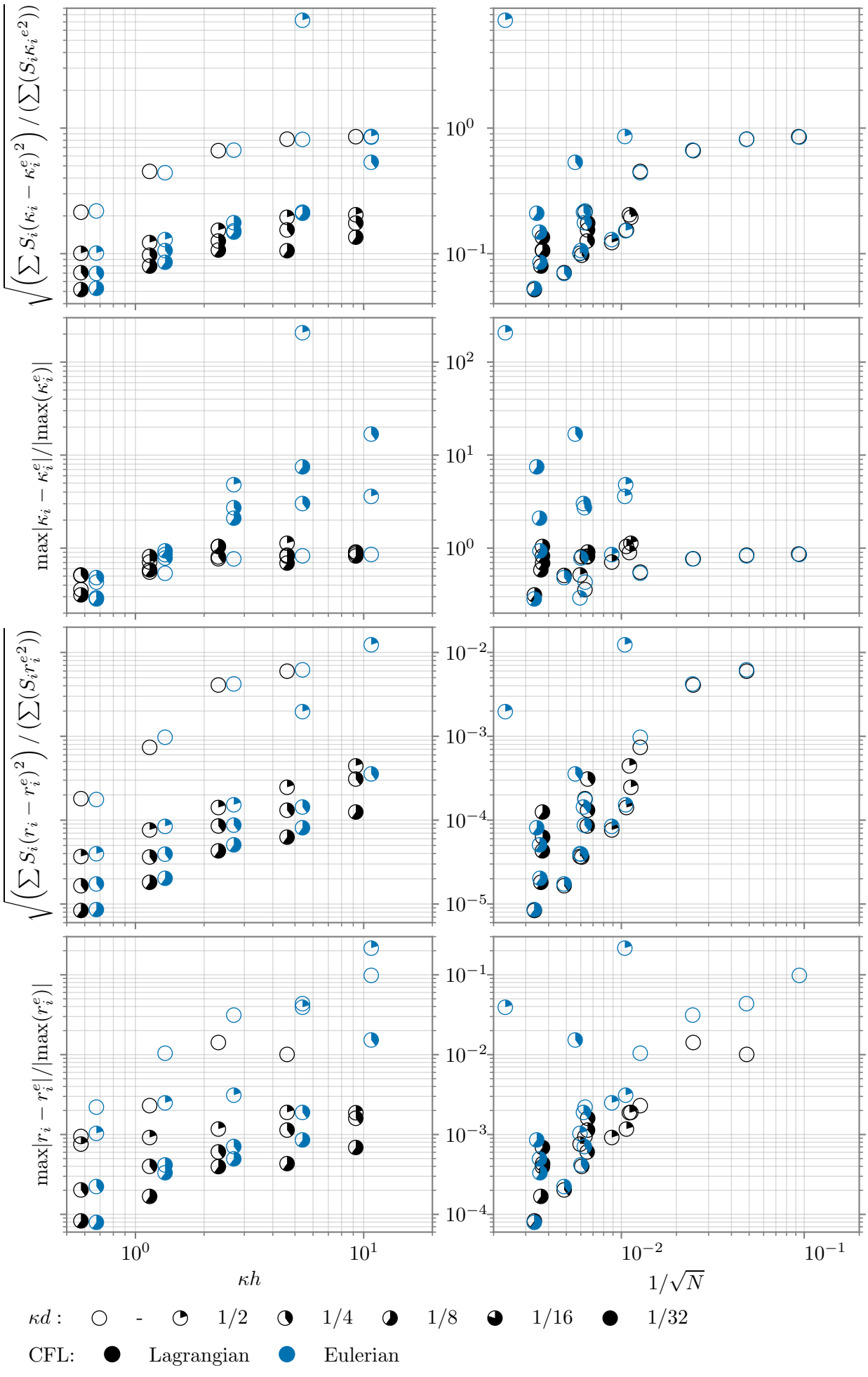


Figure D.29: Exact transport, whole domain, reference position with minimal distance, adaptation to numerical curvature edge middle, Lagrangian and Eulerian CFL, WALF (Jiao and Wang, 2012), whole domain in l_2 norm in fig. 4.21 (on page 64)

E Radial

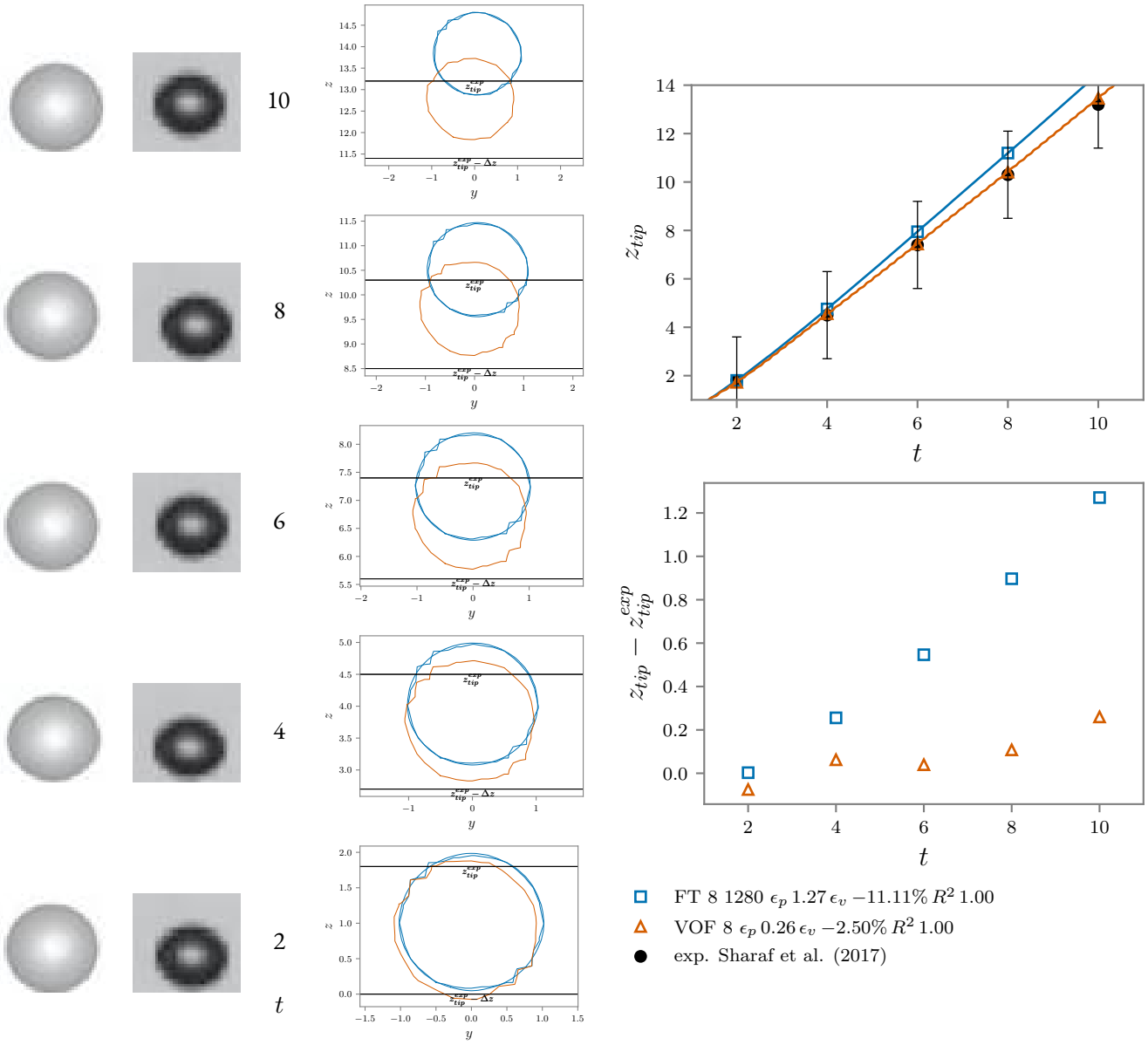


Figure E.1: Morton 2.4×10^{-7}, 8 cells per diameter, $\Delta t = 10^{-3}$. The results from Sharaf et al. (2017) are indicated in black with their experimental errors. The first two rows are figures from a simulation and experiment from Sharaf et al. (2017). Reproduced from Sharaf et al. (2017), with the permission of AIP Publishing.

Acronyms

ALE	Arbitrary Lagrangian-Eulerian 5, 71, 151
AMR	Adaptive Mesh Refinement 3, 4, 68, 71, 83, 86, 105, 151
CAD	Computer Aided Design 85, 151
CCL	Connected-Component Labeling 83, 151
CFL	Courant-Friedrichs-Lewy condition, Courant number 8, 73, 100, 151
CGAL	Computational Geometry Algorithms Library 85, 151
CLSVOF	Coupled Level-Set/Volume-of-Fluid 83, 151
CMF	Continuous Moving Frames 29, 151
CSF	Continuum Surface Force 18, 151
CUBISTA	Convergent and Universally Bounded Interpolation Scheme for the Treatment of Advection 7, 151
DIM	Diffuse Interface Model 6, 151
EBIT	Edge-Based Interface-Tracking 86, 151
Eo	Eotvos number 71, 151
FD	Fictitious Domain 4, 151
FEM	Finite Element Method 4, 71, 151
FT	Front-Tracking 23, 151
FV	Finite Volume 4, 151
FVM	Finite Volume Method 7, 8, 151
Ga	Galilei number 71, 151
GB	Grid-Based 85, 151
GF	Grid-Free 84, 85, 87, 151
GFM	Ghost-Fluid Method 7, 105, 151
HR	Homothetic Rescaling 20, 41, 45, 117, 151, 165, 168
IBM	Immersed Boundary Method 11, 151
IF	Integral Formulation 17, 18, 23, 71, 80, 97, 99, 105, 151
La	Laplace number 96, 151
LB	Lattice Boltzmann 4, 151
LBO	Laplace-Beltrami Operator 17, 23, 24, 151
LCRM	Level Contour Reconstruction Method 12, 85, 86, 151, 165
LFRM	Level Front Reconstruction Method 86, 151
LGB	Locally Grid-Based 85, 151
LS	Level-Set 5–7, 12, 83, 86, 151
MLS	Moving Least-Squares 151
Mo	Morton number 71, 151
MSA	Memoryless Simplification algorithm 20, 29, 151
OFM	One-Fluid Model 3–6, 151
PERM	Parabolic Edge Reconstruction Method 15, 49, 65, 66, 68, 73, 109, 110, 151, 166, 168
PLIC	Piecewise Linear Interface Calculation 151
RBF	Radial Basis Function 57, 151
Re	Reynolds number 151
SDF	Shape Diameter Function 85, 151

SGS	Sub-Grid Scale 3, 83, 84, 95, 100, 101, 105, 151, 167
SPH	Smoothed Particle Hydrodynamics 4, 151
SSP-RK3	Third-order Strong Stability-Preserving Runge-Kutta method 7, 16, 73, 109, 151
VBC	Velocity-Based Correction 19, 41, 45, 73, 117, 151, 165, 168
VOF	Volume Of Fluid 5–7, 9, 12, 71–78, 81, 83, 105, 151, 166, 167
WALF	Weighted Averaging of Local Fittings 29, 39, 54, 151
We	Weber number 3, 151

Nomenclature

$\operatorname{card} \mathcal{T}$	Cardinal of $\mathcal{T}$ 19, 151
χ	Phase indicator function 17, 111, 151
C	Volume fraction, color function 5, 18, 111, 151
d	Reference edge length 11, 151
δ	Dirac function 151
δ_h	Smoothed approximation to the Dirac delta function 11, 151
h	Eulerian mesh spacing 11, 23, 29, 37, 97, 100, 151, 171
κ	Mean curvature 4, 26, 46, 151
κ_1	Maximal principal curvature 26, 46, 151
κ_2	Minimal principal curvature 26, 46, 151
κ_G	Gaussian curvature 26, 46, 151
κ_{max}	The maximum of the absolute value of the principal curvatures 27, 151
$\mathcal{N}_1$	First neighborhood 13, 151, 165
$\mathcal{N}_2$	Second neighborhood 13, 37, 151, 165
Ω	Control volume 17, 18, 151
μ	Dynamic viscosity 151
ρ	Density 3, 151
σ	Surface tension 3, 4, 17, 151
θ	Polar angle 46, 151
$\mathcal{V}$	Numerical volume, volume delimited by the triangle mesh 39, 151, 165
$\mathcal{V}^e$	Exact volume 39, 151

Bibliography

Moataz O. Abu-Al-Saud, Stéphane Popinet, and Hamdi A. Tchelepi. 2018. A conservative and well-balanced surface tension model. *Journal of Computational Physics*, 371:896–913.

Alberto Roman Afanador, Stéphane Zaleski, Gretar Tryggvason, and Jiacai Lu. 2021. Effect of topology changes on the breakup of a periodic liquid jet. *Computers & Fluids*, page 105059.

Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. 2003. Anisotropic polygonal remeshing. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 485–493, New York, NY, USA. Association for Computing Machinery.

Pierre Alliez, Mark Meyer, and Mathieu Desbrun. 2002. Interactive geometry remeshing. *ACM Transactions on Graphics*, 21(3):347–354.

M. A. Alves, P. J. Oliveira, and F. T. Pinho. 2003. A convergent and universally bounded interpolation scheme for the treatment of advection. *International Journal for Numerical Methods in Fluids*, 41(1):47–75.

Philippe Angot, Charles-Henri Bruneau, and Pierre Fabrie. 1999. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, 81(4):497–520.

G. R. Anjos, N. Borhani, N. Mangiavacchi, and J. R. Thome. 2014. A 3D moving mesh Finite Element Method for two-phase flows. *Journal of Computational Physics*, 270:366–377.

N. Ashgriz and P. Givi. 1987. Binary collision dynamics of fuel droplets. *International Journal of Heat and Fluid Flow*, 8(3):205–210.

N. Ashgriz and J. Y. Poo. 1990. Coalescence and separation in binary collisions of liquid drops. *Journal of Fluid Mechanics*, 221:183–204.

G. Balarac, F. Basile, P. Bénard, F. Bordeu, J.-B. Chapelier, L. Cirrottola, G. Caumon, C. Dapogny, P. Frey, A. Froehly, G. Ghigliotti, R. Laraufie, G. Lartigue, C. Legentil, R. Mercier, V. Moureau, C. Nardoni, S. Pertant, and M. Zakari. 2022. Tetrahedral remeshing in the context of large-scale numerical simulation and high performance computing. *MathematicS In Action*, 11(1):129–164.

M. W. Baltussen, J. A. M. Kuipers, and N. G. Deen. 2014. A critical comparison of surface tension models for the volume of fluid method. *Chemical Engineering Science*, 109:65–74.

Adlène Benkenida and Jacques Magnaudet. 2000. Une méthode de simulation d'écoulements diphasiques sans reconstruction d'interfaces. *Comptes Rendus de l'Académie des Sciences - Series IIB - Mechanics-Physics-Astronomy*, 328(1):25–32.

D. Bhaga and M. E. Weber. 1981. Bubbles in viscous liquids: Shapes, wakes and velocities. *Journal of Fluid Mechanics*, 105:61–85.

S. Bnà, S. Manservisi, R. Scardovelli, P. Yecko, and S. Zaleski. 2016. Vofi — A library to initialize the volume fraction scalar field. *Computer Physics Communications*, 200:291–299.

Wurigen Bo, Xingtao Liu, James Glimm, and Xiaolin Li. 2011. A Robust Front Tracking Method: Verification and Application to Simulation of the Primary Breakup of a Liquid Jet. *SIAM Journal on Scientific Computing*, 33(4):1505–1524.

Guillaume Bois. 2017. Direct numerical simulation of a turbulent bubbly flow in a vertical channel: Towards an improved second-order reynolds stress model. *Nuclear Engineering and Design*, 321:92–103.

J. U Brackbill, D. B Kothe, and C Zemach. 1992. A continuum method for modeling surface tension. *Journal of Computational Physics*, 100(2):335–354.

G. Brenn and A. Frohn. 1989. Collision and merging of two equal droplets of propanol. *Experiments in Fluids*, 7(7):441–446.

Tyson Brochu. 2012. *Dynamic Explicit Surface Meshes and Applications*. Ph.D. thesis, University of British Columbia.

Tyson Brochu and Robert Bridson. 2009. Robust Topological Operations for Dynamic Explicit Surfaces. *SIAM Journal on Scientific Computing*, 31(4):2472–2493.

Bernard Bunner and Grétar Tryggvason. 2002. Dynamics of homogeneous bubbly flows Part 1. Rise velocity and microstructure of the bubbles. *Journal of Fluid Mechanics*, 466:17–52.

Jean-Paul Caltagirone and Stéphane Vincent. 2001. Sur une méthode de pénalisation tensorielle pour la résolution des équations de Navier–Stokes. *Comptes Rendus de l'Académie des Sciences - Series IIB - Mechanics*, 329(8):607–613.

F. Cazals and M. Pouget. 2005. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121–146.

Hector D. Ceniceros. 2010. A Robust, Fully Adaptive Hybrid Level-set/front-tracking Method for Two-phase Flows with an Accurate Surface Tension Computation. *Communications in Computational Physics*, 8(1):51–94.

Robert Chiodi and Olivier Desjardins. 2017. A reformulation of the conservative level set reinitialization equation for accurate and robust simulation of complex multiphase flows. *Journal of Computational Physics*, 343:186–200.

Robert Michael Chiodi. 2020. *Advancement of Numerical Methods for Simulating Primary Atomization*. Ph.D. thesis.

Leonardo Chirco, Jacob Maarek, Stéphane Popinet, and Stéphane Zaleski. 2022. Manifold death: A Volume of Fluid implementation of controlled topological changes in thin sheets by the signature method. *Journal of Computational Physics*, 467:111468.

Leonardo Chirco and Stéphane Zaleski. 2023. An edge-based interface-tracking method for multiphase flows. *International Journal for Numerical Methods in Fluids*, 95(3):491–497.

Alexandre Joel Chorin. 1968. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–762.

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational shape approximation. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 905–914, New York, NY, USA. Association for Computing Machinery.

Emil Coyajee and Bendiks Jan Boersma. 2009. Numerical simulation of drop impact on a liquid–liquid interface with a multiple marker front-capturing method. *Journal of Computational Physics*, 228(12):4444–4467.

Vittorio Cristini, Jerzy Bławzdziewicz, and Michael Loewenberg. 2001. An Adaptive Mesh Algorithm for Evolving Surfaces: Simulations of Drop Breakup and Coalescence. *Journal of Computational Physics*, 168(2):445–463.

Fang Da, Christopher Batty, and Eitan Grinspun. 2014. Multimaterial mesh-based surface tracking. *ACM Transactions on Graphics*, 33(4):112:1–112:11.

F. S. de Sousa, N. Mangiavacchi, L. G. Nonato, A. Castelo, M. F. Tomé, V. G. Ferreira, J. A. Cuminato, and S. McKee. 2004. A front-tracking/front-capturing method for the simulation of 3D multi-fluid flows with free surfaces. *Journal of Computational Physics*, 198(2):469–499.

N. G. Deen, M. van Sint Annaland, and J. A. M. Kuipers. 2004. Multi-scale modeling of dispersed gas–liquid two-phase flow. *Chemical Engineering Science*, 59(8):1853–1861.

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 317–324, USA. ACM Press/Addison-Wesley Publishing Co.

Olivier Desjardins, Vincent Moureau, and Heinz Pitsch. 2008. An accurate conservative level set/ghost fluid method for simulating turbulent atomization. *Journal of Computational Physics*, 227(18):8395–8416.

W. Dijkhuizen, I. Roghair, M. Van Sint Annaland, and J. A. M. Kuipers. 2010. DNS of gas bubbles behaviour using an improved 3D front tracking model—Model development. *Chemical Engineering Science*, 65(4):1427–1437.

Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. van der Vorst. 1998. *Numerical Linear Algebra for High-Performance Computers*. Software, Environments, and Tools. Society for Industrial and Applied Mathematics.

Jian Du, Brian Fix, James Glimm, Xicheng Jia, Xiaolin Li, Yuanhua Li, and Lingling Wu. 2006. A simple package for front tracking. *Journal of Computational Physics*, 213(2):613–628.

A. du Cluzeau, G. Bois, and A. Toutant. 2019. Analysis and modelling of Reynolds stresses in turbulent bubbly up-flows from direct numerical simulations. *Journal of Fluid Mechanics*, 866:132–168.

Antonio Eirís, Luis Ramírez, Iván Couceiro, Javier Fernández-Fidalgo, José París, and Xesús Nogueira. 2023. MLS-SPH-ALE: A Review of Meshless-FV Methods and a Unifying Formulation for Particle Discretizations. *Archives of Computational Methods in Engineering*.

Simon El Ouafa, Stéphane Vincent, Vincent Le Chenadec, and Benoît Trouette. 2023. Fully-coupled parallel solver for the simulation of two-phase incompressible flows. *Computers & Fluids*, 265:105995.

Simon Elouafa. 2022. *Développement d'un Solveur Tout-Couplé Parallèle 3D Pour La Simulation Des Écoulements Diphasiques Incompressibles à Forts Rapports de Viscosités et de Masses Volumiques*. Ph.D. thesis.

Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. 2002. A Hybrid Particle Level Set Method for Improved Interface Capturing. *Journal of Computational Physics*, 183(1):83–116.

Andreas Fabri, Geert-Jan Giezeman, Lutz Kettner, Stefan Schirra, and Sven Schönherr. 2000. On the design of CGAL a computational geometry algorithms library. *Software: Practice and Experience*, 30(11):1167–1202.

Ronald P Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. 1999. A Non-oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method). *Journal of Computational Physics*, 152(2):457–492.

Giulia Finotello, Roeland F. Kooiman, Johan T. Padding, Kay A. Buist, Alfred Jongsma, Fredrik Innings, and J. A. M. Kuipers. 2017. The dynamics of milk droplet–droplet collisions. *Experiments in Fluids*, 59(1):17.

Christian Focke and Dieter Bothe. 2011. Computational analysis of binary collisions of shear-thinning droplets. *Journal of Non-Newtonian Fluid Mechanics*, 166(14):799–810.

Pascal J Frey. 2000. About surface remeshing. *Proceedings of the 9th International Meshing Roundtable, Sandia National Laboratories, 2000*.

Koji Fujiwara. 1995. Eigenvalues of Laplacians on a closed Riemannian manifold and its nets. *Proceedings of the American Mathematical Society*, 123(8):2585–2594.

Daniel Fuster, Anne Bagué, Thomas Boeck, Luis Le Moyne, Anthony Leboissetier, Stéphane Popinet, Pascal Ray, Ruben Scardovelli, and Stéphane Zaleski. 2009. Simulation of primary atomization with an octree adaptive mesh refinement and VOF method. *International Journal of Multiphase Flow*, 35(6):550–565.

O. S. Galaktionov, P. D. Anderson, G. W. M. Peters, and F. N. Van de Vosse. 2000. An adaptive front tracking technique for three-dimensional transient flows. *International Journal for Numerical Methods in Fluids*, 32(2):201–217.

Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '97*, pages 209–216, Not Known. ACM Press.

Timothy D. Gatzke and Cindy M. Grimm. 2006. Estimating curvature on triangular meshes. *International Journal of Shape Modeling*, 12(01):1–28.

James Glimm, John W. Grove, X. L. Li, and D. C. Tan. 2000. Robust Computational Algorithms for Dynamic Interface Tracking in Three Dimensions. *SIAM Journal on Scientific Computing*, 21(6):2240–2256.

Ron Goldman. 2005. Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, 22(7):632–658.

Christian Gorges, Fabien Evrard, Berend van Wachem, and Fabian Denner. 2022. Reducing volume and shape errors in front tracking by divergence-preserving velocity interpolation and parabolic fit vertex positioning. *Journal of Computational Physics*, 457:111072.

Christian Gorges, Azur Hoduzić, Fabien Evrard, Berend van Wachem, Clara M. Velte, and Fabian Denner. 2023. Efficient reduction of vertex clustering using front tracking with surface normal propagation restriction. *Journal of Computational Physics*, 491:112406.

Sigal Gottlieb, Chi-Wang Shu, and Eitan Tadmor. 2001. Strong Stability-Preserving High-Order Time Discretization Methods. *SIAM Review*, 43(1):89–112.

Alfred Gray, Elsa Abbena, and Simon Salamon. 2017. *Modern Differential Geometry of Curves and Surfaces with Mathematica*, 3 edition. CRC Press, New York.

Andrew K. Gunstensen, Daniel H. Rothman, Stéphane Zaleski, and Gianluigi Zanetti. 1991. Lattice Boltzmann model of immiscible fluids. *Physical Review A*, 43(8):4320–4327.

Austin Han and Olivier Desjardins. 2021. Liquid Structure Classification Towards Topology Change Modeling. *International Conference on Liquid Atomization and Spray Systems (ICLASS)*, 1(1).

M Herrmann. 2005. Refined Level Set Grid method for tracking interfaces. In *Annu. Res. Briefs*, pages 3–18.

Marcus Herrmann. 2011. On simulating primary atomization using the refined level set grid method. *Atomization and Sprays*, 21(4).

C. W Hirt and B. D Nichols. 1981. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225.

Shunji Homma, Jiro Koga, Shiro Matsumoto, Museok Song, and Grétar Tryggvason. 2006. Breakup mode of an axisymmetric liquid jet injected into another immiscible liquid. *Chemical Engineering Science*, 61(12):3986–3996.

Kaimo Hu, Dong-Ming Yan, David Bommes, Pierre Alliez, and Bedrich Benes. 2017. Error-Bounded and Feature Preserving Surface Remeshing with Minimal Angle Improvement. *IEEE Transactions on Visualization and Computer Graphics*, 23(12):2560–2573.

Jinsong Hua and Jing Lou. 2007. Numerical simulation of bubble rising in viscous liquid. *Journal of Computational Physics*, 222(2):769–795.

Jinsong Hua, Jan F. Stene, and Ping Lin. 2008. Numerical simulation of 3D bubbles rising in viscous liquids using a front tracking method. *Journal of Computational Physics*, 227(6):3358–3382.

S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, and L. Tobiska. 2009. Quantitative benchmark computations of two-dimensional bubble dynamics. *International Journal for Numerical Methods in Fluids*, 60(11):1259–1288.

Romain Janodet, Carlos Guillamón, Vincent Moureau, Renaud Mercier, Ghislain Lartigue, Pierre Bénard, Thibaut Ménard, and Alain Berlemont. 2022. A massively parallel accurate conservative level set algorithm for simulating turbulent atomization on adaptive unstructured grids. *Journal of Computational Physics*, 458:111075.

O. W. Jayaratne, Basil John Mason, J. T. Bartlett, and Patrick Maynard Stuart Blackett. 1997. The coalescence and bouncing of water drops at an air/water interface. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 280(1383):545–565.

X. Jiang and A. J. James. 2007. Numerical simulation of the head-on collision of two equal-sized drops with van der Waals forces. *Journal of Engineering Mathematics*, 59(1):99–121.

Xiangmin Jiao. 2007. Face offsetting: A unified approach for explicit moving interfaces. *Journal of Computational Physics*, 220(2):612–625.

Xiangmin Jiao, Andrew Colombi, Xinlai Ni, and John Hart. 2010. Anisotropic mesh adaptation for evolving triangulated surfaces. *Engineering with Computers*, 26(4):363–376.

Xiangmin Jiao and Duo Wang. 2012. Reconstructing high-order surfaces for meshing. *Engineering with Computers*, 28(4):361–373.

Xiangmin Jiao, Duo Wang, and Hongyuan Zha. 2011. Simple and effective variational optimization of surface and volume triangulations. *Engineering with Computers*, 27(1):81–94.

Xiangmin Jiao and Hongyuan Zha. 2008. Consistent computation of first- and second-order differential quantities for surface meshes. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*, SPM '08, pages 159–170, New York, NY, USA. Association for Computing Machinery.

Myungjoo Kang, Ronald P. Fedkiw, and Xu-Dong Liu. 2000. A Boundary Condition Capturing Method for Multiphase Incompressible Flow. *Journal of Scientific Computing*, 15(3):323–360.

I. Kataoka. 1986. Local instant formulation of two-phase flow. *International Journal of Multiphase Flow*, 12(5):745–758.

Khodor Khadra, Philippe Angot, Sacha Parneix, and Jean-Paul Caltagirone. 2000. Fictitious domain approach for numerical modelling of Navier–Stokes equations. *International Journal for Numerical Methods in Fluids*, 34(8):651–684.

Désir-André Koffi Bi. 2021. *Caractérisation numérique des courbures et des normales aux interfaces d'écoulements diphasiques*. Ph.D. thesis, Université Gustave Eiffel.

Désir-André Koffi Bi, Mathilde Tavares, Eric Chénier, and Stéphane Vincent. 2022. Accuracy and convergence of the curvature and normal vector discretizations for 3D static and dynamic front-tracking interfaces. *Journal of Computational Physics*, page 111197.

R. Krishna, M. I. Urseanu, J. M. van Baten, and J. Ellenberger. 1999. Wall effects on the rise of single gas bubbles in liquids. *International Communications in Heat and Mass Transfer*, 26(6):781–790.

Andrew Kuprat, Ahmed Khamayseh, Denise George, and Levi Larkey. 2001. Volume Conserving Smoothing for Piecewise Linear Curves, Surfaces, and Triple Lines. *Journal of Computational Physics*, 172(1):99–118.

Marcel Kwakkel, Wim-Paul Breugem, and Bendiks Jan Boersma. 2013. Extension of a CLSVOF method for droplet-laden flows with a coalescence/breakup model. *Journal of Computational Physics*, 253:166–188.

Vincent Le Chenadec and Heinz Pitsch. 2013. A monotonicity preserving conservative sharp interface flow solver for high density ratio two-phase flows. *Journal of Computational Physics*, 249:185–203.

Randall J. LeVeque. 1996. High-Resolution Conservative Algorithms for Advection in Incompressible Flow. *SIAM Journal on Numerical Analysis*, 33(2):627–665.

Bruno Lévy and Hao (Richard) Zhang. 2010. Spectral mesh processing. In *ACM SIGGRAPH 2010 Courses*, SIGGRAPH '10, pages 1–312, New York, NY, USA. Association for Computing Machinery.

Jie Li, Yuriko Y. Renardy, and Michael Renardy. 2000. Numerical simulation of breakup of a viscous drop in simple shear flow through a volume-of-fluid method. *Physics of Fluids*, 12(2):269–282.

Yipeng Li, Xinglin Zhao, Navamita Ray, and Xiangmin Jiao. 2021. Compact feature-aware Hermite-style high-order surface reconstruction. *Engineering with Computers*, 37(1):187–210.

Yuanhua Li. 2007. *Enhanced 3D Front Tracking Method with Locally Grid Based Interface Tracking*. Thesis, The Graduate School, Stony Brook University: Stony Brook, NY.

Peter Liepa. 2003. *Filling Holes in Meshes*. The Eurographics Association.

P. Lindstrom and G. Turk. 1998. Fast and memory efficient polygonal simplification. In *Proceedings Visualization '98 (Cat. No.98CB36276)*, pages 279–286.

Y. Ling, S. Zaleski, and R. Scardovelli. 2015. Multiscale simulation of atomization with small droplets represented by a Lagrangian point-particle model. *International Journal of Multiphase Flow*, 76:122–143.

Jun Liu, Tobias Tolle, Dieter Bothe, and Tomislav Marić. 2023. An unstructured finite-volume level set / front tracking method for two-phase flows with large density-ratios. *Journal of Computational Physics*, 493:112426.

M. Liu and D. Bothe. 2019. Toward the predictive simulation of bouncing versus coalescence in binary droplet collisions. *Acta Mechanica*, 230(2):623–644.

Xinguo Liu, Hujun Bao, Heung-Yeung Shum, and Qunsheng Peng. 2002. A Novel Volume Constrained Smoothing Method for Meshes. *Graphical Models*, 64(3):169–182.

Sean Mauch. 2000. A fast algorithm for computing the closest point and distance transform. *Go online to http://www. acm. caltech. edu/seanm/software/cpt/cpt. pdf*.

Nelson Max. 1999. Weights for Computing Vertex Normals from Facet Normals. *Journal of Graphics Tools*, 4(2):1–6.

R. McDermott and S. B. Pope. 2008. The parabolic edge reconstruction method (PERM) for Lagrangian particle advection. *Journal of Computational Physics*, 227(11):5447–5491.

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. 2003. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In *Visualization and Mathematics III*, Mathematics and Visualization, pages 35–57, Berlin, Heidelberg. Springer.

Nicolas Moës, Jean-François Remacle, Jonathan Lambrechts, Benoît Lé, and Nicolas Chevaugeon. 2023. The eXtreme Mesh deformation approach (X-MESH) for the Stefan phase change model. *Journal of Computational Physics*, 477:111878.

Tomas Möller and Ben Trumbore. 1997. Fast, Minimum Storage Ray-Triangle Intersection. *Journal of Graphics Tools*, 2(1):21–28.

Nishant Nangia, Boyce E. Griffith, Neelesh A. Patankar, and Amneet Pal Singh Bhalla. 2019. A robust incompressible Navier-Stokes solver for high density ratio multiphase flows. *Journal of Computational Physics*, 390:548–594.

Mani Chandan Naru. 2021. *Numerical Simulation of a Water/Oil Emulsion in a Multiscale/Multiphysics Context*. Ph.D. thesis, Sorbonne Université.

M. R. H. Nobari and Gretar Tryggvason. 1996. Numerical simulations of three-dimensional drop collisions. *AIAA Journal*, 34(4):750–755.

Carlos J. Ogayar, Rafael J. Segura, and Francisco R. Feito. 2005. Point in solid strategies. *Computers & Graphics*, 29(4):616–624.

Elin Olsson and Gunilla Kreiss. 2005. A conservative level set method for two phase flow. *Journal of Computational Physics*, 210(1):225–246.

Elin Olsson, Gunilla Kreiss, and Sara Zahedi. 2007. A conservative level set method for two phase flow II. *Journal of Computational Physics*, 225(1):785–807.

Mohamed El Ouafa. 2021. Navier-Stokes Solvers for Incompressible Single- and Two-Phase Flows. *Communications in Computational Physics*, 29(4):1213–1245.

Jieyun Pan, Tian Long, Leonardo Chirco, Ruben Scardovelli, Stéphane Popinet, and Stéphane Zaleski. 2023. An Edge-based Interface Tracking (EBIT) Method for Multiphase-flows Simulation with Surface Tension.

Kuo-Long Pan, Ping-Chung Chou, and Yu-Jen Tseng. 2009. Binary droplet collision at high Weber number. *Physical Review E*, 80(3):036301.

Charles S Peskin. 1977. Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25(3):220–252.

Charles S. Peskin. 2002. The immersed boundary method. *Acta Numerica*, 11:479–517.

Charles S. Peskin and Beth Feller Printz. 1993. Improved Volume Conservation in the Computation of Flows with Immersed Elastic Boundaries. *Journal of Computational Physics*, 105(1):33–46.

Sylvain Petitjean. 2002. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys*, 34(2):211–262.

G. Pianet, S. Vincent, J. Leboi, J. P. Caltagirone, and M. Anderhuber. 2010. Simulating compressible gas bubbles with a smooth volume tracking 1-Fluid method. *International Journal of Multiphase Flow*, 36(4):273–283.

M. R. Pivello, M. M. Villar, R. Serfaty, A. M. Roma, and A. Silveira-Neto. 2014. A fully adaptive front tracking method for the simulation of two phase flows. *International Journal of Multiphase Flow*, 58:72–82.

Márcio Ricardo Pivello. 2012. Um método front-tracking completamente adaptativo para a simulação de escoamentos tridimensionais bifásicos.

Carole Planchette, Francesco Marangon, Wen-Kai Hsiao, and Günter Brenn. 2019. Breakup of asymmetric liquid ligaments. *Physical Review Fluids*, 4(12):124004.

Stéphane Popinet. 2003. Gerris: A tree-based adaptive solver for the incompressible Euler equations in complex geometries. *Journal of Computational Physics*, 190(2):572–600.

Stéphane Popinet. 2009. An accurate adaptive solver for surface-tension-driven interfacial flows. *Journal of Computational Physics*, 228(16):5838–5866.

Stéphane Popinet. 2018. Numerical Models of Surface Tension. *Annual Review of Fluid Mechanics*, 50(1):49–75.

Stéphane Popinet and Stéphane Zaleski. 1999. A front-tracking algorithm for accurate representation of surface tension. *International Journal for Numerical Methods in Fluids*, 30(6):775–793.

J. Qian and C. K. Law. 1997. Regimes of coalescence and separation in droplet collision. *Journal of Fluid Mechanics*, 331:59–80.

Shaoping Quan, Jing Lou, and David P. Schmidt. 2009. Modeling merging and breakup in the moving mesh interface tracking method for multiphase flow simulations. *Journal of Computational Physics*, 228(7):2660–2675.

Shaoping Quan and David P. Schmidt. 2007. A moving mesh interface tracking method for 3D incompressible two-phase flows. *Journal of Computational Physics*, 221(2):761–780.

A. H. Rajkotwala, E. J. Gelissen, E. A. J. F. Peters, M. W. Baltussen, C. W. M. van der Geld, J. G. M. Kuerten, and J. A. M. Kuipers. 2020. Comparison of the local front reconstruction method with a diffuse interface model for the modeling of droplet collisions. *Chemical Engineering Science: X*, 7:100066.

Adnan Hatim Rajkotwala. 2020. Numerical simulations of boiling flows using the Local Front Reconstruction Method.

Navamita Ray. 2013. *High-Order Surface Reconstruction and Its Applications to Surface Integrals and Surface Remeshing*. Ph.D. thesis, The Graduate School, Stony Brook University: Stony Brook, NY.

Navamita Ray, Tristan Delaney, Daniel R. Einstein, and Xiangmin Jiao. 2014. Surface remeshing with robust high-order reconstruction. *Engineering with Computers*, 30(4):487–502.

M. Razizadeh, S. Mortazavi, and H. Shahin. 2018. Drop breakup and drop pair coalescence using front-tracking method in three dimensions. *Acta Mechanica*, 229(3):1021–1043.

R. D. Reitz and F. V. Bracco. 1982. Mechanism of atomization of a liquid jet. *The Physics of Fluids*, 25(10):1730–1742.

I. Roghair, M. Van Sint Annaland, and J. A. M. Kuipers. 2016. An improved Front-Tracking technique for the simulation of mass transfer in dense bubbly flows. *Chemical Engineering Science*, 152:351–369.

Ilia V. Roisman, Edin Berberović, and Cam Tropea. 2009. Inertia dominated drop collisions. I. On the universal flow in the lamella. *Physics of Fluids*, 21(5):052103.

A. Sarthou, S. Vincent, and J. P. Caltagirone. 2011. A second-order curvilinear to Cartesian transformation of immersed interfaces and boundaries. Application to fictitious domains and multiphase flows. *Computers & Fluids*, 46(1):422–428.

Ryan Schmidt and Tyson Brochu. 2016. Adaptive Mesh Booleans.

Xinglong Shang, Zhengyuan Luo, Elizaveta Ya. Gatapova, Oleg A. Kabov, and Bofeng Bai. 2018. GNBC-based front-tracking method for the three-dimensional simulation of droplet motion on a solid surface. *Computers & Fluids*, 172:181–195.

Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259.

D. M. Sharaf, A. R. Premlata, Manoj Kumar Tripathi, Badarinath Karri, and Kirti Chandra Sahu. 2017. Shapes and paths of an air bubble rising in quiescent liquids. *Physics of Fluids*, 29(12):122104.

Seungwon Shin, S. I. Abdel-Khalik, Virginie Daru, and Damir Juric. 2005. Accurate representation of surface tension using the level contour reconstruction method. *Journal of Computational Physics*, 203(2):493–516.

Seungwon Shin, Jalel Chergui, Damir Juric, Asma Farhaoui, Lyes Kahouadji, Laurette S Tuckerman, and Nicolas Périnet. 2013. Parallel Direct Numerical Simulation of Three-Dimensional Two-Phase Flows.

Seungwon Shin, Jalel Chergui, Damir Juric, Lyes Kahouadji, Omar K. Matar, and Richard V. Craster. 2018. A hybrid interface tracking – level set technique for multiphase flow with soluble surfactant. *Journal of Computational Physics*, 359:409–435.

Seungwon Shin and Damir Juric. 2002. Modeling Three-Dimensional Multiphase Flow Using a Level Contour Reconstruction Method for Front Tracking without Connectivity. *Journal of Computational Physics*, 180(2):427–470.

Seungwon Shin and Damir Juric. 2007. High order level contour reconstruction method. *Journal of Mechanical Science and Technology*, 21(2):311–326.

Seungwon Shin and Damir Juric. 2009. A hybrid interface method for three-dimensional multiphase flows based on front tracking and level set techniques. *International Journal for Numerical Methods in Fluids*, 60(7):753–778.

Seungwon Shin, Ikroh Yoon, and Damir Juric. 2011. The Local Front Reconstruction Method for direct simulation of two- and three-dimensional multiphase flows. *Journal of Computational Physics*, 230(17):6605–6646.

J. Shinjo and A. Umemura. 2010. Simulation of liquid jet primary breakup: Dynamics of ligament and droplet formation. *International Journal of Multiphase Flow*, 36(7):513–532.

Rajkeshar Singh and Wei Shyy. 2007. Three-dimensional adaptive Cartesian grid method with conservative interface restructuring and reconstruction. *Journal of Computational Physics*, 224(1):150–167.

M. Sussman, K. M. Smith, M. Y. Hussaini, M. Ohta, and R. Zhi-Wei. 2007. A sharp interface method for incompressible two-phase flows. *Journal of Computational Physics*, 221(2):469–505.

Mark Sussman and Elbridge Gerry Puckett. 2000. A Coupled Level Set and Volume-of-Fluid Method for Computing 3D and Axisymmetric Incompressible Two-Phase Flows. *Journal of Computational Physics*, 162(2):301–337.

Shintaro Takeuchi and Grétar Tryggvason. 2020. Volume conservation method for the three-dimensional front-tracking method. *Mechanical Engineering Letters*, 6:20–00216–20–00216.

G. Taubin. 1995. Curve and surface smoothing without shrinkage. In *Proceedings of IEEE International Conference on Computer Vision*, pages 852–857.

Mathilde Tavares. 2019. *Simulation et Modélisation Multi-Échelle d'écoulements Diphasiques*. Ph.D. thesis.

Mathilde Tavares, Désir-André Koffi Bi, Eric Chénier, and Stéphane Vincent. 2021. A Front-Tracking Method for Multiphase Flows with a Sharp Interface Representation. In *Turbulence and Interactions*, Notes on Numerical Fluid Mechanics and Multidisciplinary Design, pages 189–195, Cham. Springer International Publishing.

Hiroshi Terashima and Grétar Tryggvason. 2009. A front-tracking/ghost-fluid method for fluid interfaces in compressible flows. *Journal of Computational Physics*, 228(11):4012–4037.

Tobias Tolle, Dieter Bothe, and Tomislav Marić. 2020. SAAMPLE: A Segregated Accuracy-driven Algorithm for Multiphase Pressure-Linked Equations. *Computers & Fluids*, 200:104450.

D. J. Torres and J. U. Brackbill. 2000. The Point-Set Method: Front-Tracking without Connectivity. *Journal of Computational Physics*, 165(2):620–644.

A. Toutant, B. Mathieu, and O. Lebaigue. 2012. Volume-conserving mesh smoothing for front-tracking methods. *Computers & Fluids*, 67:16–25.

G. M. Treece, R. W. Prager, and A. H. Gee. 1999. Regularised marching tetrahedra: Improved iso-surface extraction. *Computers & Graphics*, 23(4):583–598.

Manoj Kumar Tripathi, Kirti Chandra Sahu, and Rama Govindarajan. 2015. Dynamics of an initially spherical bubble rising in quiescent liquid. *Nature Communications*, 6(1):6268.

G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y. J. Jan. 2001. A Front-Tracking Method for the Computations of Multiphase Flow. *Journal of Computational Physics*, 169(2):708–759.

Gretar Tryggvason, Ruben Scardovelli, and Stéphane Zaleski. 2011. *Direct Numerical Simulations of Gas-Liquid Multiphase Flows*. Cambridge University Press, Cambridge.

Salih Ozen Unverdi and Grétar Tryggvason. 1992. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1):25–37.

Stéphane Vincent, Jorge César Brändle de Motta, Arthur Sarthou, Jean-Luc Estivalezes, Olivier Simonin, and Eric Climent. 2014. A Lagrangian VOF tensorial penalty method for the DNS of resolved particle-laden flows. *Journal of Computational Physics*, 256:582–614.

Duo Wang, Xiangmin Jiao, Rebecca Conley, and James Glimm. 2013. On the curvature effect of thin membranes. *Journal of Computational Physics*, 233:449–463.

Holger Wendland. 1995. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1):389–396.

G. D. Weymouth and Dick K. P. Yue. 2010. Conservative Volume-of-Fluid method for free-surface simulations on Cartesian-grids. *Journal of Computational Physics*, 229(8):2853–2865.

Sven Woop, Carsten Benthin, and Ingo Wald. 2013. Watertight ray/triangle intersection. *Journal of Computer Graphics Techniques (JCGT)*, 2(1):65–82.

Ilker O. Yaz and Sébastien Loriot. 2023. Triangulated surface mesh segmentation. In *CGAL User and Reference Manual*, 5.5.2 edition. CGAL Editorial Board.

I. Yoon and S. Shin. 2010. Tetra-marching procedure for high order Level Contour Reconstruction Method. In *AFM 2010*, pages 507–518, Algarve, Portugal.

Pengtao Yue, James J. Feng, Chun Liu, and Jie Shen. 2004. A diffuse-interface method for simulating two-phase flows of complex fluids. *Journal of Fluid Mechanics*, 515:293–317.

Peng Zhang and Chung K. Law. 2011. An analysis of head-on droplet collision with large deformation in gaseous medium. *Physics of Fluids*, 23(4):042102.

Alexander Z. Zinchenko, Michael A. Rother, and Robert H. Davis. 1997. A novel boundary-integral algorithm for viscous interaction of deformable drops. *Physics of Fluids*, 9(6):1493–1511.

List of Figures

List of Tables