An octree-based adaptive semi-Lagrangian free surface flow solver

THÈSE N^{O} 7011 (2016)

PRÉSENTÉE LE 12 MAI 2016

À LA FACULTÉ DES SCIENCES DE BASE
CHAIRE D'ANALYSE ET DE SIMULATION NUMÉRIQUE
PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Viljami Henrikki LAURMAA

acceptée sur proposition du jury:

Prof. T. Mountford, président du jury
Prof. M. Picasso, directeur de thèse
Dr A. Caboussat, rapporteur
Dr S. Boyaval, rapporteur
Prof. J. Hesthaven, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2016

Some people never go crazy.
What truly horrible lives they must lead.
— Charles Bukowski

To my family…

Acknowledgements

First of all I would like to thank Professor Marco Picasso for giving me not only the opportunity to work within his group for these years but also being always fair, available for discussions, for giving me constructive criticism whenever needed and for leading this group with a great atmosphere. It has been a fantastic experience.

I would also like to thank Dr. Gilles Steiner, Dr. Alexandre Masserey from *Ycoor Systems SA* for allowing me to work with and contribute to the software *cfsFlow++* and in particular Dr Gilles Steiner, with whom we have had frequent discussions about how to best tackle implementation issues. I also appreciate the availability of Dr Alexandre Caboussat to discuss various topics and his precious corrections to this thesis.

I am grateful to Prof. Jan Hesthaven, Dr. Sébastien Boyaval, Dr. Alexandre Caboussat for being part of my jury as well as to Prof. Thomas Mountford for presiding it. I would like to thank the Swiss National Science Foundation for their financial support.

Thanks to a great atmosphere in the group throughout these years, I have enjoyed not only working on this thesis but also spending personal time with my (crazy) colleagues, Laurent Michel, Samuel Dubuis, Jonathan Rochat, Thomas Foetisch, Diane Guignard, Sylvain Vallaghe, Michel Flueck and the numerous other colleagues beyond the group. Thank you to all my friends, family, flatmates, strange acquaintances and Terry for making this experience a great one and supporting me.

I also acknowledge all the people working hard on all these open source projects that make a thesis much richer. To name a few projects, for example *FEniCS*, *PETSc*, *Python*, *Matplotlib*, *Seaborn*, *Pandas*, *Gmsh*, *Paraview*, *Atom* and many more.

Abstract

A numerical method based on an adaptive octree space discretization for the simulation of 3D free-surface fluid flows is proposed. The Navier-Stokes equations are solved with a time-splitting scheme, which decouples advection from diffusion/incompressibility. The advection step is solved with a semi-Lagrangian VOF-based scheme on the octree.
An interface prediction algorithm is used to refine the octree at the predicted location of the interface in order to ensure detail preservation. Subsequently, the fluid is advected and a coarsening algorithm adapts the mesh to avoid excess refinement in non-interfacial regions. SLIC and decompression algorithms are used for post-processing to limit numerical diffusion and correct numerical compression of the VOF function. The octree scheme allows anisotropy, refinement of interfacial cells to an arbitrary level and supports arbitrary complex domains. It does not require a 2:1 cell size ratio condition between adjacent cells. The octree is then coupled with a tetrahedral mesh on which we solve the second step of the splitting algorithm, the Stokes' equations. Numerical validation is done on both advection benchmark test cases and results are compared with the uniform cell grid scheme. Paddle-generated water waves are also simulated and results are compared with experimental water wave profile measurements.

First order finite element stabilization schemes for the time-dependent Stokes' equations are studied. A unified proof of stability and convergence of velocity and pressure for consistent and non-consistent PSPG schemes for the time-dependent Stokes' equations is given with explicit dependence on viscosity and stabilization parameter. The link between bubble enrichment and Pressure Stabilized Petrov-Galerkin (PSPG) schemes in the context of time-dependent Stokes' equations is discussed and two bubble-based PSPG-type schemes are studied. Different possibilities for stabilization parameters are discussed. Numerical comparisons are done to determine stability, convergence and conditioning issues associated with different PSPG schemes, bubble-based schemes and local pressure projection schemes in different settings.

Key words: Finite elements, octree, semi-Lagrangian, free surface flows, VOF, SLIC, advection, time-splitting, Stokes, Navier-Stokes, transient, time-dependent, PSPG, local pressure projection, bubble

Résumé

Une méthode numérique basée sur une discrétisation adaptative en espace de type octree est proposée et appliquée à des simulations à surface libre en 3D. Les équations de Navier-Stokes sont résolues avec un algorithme de splitting en temps qui découple l'advection de la diffusion/incompressibilité. L'advection est résolue avec un schéma numérique semi-Lagrangien avec discrétisation de type VOF sur l'octree.
Un algorithme de prédiction d'interface est utilisé pour raffiner l'octree à la position prédite de l'interface pour conserver les détails. Ensuite, le fluide est transporté et un algorithme de déraffinage adapte le maillage pour éviter un raffinement trop important dans les régions non-interfaciales. Des algorithmes SLIC et de décompression sont utilisés pour faire un post-processing qui a pour but de limiter la diffusion numérique et la compression numérique. Le schéma basé sur l'octree supporte l'anisotropie, le raffinage jusqu'à un niveau arbitraire et les domaines complexes. Il ne requiert pas de ratio de taille 2 : 1 entre les cellules adjacentes. L'octree est ensuite couplé à un maillage en tetraèdres pour résoudre la deuxième partie de l'algorithme de splitting, les équations de Stokes. Une partie de la validation numérique est faite sur des cas tests de transport pur et les résultats sont comparés avec le schéma avec grille uniforme structurée. Des vagues générées par un piston pneumatique sont aussi simulées et les résultats sont comparés avec des données expérimentales en cuves réelles.

Des méthodes de stabilisation du premier ordre pour les équations de Stokes évolutives sont étudiées. Une preuve de stabilité et de convergence de la vitesse et de la pression pour les schémas PSPG consistents et non-consistents pour les équations de Stokes évolutives est donnée en détaillant les dépendances de la viscosité et du paramètre de stabilisation. Un lien entre l'élément bulle et les schémas PSPG est explicité dans le cadre des équations de Stokes évolutives et deux schémas de type PSPG basés sur l'élément bulle sont étudiés. Différentes possibilités pour le paramètre de stabilisation sont mentionnées. Des comparaisons numériques sont faites pour déterminer la stabilité, la convergence et les problèmes de conditionnement associés avec les schémas PSPG, bulle et Local Pressure Projection pour plusieurs cas tests.

Mots clefs : Eléments finis, octree, semi-Lagrangien, surface libre, VOF, SLIC, advection, transport, time-splitting, Stokes, Navier-Stokes, évolutif, PSPG, local pressure projection, bulle

Contents

List of Figures

Introduction

It is no accident that such a wide variety of researchers and engineers, from hydrologists, nuclear engineers to computer graphics researchers at movie studios all share the common interest of simulating three dimensional liquid behaviour. Indeed, the complexity of free surface fluid flows and difficulty of simulating them accurately make it an active area of research.

The goal here is to be able to accurately simulate three dimensional free surface flows and in particular, the generation, propagation and possibly breaking of water waves. Efficient methods are required to describe the transport of the liquid-air interface and to preserve its topological complexity throughout the simulation. This challenging task has been investigated using several different approaches yielding different tradeoffs between robustness, speed, stability and accuracy. Most schemes take either a particulate Lagrangian or a grid-based Eulerian approach.

Some Lagrangian methods such as smoothed particle hydrodynamics (SPH) [1, 2], grid based particle method (GBPM) [3] and moving least squares approximations (MLS) [4, 5] avoid explicitly describing the free surface altogether by using a particle-based approach which leads to a natural handling of topological changes in the fluid bulk. The free surface then has to be reconstructed for instance using the marching cubes algorithm [6, 7]. Other Lagrangian schemes such as front-tracking algorithms [8, 9] track the free surface with connected marker particles at the interface whereas the flow field is computed on a stationary grid. As a consequence, the interface is tracked accurately but topological changes in the interface can require complex algorithms to reconnect the marker particles correctly. The more recent Remeshed Particle Methods (RPM) [10] are semi-Lagrangian schemes based on following trajectories originating at grid points and projection is based on explicit remeshing kernels yielding arbitrary order methods.

Level set methods [11, 12, 13, 14] are Eulerian methods which describe the interface implicitly by handling a function called the level set function. The level set function takes positive values inside the liquid, negative values outside the liquid and the interface is then given by the zero level set of that function. The setting yields a smooth description of the interface and accurate normals but the original algorithm suffers from a fluid mass loss when solving the advection equation [15]. Subsequent works have significantly improved that aspect using e.g. massless

marker particles [16]. A review of many of these methods can also be found in [17, 18, 19]. Another Eulerian approach for resolving free surface flows consists in using a two-phase flow model where the flow is resolved both in the liquid and the air. The efficient capturing of a discontinuous interface by a continuous field then requires mesh-adaptive methods such as in [20, 21].

Volume of fluid (VOF) methods [22, 23, 24] are popular Eulerian schemes that track the fluid volume on a grid using a discretization of the discontinuous characteristic function. This function φ, called the volume fraction or color function, takes value one inside the liquid and zero outside and interfacial cells satisfy $0 < \varphi < 1$. Interface breakup and reconnection require no ad hoc treatment with this approach. Volume conservation is naturally maintained through the advection of φ given by the equation

$$\frac{\partial \varphi}{\partial t} + v \cdot \nabla \varphi = 0$$

where v is the velocity of the fluid. Since only a scalar volume fraction is stored for interfacial cells, the geometric reconstruction of the interface requires specific treatment. The simple line interface calculation (SLIC) algorithm introduced in [25] consists in reconstructing the interface with axis-aligned segments (2D) or rectangles (3D). Improved versions were developed in [22, 26]. Several higher order reconstruction techniques were subsequently developed [27, 24, 28].Although it has been shown that SLIC algorithms generate spurious flotsam in some scenarios [29], higher order reconstructions such as PLIC typically require computing a normal for the reconstructed plane and its location. Finding the location of the plane involves solving an inverse problem with the constraint that the volume enclosed by the plane and the interfacial cell should equal φh^3 where h is the cell size (3D) [30, 28].

The use of a fine structured cartesian grid for fluid tracking requires a prohibitive amount of cells for fine-scale simulations. This limitation can be overcome using dynamic adaptive meshing. A natural and efficient approach to achieve this is with an octree structure [31]. An octree structure is a hierarchical 3D structure based on an axis-aligned hexahedron which is split into eight hexahedra and where each hexahedron can be split into eight further hexahedra or remain unsplit, resulting in a tree-like structure. The octree structure allows efficient refining, coarsening, access to neighbour, parent and children cells, as well as traversal of all cells and retrieval of a cell containing a specified point. Free surface flow octrees solvers have been combined either with the level set method [32, 33, 34] or with a VOF approach [35].

In Chapter 1 we describe a 3D numerical scheme for solving the time dependent advection equation using a VOF discretization on an adaptive structured grid. The solver differs from standard VOF solvers in the sense that instead of a flux-based advection, we use an explicit unconditionally stable semi-Lagrangian scheme. The scheme was introduced in [36] for fixed structured grids and has successfully been used to simulate bubbles with surface tension computation [37] and viscoelastic fluid flows [38] when coupled with the mass and momentum equations. The semi-Lagrangian nature of the scheme bears some similarity with RPM [10]

and Vortex methods [39]. In our scheme, cells are advected along their characteristics and projected back on the grid after a refinement step designed to preserve detail at the interface. The characteristics based time-stepping scheme is not subject to the usual CFL condition but allows for CFL numbers larger than one. Interfacial cells are refined up to an arbitrary level of accuracy at every timestep, hence preserving detail at the interface while coarsening cells away from the interface. This new scheme allows a more accurate capture of the free surface and does not require a $2:1$ cell size ratio between neighbours (or equivalently at most one hanging node per cell face) as some operator discretization schemes require, see for instance [34].

A SLIC algorithm is used to avoid costly reconstruction computations at the interface and enable fast projection on the octree. An improved smoothing based decompression scheme is implemented to limit numerical compression which occurs during projection. Given a complex domain description, the octree can be refined to capture the domain boundary accurately using flags to indicate which cells are inside the computational domain. Handling of anisotropic cells with arbitrary aspect ratio is also supported in our implementation.

In order to simulate free surface fluid flows, we then solve the full Navier Stokes equations coupled with the previously described advection equation on φ which describes the movement of the fluid particles with the fluid velocity. To do this, we use the splitting scheme described in [40] and we decouple the advection part from other phenomena, such as diffusion, incompressibility and effect of external forces. This time splitting with a structured cartesian mesh has been successfully used in [41, 36, 42, 37, 38, 43, 44] and we adapt the scheme to replace the structured grid with an adaptive octree.

We solve advection equations for both φ and the velocity on the octree and the Stokes equations on a tetrahedral mesh. This choice is motivated by the ease of capturing complex domains and boundary normals with tetrahedral meshes and it also allows coarse elements for solving the Stokes equation and fine interfacial cells for capturing the liquid domain accurately. Interpolation of the liquid characteristic function φ from the octree to the tetrahedral mesh is performed in order to determine which tetrahedra are liquid and therefore included in the solution of the Stokes problem. Velocity is interpolated as well and the Stokes equations are then solved. Only the velocity field is interpolated back onto the octree for the next timestep.

Chapter 2 shows numerical results for the octree scheme. The first part validates the free surface displacement scheme for the transport equation given a velocity field using standard benchmarks. For the second part, paddle-generated water waves in a tilted cavity are simulated and compared agains experimental data kindly provided by the *VAW* at *ETH Zürich* available in [45]. Wave generation, propagation and breaking are all simulated. It is then shown that fully three-dimensional waves can be simulated with the octree scheme.

Solving Stokes' equations with finite elements requires an inf-sup stable velocity-pressure finite element space pair. We are specifically interested in solving the time-dependent Stokes' equations. $\mathbb{P}_2 - \mathbb{P}_1$ and the bubble-enriched $\mathbb{P}_1 b - \mathbb{P}_1$ space pairs are known to be inf-sup

stable unlike the $\mathbb{P}_1 - \mathbb{P}_1$ space pair [46]. They however also yield larger linear systems and consume more memory than using simply $\mathbb{P}_1 - \mathbb{P}_1$ finite elements, therefore it is of interest to study $\mathbb{P}_1 - \mathbb{P}_1$ stabilization procedures. Many of them have been described and we attempt to answer the question of which one should be preferably used and when. Streamline Upwind Galerkin (SUPG) stabilization [47], the Brezzi-Pitkäranta stabilization [48] and the Galerkin Least Squares (GLS) stabilization [46] have been used for the stationary Stokes' equations and many possibilities exist to extend their use to time-dependent Stokes equations. They fall into a category that is also called Petrov-Galerkin Pressure Stabilized (PSPG) methods that encompasses stabilizations where a possibly truncated momentum equation residual is integrated against the pressure test function gradient and possibly also the velocity test function, then multiplied by a stabilization parameter.

We choose to study residual-based PSPG schemes with or without the time derivative term which we call consistent and non-consistent PSPG schemes respectively and with or without the same contribution to the momentum equation. Another question we shall attempt to answer is the question of the choice of the stabilization parameter. In [49, 50], a link has been established between bubble enriched stabilization and PSPG schemes in the context of the stationary Stokes equations by eliminating the bubble variable. For the time-dependent Stokes' equations, it is not strictly possible to eliminate the bubble as far as we know but we propose two possible schemes to solve a smaller linear system. The bubble elimination scheme is proposed, in which we ignore the bubble terms from the previous timestep. A bubble reconstruction scheme is also proposed, in which we keep the bubble terms from the previous timestep, and reconstruct them for the current timestep after solving the linear system. By eliminating the bubble we recover a PSPG-type scheme with a stabilization parameter involving the timestep called the transient stabilization parameter. Making the quasi-static assumption [51] on the bubbles, we recover the classic $\frac{H^2}{\nu}$ stabilization parameter, that we call the spatial stabilization parameter, where H is mesh size and ν the viscosity. Note that this link differs from the work in [51] since no assumption is made on the expression of the bubble term.

Other types of stabilizations, including orthogonal subscales stabilization [52, 53] and local pressure projection [54, 55] have been introduced as well.

In [56], a proof of stability and convergence of a general class of symmetric stabilizations for the time-dependent Stokes' equations including the Brezzi-Pitkäranta stabilization has been given. It resembles a non-consistent PSPG scheme although the force term is absent. In [57], stability and convergence for the velocity in the fully discrete case has been proven for the consistent PSPG stabilization. Stability of the pressure is also proven and the L^2 convergence is mentioned but the proof is omitted. In [58], a semi-discrete analysis proves stability and convergence for both velocity and pressure in the L^2 norm for the consistent PSPG scheme and for the velocity in the fully discrete case. We give a unified proof for both consistent and non-consistent PSPG stabilization schemes assuming the stability condition $\Delta t \geq \alpha_K$ where Δt is the timestep and α_K the stabilization coefficient. In the proof it is also assumed

that the stabilization coefficient is larger than H^2 where H is the mesh size to ensure spatial stability. We keep track of the viscosity and stabilization parameters and express stability and convergence bounds with respect to them.

At the end of Chapter 3, we provide a comparison of consistent and non-consistent PSPG schemes for different stabilization parameters, local pressure projection and bubble enriched schemes on two time-dependent Stokes test cases and one full Navier-Stokes wave simulation from Chapter 2. It is determined that transient stabilization parameters provide spatially unstable solutions although they seem to allow the correct time evolution of the solution otherwise. With a spatial stabilization parameter in the large timestep case, it seems that both consistent and non-consistent PSPG schemes converge in a similar fashion as is proven in the Chapter 3. In the small timestep case, it seems that the non-consistent PSPG scheme converges to the wrong solution and the consistent one yields a singular matrix. Bubble enrichment and local pressure projection both provide stable and convergent solutions. In the context of full Navier-Stokes equations, it seems that stabilization parameters should be divided by the local Reynolds number and the local pressure projection scheme also requires a parameter to set for correct results. Bubble enrichment, bubble elimination and bubble reconstruction all give good results. The bubble reconstruction scheme seems to be the most reliable and performant method overall. Indeed, the linear system to solve is of the same size as a $\mathbb{P}_1 - \mathbb{P}_1$ scheme at the cost of storing one extra memory word per element and a negligible running time overhead of reconstructing the bubble terms for which we have an explicit form.

1 Octree-based numerical scheme for Navier Stokes free surface flows

In [36], a time splitting method on two grids was proposed to solve the time-dependent incompressible Navier Stokes equations for free surface flows. The advection is handled by a semi-Lagrangian scheme on a structured grid. In this Chapter, we propose a more efficient octree scheme to solve advection. A description of the scheme along with numerical results has also been published in [59]. This octree scheme can be coupled with a Stokes solver in order to solve the time-dependent incompressible Navier Stokes equations with free surface flows.

1.1 The advection equation

A VOF approach is introduced for the description of the liquid domain. Let $\Lambda \subset \mathbb{R}^3$ be the bounded computational domain containing the liquid at all times $t \in [0, T]$ where $T > 0$ is the final time of simulation. Let $\Omega(t) \subset \Lambda$ be the region occupied by the liquid at time t.

Let $\varphi : \Lambda \times [0, T] \to \mathbb{R}$ the characteristic function of the liquid, in other words the volume fraction of liquid which equals one if liquid is present and zero if not. Let $v : \Lambda \times [0, T] \to \mathbb{R}^3$ be a given velocity field. At this point it is assumed that v is known for all $x \in \Lambda$. Later, in Section 1.6, v is known only in the liquid domain. As an initial condition, we set the initial liquid domain $\Omega(0) = \{x \in \Lambda : \varphi(x, 0) = 1\}$. The liquid domain $\Omega(t)$ is then defined as $\{x \in \Lambda : \varphi(x, t) = 1\}$.

Since the fluid particles move at velocity v, the time evolution of φ is given by the transport equation

$$\frac{\partial \varphi}{\partial t} + v \cdot \nabla \varphi = 0 \quad \text{in } \Lambda \times [0, T]. \tag{1.1}$$

Let $X : [0, T] \to \mathbb{R}^3$ be the characteristics (trajectories of the fluid particles) defined by

$$\dot{X}(t) = v(X(t), t) \quad \forall t \in [0, T], \tag{1.2}$$

the solution of (1.1) then satisfies

$$\varphi(X(t),t) = \varphi(X(0),0) \quad \forall t \in [0,T].$$

Let $\Delta t = \frac{T}{N}$ be the timestep with $N > 0$ an integer. The time discretization points are $t^n = n\Delta t,\ n = 0,1,...,N$. Let $X(t;x,s)$ be the solution of (1.2) with $X(s) = x$. We then have

$$\varphi(X(t^{n+1};x,t^n),t^{n+1}) = \varphi(x,t^n) \quad \forall x \in \Lambda. \tag{1.3}$$

$X(t^{n+1};x,t^n)$ will be approached by a p-th order numerical approximation $X_p^{n+1}(x,t^n)$. We can use for example an explicit first order Euler method

$$X_1^{n+1}(x,t^n) = x + \Delta t\ v(x,t^n) \tag{1.4}$$

or a more accurate classical 4-th order Runge Kutta method noted $X_4^{n+1}(x,t^n)$.

$$X_4^{n+1}(x,t^n) = x + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \tag{1.5}$$

where

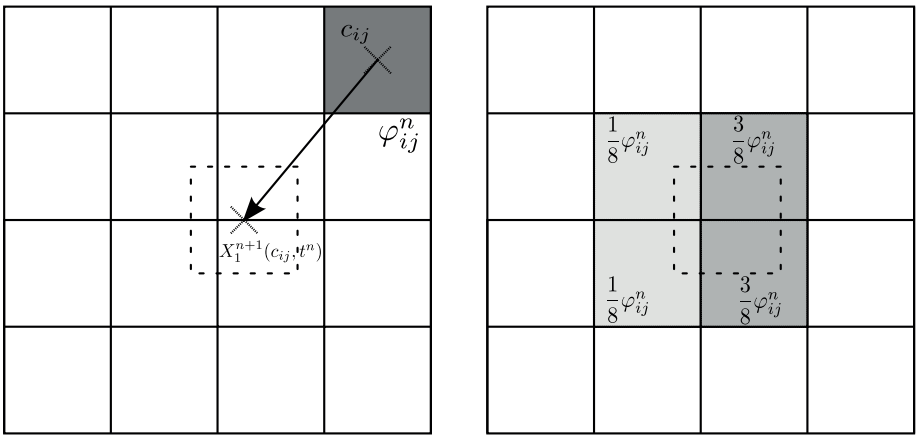
$$\begin{aligned}
k_1 &= \Delta t\ v(x,t^n)\\
k_2 &= \Delta t\ v(x+\frac{k_1}{2},t^n+\frac{\Delta t}{2})\\
k_3 &= \Delta t\ v(x+\frac{k_2}{2},t^n+\frac{\Delta t}{2})\\
k_4 &= \Delta t\ v(x+k_3,t^n+\Delta t).
\end{aligned}$$

Let φ^n and Ω^n be approximations of φ and Ω respectively at time t^n. The time discretization is as follows; we solve the advection problem (1.1) between t^n and t^{n+1} by translation of cells with the method of characteristics and a projection detailed in Section 1.2 to obtain φ^{n+1} which yields a new liquid domain $\Omega^{n+1} = \{x \in \Lambda : \varphi^{n+1}(x) = 1\}$.

Given an approximation $X_p^{n+1}(x,t^n)$ of $X(t^{n+1};x,t^n)$ and an approximation of $\varphi(\cdot,t^n)$, we compute φ^{n+1} with an approximation of (1.3) given by

$$\varphi^{n+1}(X_p^{n+1}(x,t^n)) = \varphi(x,t^n). \tag{1.6}$$

The difficulty resides in conciliating the Lagrangian approach of the approximation (1.6) with a Eulerian spatial discretization. This will be achieved by translating cells of the spatial discretization along their characteristics and projecting them on the mesh. In the sequel the projection of (1.6) is presented on a structured grid and then on the octree.



(a) Advection of φ^n_{ij}

(b) Projection onto the grid

Figure 1.1: φ^n_{ij} is transported along the characteristics and then projected back on the grid.

1.2 Advection on structured grid

The implementation of (1.6) on a uniform cartesian grid as in [36] is recalled. It consists in decomposing the domain into an axis-aligned cartesian structured grid with cell size h. Each cell is indexed by a 3-dimensional index ijk. The characteristic function φ at the center of cell ijk and at time t^n is denoted φ^n_{ijk}. Using (1.6), the advection step for cell ijk consists in advecting φ^n_{ijk} by $X_p^{n+1}(c_{ijk}, t^n)$ and projecting it back on the grid where c_{ijk} is the center of cell ijk. For any receiver cell D in the structured grid its φ value after projection is given by

$$\varphi^{n+1}(D) = \sum_{i,j,k} \frac{volume(D \cap X_4^{n+1}(\tilde{C}_{ijk}, t^n))}{volume(D)}$$

where $\tilde{C}_{ijk}$ represents the cell C_{ijk} shifted by $X_p^{n+1}(c_{ijk}, t^n)$ with c_{ijk} the cell center of C_{ijk}. Using (1.4), the advection step for cell ijk consists in advecting φ^n_{ijk} by $\Delta t\ v(c_{ijk}, t^n)$ and projecting it back on the grid where c_{ijk} is the center of cell ijk. A 2-dimensional illustration is given in Figure 1.1.

The CFL (Courant-Friedrichs-Levy) number is then the number of cells traversed by the fluid in one iteration. Given a velocity v, the CFL number is then given by $CFL = \frac{\|v\|\Delta t}{h}$. Typically in explicit finite volume solvers for the transport equation, a CFL condition is assumed [60], meaning that a restriction is imposed on the timestep to satisfy $\frac{\|v\|\Delta t}{h} \leq 1$. Thanks to the semi-Lagrangian nature of the scheme, no such condition is assumed here. Moreover the accuracy of the scheme clearly depends on the value of the CFL number, as reported in Section 2. Accurate simulations are obtained with CFL numbers ranging from below 1 to 20.

Let us analyze a simplified one-dimensional analog of the proposed scheme. Assume that the

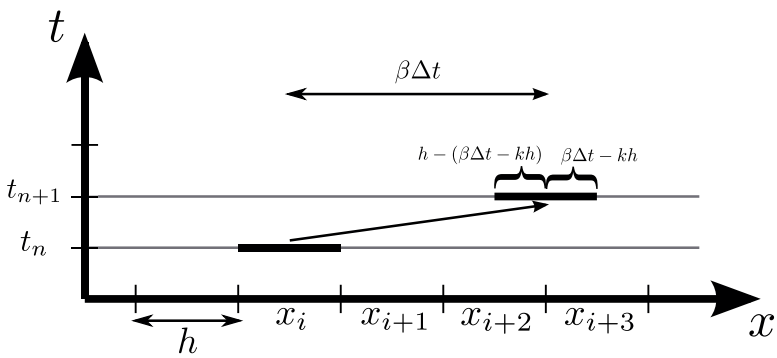


Figure 1.2: φ_i^n is transported along the characteristics and then projected back on the grid.

velocity is a constant $\beta > 0$. Assume that φ is continuously differentiable in space and in time. The problem is then the following. Find φ such that

$$\frac{\partial \varphi}{\partial t} + \beta \frac{\partial \varphi}{\partial x} = 0 \quad \text{in } \mathbb{R} \times [0, T] \tag{1.7}$$

with $\varphi(\cdot, 0) = \varphi^0$ a given initial condition.

Consider the numerical scheme illustrated in Figure 1.2. Each cell is translated along the characteristics for one timestep and projected on the intersecting cells. Let us decompose the spatial domain $\mathbb{R}$ into cells $[x_{i-1/2}, x_{i+1/2}[$ of uniform length $h = x_{i+1/2} - x_{i-1/2}$ for $i \in \mathbb{Z}$. Cell centers are denoted $x_i = (x_{i+1/2} + x_{i-1/2})/2$ for $i \in \mathbb{Z}$ and $k = \lfloor CFL \rfloor = \left\lfloor \frac{\beta \Delta t}{h} \right\rfloor$. In the illustration, $CFL = 2.5$, $k = 2$ and the projection of φ_i^n then contributes to φ_{i+2}^{n+1} and φ_{i+3}^{n+1} with weights of respectively $1 - \left(\frac{\beta \Delta t}{h} - k\right)$ and $\frac{\beta \Delta t}{h} - k$. The numerical scheme can thus be written as

$$\begin{cases} \varphi_i^{n+1} = \left(1 - \left(\frac{\beta \Delta t}{h} - k\right)\right) \varphi_{i-k}^n + \left(\frac{\beta \Delta t}{h} - k\right) \varphi_{i-k-1}^n & \forall n \geq 0,\ \forall i \in \mathbb{Z} \text{ with } k = \left\lfloor \frac{\beta \Delta t}{h} \right\rfloor \\ \varphi_i^0 = \varphi^0(x_i) \quad \forall i \in \mathbb{Z}. \end{cases} \tag{1.8}$$

The following convergence result is classical, its proof is repeated for the sake of clarity.

Proposition 1. *Let φ be the exact solution to the problem* (1.7) *and φ_i^n given by the numerical scheme* (1.8) *with initial condition $\varphi^0 \in \mathscr{C}^2(\mathbb{R})$. Then, there exists C independent of h and Δt such that*

$$\sup_{i \in \mathbb{Z}} \left| \varphi(x_i, t_n) - \varphi_i^n \right| \leq TC \left(\Delta t + h + \frac{h^2}{\Delta t} \right)$$

Proof. The exact solution to equation (1.7) is given by

$$\varphi(x + \beta t, t) = \varphi^0(x) \quad \forall x \in \mathbb{R},\ \forall t \in [0, T]$$

and is therefore twice continuously differentiable in space and time. Using a Taylor expansion, we get

$$\begin{aligned}
&\left(1-\left(\frac{\beta\Delta t}{h}-k\right)\right)\varphi(x_{i-k},t_n)+\left(\frac{\beta\Delta t}{h}-k\right)\varphi(x_{i-k-1},t_n)\\
=&\left(1-\left(\frac{\beta\Delta t}{h}-k\right)\right)\varphi(x_{i-k},t_n)+\left(\frac{\beta\Delta t}{h}-k\right)\left(\varphi(x_{i-k},t_n)-h\frac{\partial\varphi}{\partial x}(x_{i-k},t_n)+\frac{h^2}{2}\frac{\partial^2\varphi}{\partial x^2}(\xi,t_n)\,dx\right)\\
=&\varphi(x_{i-k},t_n)-\left(\Delta t\beta-kh\right)\frac{\partial\varphi}{\partial x}(x_{i-k},t_n)+\left(\frac{\beta\Delta t}{h}-k\right)\frac{h^2}{2}\frac{\partial^2\varphi}{\partial x^2}(\xi,t_n) \qquad (1.9)
\end{aligned}$$

for some $\xi\in[x_{i-k-1},x_{i-k}]$. Using another Taylor expansion and then (1.7), we also obtain

$$\begin{aligned}
\varphi(x_i,t_{n+1})&=\varphi(x_{i-k},t_n)+\Delta t\frac{\partial\varphi}{\partial t}(x_{i-k},t_n)+kh\frac{\partial\varphi}{\partial x}(x_{i-k},t_n)+\tilde{r}_{h,\Delta t,\varphi}\\
&=\varphi(x_{i-k},t_n)-\left(\Delta t\beta-kh\right)\frac{\partial\varphi}{\partial x}(x_{i-k},t_n)+\tilde{r}_{h,\Delta t,\varphi} \qquad (1.10)
\end{aligned}$$

where $\tilde{r}_{h,\Delta t,\varphi}$ is a remainder such that $\left|\tilde{r}_{h,\Delta t,\varphi}\right|\le C(\Delta t^2+\Delta t h+h^2)$ where C depends only on the second derivatives of φ. Combining (1.9) and (1.10), we get

$$\varphi(x_i,t_{n+1})=\left(1-\left(\frac{\beta\Delta t}{h}-k\right)\right)\varphi(x_{i-k},t_n)+\left(\frac{\beta\Delta t}{h}-k\right)\varphi(x_{i-k-1},t_n)+r_{h,\Delta t,\varphi} \qquad (1.11)$$

where $r_{h,\Delta t,\varphi}$ is a remainder such that $\left|r_{h,\Delta t,\varphi}\right|\le C(\Delta t^2+\Delta t h+h^2)$ where C is a constant that depends on φ. Let $e_i^n=\varphi(x_i,t_n)-\varphi_i^n$ for $i\in\mathbb{Z}$ and $n\ge 0$. From (1.12) and (1.8), we get

$$e_i^{n+1}=\left(1-\left(\frac{\beta\Delta t}{h}-k\right)\right)e_{i-k}^n+\left(\frac{\beta\Delta t}{h}-k\right)e_{i-k-1}^n+r_{h,\Delta t,\varphi} \qquad (1.12)$$

By induction, we can prove that for any $n>0$,

$$e_i^n=\sum_{j=0}^{n}\binom{n}{j}\left(1-\left(\frac{\beta\Delta t}{h}-k\right)\right)^{n-j}\left(\frac{\beta\Delta t}{h}-k\right)^{j}e_{i-nk-j}^0+n\,r_{h,\Delta t,\varphi} \qquad (1.13)$$

and since we have $\sum_{j=0}^{n}\binom{n}{j}\left(1-\left(\frac{\beta\Delta t}{h}-k\right)\right)^{n-j}\left(\frac{\beta\Delta t}{h}-k\right)^{j}=1$ and $n\Delta t=T$, (1.13) implies that

$$\left|e_i^n\right|\le\sup_{l\in\mathbb{Z}}\left|e_l^0\right|+TC\left(\Delta t+h+\frac{h^2}{\Delta t}\right). \qquad (1.14)$$

Choosing $\varphi_i^0=\varphi(x_i,0)$ concludes the proof. □

Remark 1. *This convergence result shows that to get the optimal order, Δt needs to decrease as h. The error bounds are consistent with the $\left(\Delta t+h+\frac{h^2}{\Delta t}\right)$ error bound in [61] for the characteristics method in the context of finite elements.*

It is assumed for this result that φ is smooth, which will not be the case when transporting

the volume of fluid characteristic function, however the timestep Δt with respect to h will be chosen based on this result. In Chapter 2, we will perform numerical validation of the scheme and show evidence of convergence of the scheme even in a non-smooth context with interface reconstruction and non-constant velocity field. Note that the scheme error is a linear function of the maximal initial error and of final time.

The proof sheds some light as to how the error depends on the error previous timesteps. Indeed, the induction result (1.13) shows that for any $k, n > 0$ the error at time $n+k$ is a weighted sum of errors at timestep k where the coefficients are given by the probability mass function of a binomial distribution $\mathscr{B}\left(n, \frac{\beta\Delta t}{h} - \left\lfloor \frac{\beta\Delta t}{h} \right\rfloor\right)$.

Another parallel can be drawn with convolution kernels in image processing. Convolution kernels are a well-known technique in image processing, see for instance [62]. Given an image represented as an array of pixels and a matrix of weights called the kernel, the image is processed by replacing each pixel with a linear combination of the original pixels with weights given by the kernel centered at the pixel of interest. Different choices of the kernel can yield for instance to Gaussian blur or edge detection effects on the image. For constant velocity in a 2 dimensional setting, one step of the numerical scheme above is then exactly a kernel convolution with a kernel containing decentered weights. With non-constant velocity, the kernel would depend on the location of the pixel. Although this has not been studied here, this parallel opens new possibilities for fast GPU implementations since pre-existing fast kernel convolution codes can be used, such as described in [63].

The large memory consumption of the structured grid is a drawback that becomes apparent when running large scale simulations. These high memory requirements motivate the switch from a structured Cartesian grid to a more flexible adaptive structure. Nonetheless, we would like to have an axis-aligned structure in order to preserve the computational speed of the projection step.

1.3 Octree definition

Structured adaptive grids have been successfully used in computational fluid dynamics. The Adaptive Mesh Refinement (AMR) framework described in [64] uses a base structured grid and overlays hierarchical patches of finer structured grids over regions with large error. Free-surface flow simulations have been performed at first on a 2-dimensional adaptive structure called quadtree [65], of which the octree is the 3D equivalent. Octree structures [31], [66] consist in embedding the computational domain in an axis-aligned hexahedron and splitting it into 8 hexahedral cells of equal dimensions. Each cell can then be recursively split into 8 further cells or kept as is. This structure allows full flexibility in the refinement level of each cell while satisfying our requirement to have an axis-aligned structured mesh. As illustrated in Figure 1.3, the process yields a graph which has the property of being a tree [67], hence the name octree. Previous works have used octree structures to solve the Euler and the Navier-

Stokes equations using either finite volume methods [68] or more commonly, finite difference methods [35], [69]. Implementations of octree-based Navier-Stokes free-surface flow solvers using finite differences have also been developed, either coupled with a level set method [33], [34] for surface-tracking or with a Volume Of Fluid (VOF) method [70]. Our scheme will also be using a VOF-type discretization but a semi-Lagrangian method for time-stepping instead of a classic VOF method.

Each node of the graph which is split in 8 further nodes will be called a *parent* node of its 8 *children* nodes. The 8 children nodes are numbered according to their Morton ordering [71] illustrated in Figure 1.3. Nodes that have no children are called *leaf* nodes and correspond to computational cells in the mesh.

We will derive a new scheme by replacing the structured hexahedral grid with an octree structure with VOF-type fluid tracking which will allow both accurate capture of the interface and coarsening of the mesh away from the interface, hence decreasing memory usage of the scheme. We eventually want to implement a transport equation solver on a non-structured grid using an explicit semi-Lagrangian scheme based on (1.6). Illustrations are done in 2D for simplicity but the numerical scheme is implemented in 3D.

The octree graph is illustrated along with its corresponding octree mesh in Figure 1.3. The base axis-aligned hexahedron corresponding to the base node in the octree graph is chosen to be the bounding box of the computational domain.

In the Figure 1.3, the level 0 consists of a grouping of 8 cells which we call an *oct*. In order to avoid having the bounding box dictate the anisotropy of cells, we instead subdivide the bounding box into an array of level 0 octs with respectively B_x, B_y, B_z octs in directions x, y, z as illustrated in 2D in Figure 1.4. This way, the aspect ratio of the cells can be selected and remains constant throughout the simulation. The array of level 0 cells defines the level 0 cell grid. Each subsequent subdivision adds one to the level of the corresponding cell. More formally, the level of a cell C is defined as m where m is the number of edges in the octree graph separating the node corresponding to C from a level 0 node as seen in Figure 1.3.

We choose the maximum refinement level of the octree l_{max} according to the desired accuracy of the spatial discretization at the interface. We also choose the level l_{liquid} which is the minimal level of liquid cells defined below according to the desired accuracy of the discretization of the velocity field.

The parameters B_x, B_y, B_z, l_{max} and l_{liquid} fully specify the cell aspect ratio, the smallest cell size and maximal liquid cell size and remain constant throughout the simulation. An example of advection with the octree with the chosen parameters $B_x = B_y = 1$, $l_{liquid} = 1$ and $l_{max} = 3$ is shown in Figure 1.5.

Numerical experiments have indicated that the choice $l_{liquid} = l_{max} - 2$ (i.e. the edge length of liquid cells is at most 4 times larger than the edge length of smallest cells) seems to be a

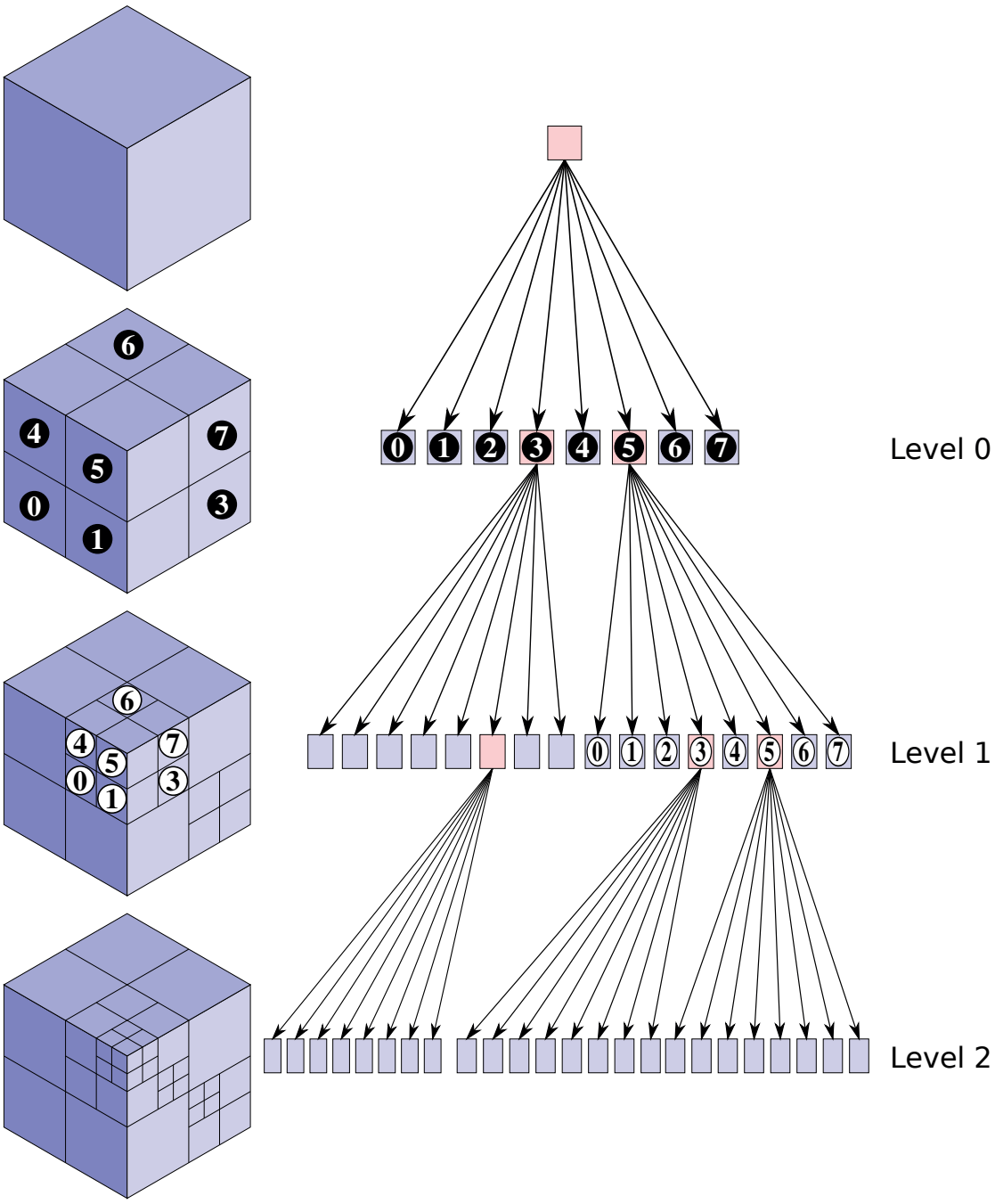
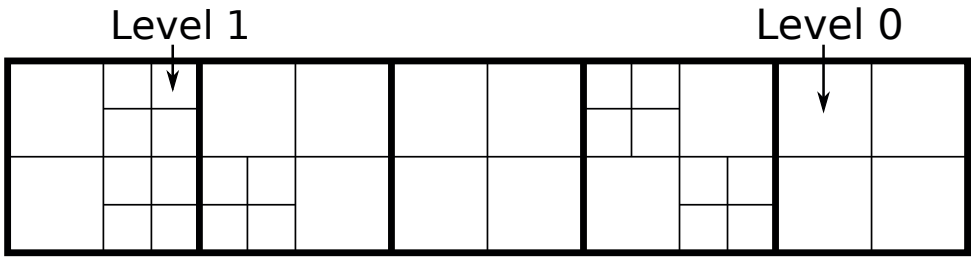
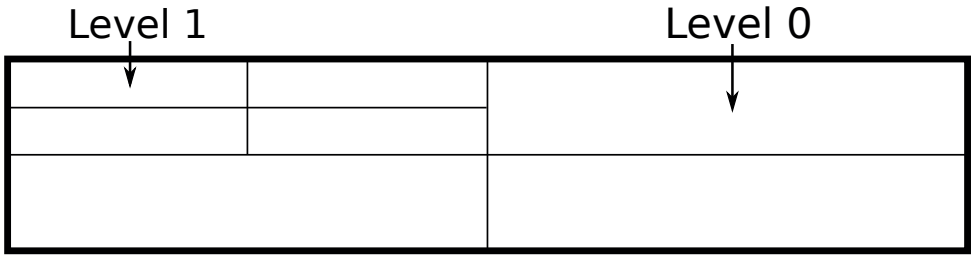


Figure 1.3: The octree structure as a mesh (left) and as a graph (right) with Morton ordering. Everytime a cell in the octree is split into 8 cells, the corresponding node in the graph gets 8 children nodes. The leaf nodes in the graph, shown in purple, are the computational cells in the mesh.



(a) Five base octs, $B_x = 5$, $B_y = 1$



(b) One base oct, $B_x = 1$, $B_y = 1$

Figure 1.4: Choosing B_x, B_y and B_z allows adjusting of cell anisotropy.
The level 0 corresponds to an oct divided in 8 in 3D, here in 4 in 2D for illustration purposes.

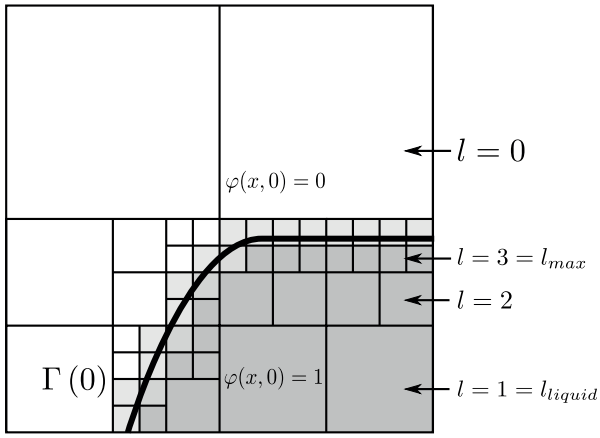
good one in the sense that it yields a good tradeoff between efficiency and error. This choice is discussed further in Section 2.

In Section 1.6.5 we will explain how complex domains can be represented by refining the octree at the boundary and using flags to set cells to inside or outside the domain.

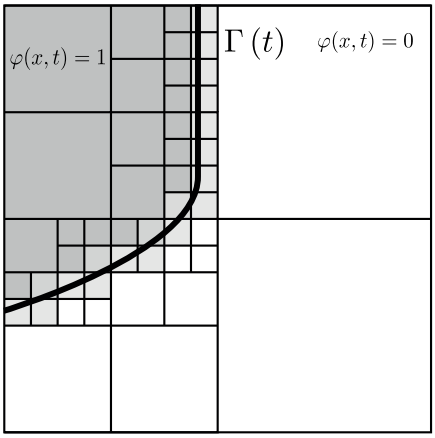
1.4 Advection on octree grid

Our goal is to implement (1.6) on an octree rather than a structured grid. Let us denote by $\mathscr{D}^n$ the octree mesh at time t^n which we will see how to initialize. At time t^n, to each cell of the octree mesh $C \in \mathscr{D}^n$ are associated the fields $\varphi^n(C)$ and $v^n(C)$ which are piecewise constant approximations of respectively $\varphi(\cdot, t^n)$ and $v(\cdot, t^n)$ at the center of the cell C. An octree advection step starts with the octree grid $\mathscr{D}^n$. A prediction algorithm refines the octree so as to preserve interfacial detail and yields the refined grid $\mathscr{D}^{n+1/2}$. In the advection step, φ is advected and φ^{n+1} is defined on the grid $\mathscr{D}^{n+1/2}$. φ^{n+1} is then post-processed in the decompression step to ensure $0 \leq \varphi^{n+1} \leq 1$. Finally, the octree is coarsened which yields the grid $\mathscr{D}^{n+1}$.

Algorithm 1 gives an outline of the scheme which we will detail in the following subsections.



(a) Time $t = 0$

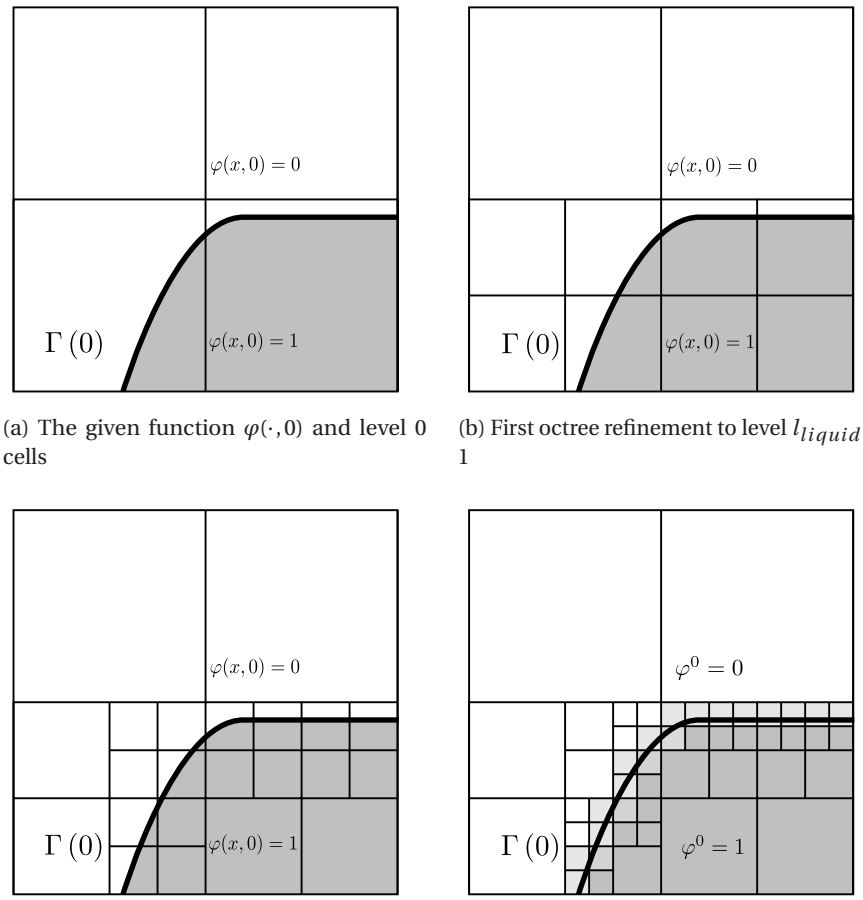


(b) Time t

Figure 1.5: Example of octree refinement around the interface $\Gamma(t)$. Interfacial cells are level $l_{max} = 3$, liquid cells are level $l_{liquid} = 1$ and empty cells are level $l = 0$.

Algorithm 1 Summary of the octree advection

1. Initializing the octree
2. Iterate for $n = 0, 1, 2, ...$
   (a) Interface prediction refinement
   (b) Advection
   (c) Decompression
   (d) Coarsening



(a) The given function $\varphi(\cdot,0)$ and level 0 cells

(b) First octree refinement to level $l_{liquid} = 1$

(c) Second octree refinement at the interface to level 2

(d) Interfacial cells reach level $l_{max} = 3$ and φ^0 is defined

Figure 1.6: 2D representation of the octree initialization with $l_{liquid} = 1$ and $l_{max} = 3$.

1.4.1 Initialization

Given the initial condition $\varphi(\cdot,0) : \Lambda \to \mathbb{R}$, we define the initial liquid domain $\Omega(0) = \{x \in \Lambda | \varphi(x,0) = 1\}$. At first the initial octree mesh $\mathscr{D}^0$ consists of the bounding box subdivided into a user-specified grid of level 0 cells as explained in Figure 1.4. For now, we suppose that the bounding box of the computational domain is the computational domain itself but it is possible to extend the formulation to more complex domains, see end of Section 1.5.

The initialization process is illustrated in Figure 1.6. Let us denote by x_i^C the i-th vertex of cell C, $i = 0,...,7$, and by x_m^C the center of cell C. We recursively refine cells intersecting the region of the space $L^+ = \{x \in \Lambda \mid \varphi(x,0) > 0.5\}$ (superlevel set) until every such cell C is at level l_{liquid} as shown in Figure 1.6b. We say that a cell is intersecting L^+ if at least one of its vertices x_i^C, $i = 0,...,7$ satisfies $\varphi(x_i^C,0) > 0.5$.

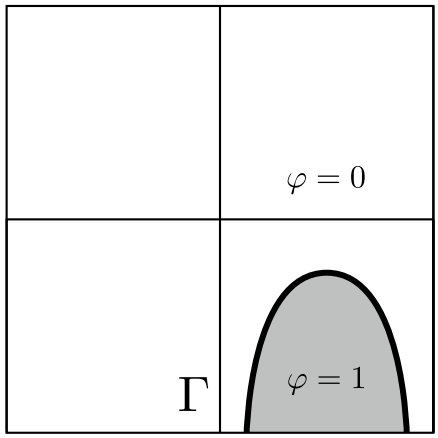


Figure 1.7: Case when the liquid domain fails to be detected by simple evaluation of φ_0 at the cell vertices.

Now that all liquid cells are refined to a desired level l_{liquid}, we refine to level l_{max} the cells intersecting the interface as shown in Figures 1.6c and 1.6d. Let us define the level set which corresponds to the initial interface

$$L = \left\{ x \in \Lambda \mid \varphi(x, 0) = 0.5 \right\}.$$

We say that a cell is intersecting the level set L if at least one of its vertices $x_i^C,\ i = 0,..,7$ satisfies $\varphi(x_i^C, 0) \geq 0.5$ and another vertex x_j^C satisfies $\varphi(x_j^C, 0) < 0.5$. We recursively refine all cells in $\mathscr{D}^0$ intersecting L until all such cells are at level l_{max}.

After the octree refinement, we define $\varphi^0(C)$ for each cell $C \in \mathscr{D}^0$. We set $\varphi^0(C) = 1$ if C intersects L^+ and is at level l_{liquid}, we set $\varphi^0(C) = 0.5$ if C intersects L and otherwise we set $\varphi^0(C) = 0$. This initialization method is a choice and we have not observed major differences with initializing interfacial cells to $\varphi^0(C) = 0$ or $\varphi^0(C) = 1$.

Evaluating $\varphi(\cdot, 0)$ on the vertices might not be sufficient to capture the interface if the fluid region does not contain any vertex of any level 0 cell as illustrated in Figure 1.7. The user can specify a finer initial grid with larger parameters B_x, B_y, B_z to capture the initial liquid region. Alternatively it is also possible to sample a larger number of points inside a cell to determine if it intersects L^+.

1.4.2 Prediction

A crucial feature of the numerical scheme is detail preservation at the interface and we present a procedure to ensure that the interface is always captured by fine cells. Note that it is possible to extend the algorithm to refine the octree in the bulk of the fluid in regions of high velocity gradients for better accuracy which could arise for instance in bottleneck situations. No such

extension is considered in this work since we focus on detail preservation at the interface but this could be implemented given a refinement criterion. We first define precisely what we mean by liquid and interfacial cells. A cell $C \in \mathscr{D}^n$ is called *liquid* if $\varphi^n(C) \geq 0.5$. A cell $C \in \mathscr{D}^n$ will be called *interfacial* if it is at level l_{max} and shares a face with a neighbouring cell C_n such that one of the following conditions is satisfied. Either C is liquid while C_n is non-liquid, or C is non-liquid while C_n is liquid.

Before transporting φ, it is necessary to refine the octree in regions that will receive interfacial cells. We achieve this by doing a first prediction pass as follows.

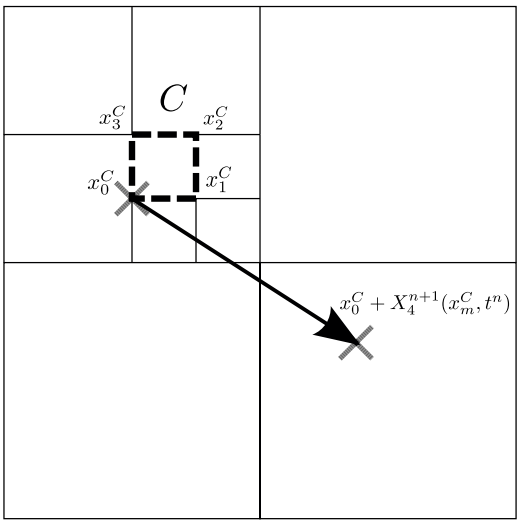
$\mathscr{D}^n$ will be modified to ascertain sufficient refinement of receiver cells in the advection process and yield the refined grid $\mathscr{D}^{n+1/2}$. $\mathscr{D}^{n+1/2}$ will be receiving advected cells but we still need $\mathscr{D}^n$ to advect cells from. Let $C \in \mathscr{D}^n$, l_C the level of cell C, x_m^C the center of C and x_i^C for $i = 0, ..., 7$ the vertices of C. Again, $X_4^{n+1} : \Lambda \times [0, T] \to \mathbb{R}^3$ gives an approximation of the displacement of a fluid particle by following the characteristics for one timestep with velocity v using a classical 4th order Runge Kutta method, see Section 1.2. The refinement algorithm is illustrated in Figure 1.8 and described in Algorithm 2 which yields $\mathscr{D}^{n+1/2}$, the refined octree.

Algorithm 2 Interface prediction refinement at time t^n

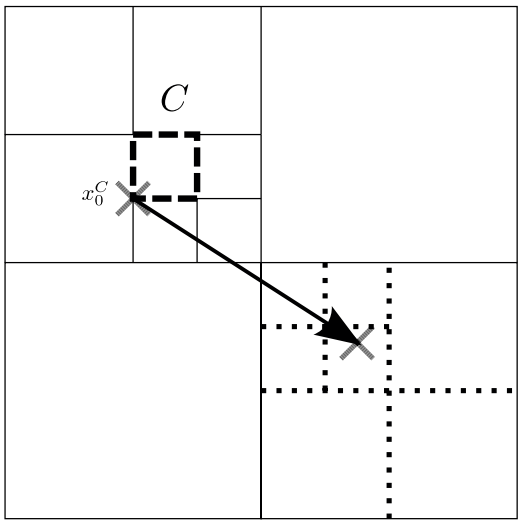
```
𝒟^{n+1/2} ← 𝒟^n
for all cell C ∈ 𝒟^n do
    for all vertex x_i^C of C do
        set receiver cell C_r to cell in 𝒟^{n+1/2} which
                        contains x_i^C + X_4^{n+1}(x_m^C, t^n)
        while level(C_r) < level(C) do
            split C_r in 𝒟^{n+1/2}
            set C_r to cell in 𝒟^{n+1/2} containing
                        x_i^C + X_4^{n+1}(x_m^C, t^n)
        end while
    end for
end for
```

This algorithm ensures that receiver cells in $\mathscr{D}^{n+1/2}$ are at least of same level as the cells in $\mathscr{D}^n$ to be projected. Note that the displacement vector $X_4^{n+1}(x_m^C, t^n)$ which approximates a piece of the characteristics is computed from cell center x_m^C and not from the vertex x_i^C in order to refine the mesh at the points where the cell will be projected during the advection phase. It is necessary to perform this refining algorithm on liquid cells and on non-liquid interfacial cells.

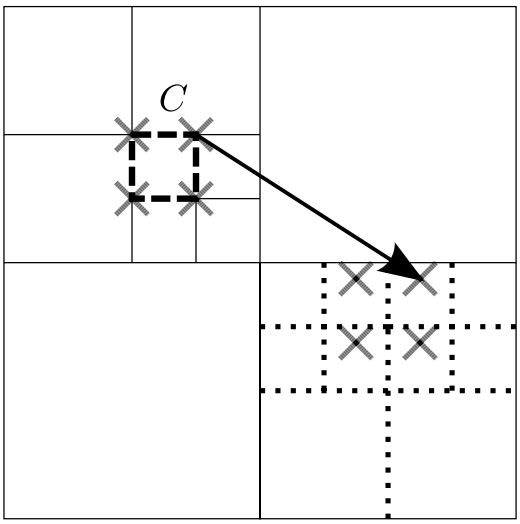
Non-liquid interfacial cells need to be treated for the following reason. When the interface coincides with cell faces (or cell edges in 2D), a full cell is neighbouring an empty cell. During initialization, if the initial φ function returns 0 when evaluated on vertices exactly on the interface, the full cell will be refined to l_{max}. However, if the evaluation returns 1, only the empty cell will be refined to l_{max}. If the empty cell is refined up to l_{max} and the full cell is coarse, we need to treat the empty cell to ensure that at the next step, interfacial cells remain



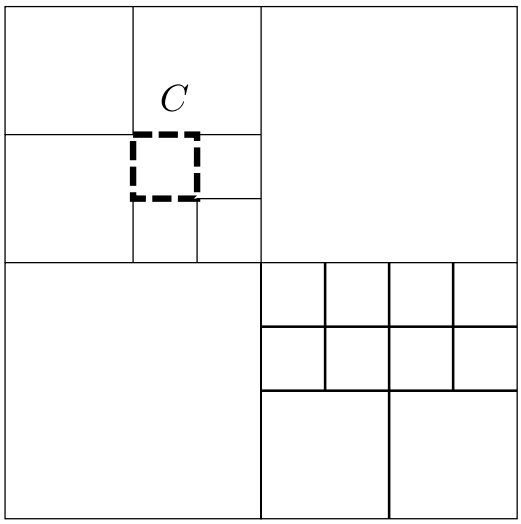
(a) Cell C in dashed lines, x_0^C the first vertex of C. We locate the cell containing $x_0^C + X_4^{n+1}(x_m^C, t^n)$ where x_m^C is the center of C



(b) We recursively refine the receiver cells in $\mathscr{D}^{n+1/2}$. Dotted lines indicate new refined cells up to level l_C in $\mathscr{D}^{n+1/2}$



(c) Result after repeating the process for each vertex of C



(d) $\mathscr{D}^{n+1/2}$ after applying the refining algorithm to C

Figure 1.8: Interface prediction refinement. Each vertex is translated with the displacement computed from the center of its associated cell with the Runge Kutta scheme. The cell containing the resulting point is refined until it has same level as the original cell.

at level l_{max}. This case can also arise during the simulation in rare cases.

1.4.3 Advection

In the refinement step we have ensured that cells from the octree $\mathscr{D}^n$ will be advected on cells of at least same level of refinement of the refined octree $\mathscr{D}^{n+1/2}$. All that is left to do is to translate cell hexahedra of $\mathscr{D}^n$ along characteristics and project them on $\mathscr{D}^{n+1/2}$. This scheme implements (1.6) on the octree and is used in [36] on structured grids.

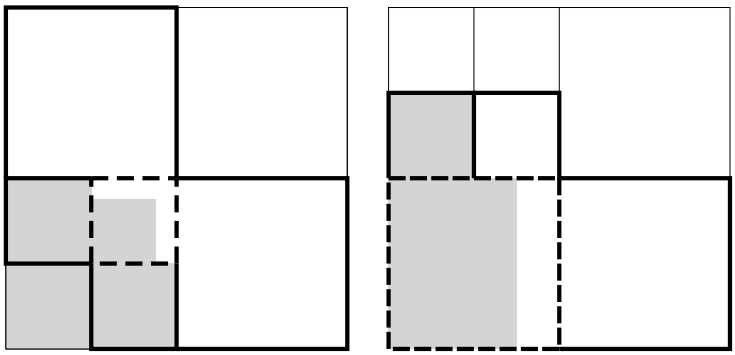
In a Eulerian setting such as ours, we use a semi-Lagrangian scheme where cells are transported using the velocity at their center and projected onto the octree. The projection scheme is necessary since the advected cell will rarely if ever coincide exactly with another cell. Let $C \in \mathscr{D}^n$ be a cell to be transported and projected. The scheme consists in transporting C first treated with a SLIC algorithm described below and finding the cells of $\mathscr{D}^{n+1/2}$ intersecting the transported cell. Each such cell will receive a contribution to its φ value weighted by the volume of the intersection.

Note that the scheme is unconditionally stable with respect to the Courant (or CFL) number and so we can choose arbitrarily large timesteps as long as the Runge Kutta scheme captures the characteristics to a desired level of accuracy. Some numerical schemes such as [35, 34] make use of a graded octree which requires a $2:1$ cell size ratio between neighbouring cells to simplify gradient and flux calculations. This requirement is not necessary here due to the natural handling of different cell sizes via projection.

The SLIC algorithm that we apply is a simple generalization of the SLIC algorithm presented in [36] which was adapted from [26]. Recall that the algorithm reduces numerical diffusion and consists in redistributing the fluid inside cells C with $0 < \varphi(C) < 1$ such that the fluid is "pushed" against faces shared with neighbouring liquid cells. We retrieve the values of $\varphi^n(C_a)$ where C_a are adjacent cells and according to the distribution of the values $\varphi^n(C_a)$ around the cell, we select the "pushing" pattern.

The choice of the SLIC algorithm and not one of the higher order reconstruction techniques such as PLIC [27, 24, 28] is motivated by computational speed. Indeed, computing the reconstructed hexahedral shape is straightforward for SLIC as is projecting the axis-aligned hexahedron on an axis-aligned hexahedral grid during the advection phase. For more advanced algorithms such as PLIC, an inverse problem has to be solved to compute the reconstruction [30, 28]. We may try to compensate the loss of accuracy for not using a PLIC-type method by refining the octree further, increasing l_{max}.

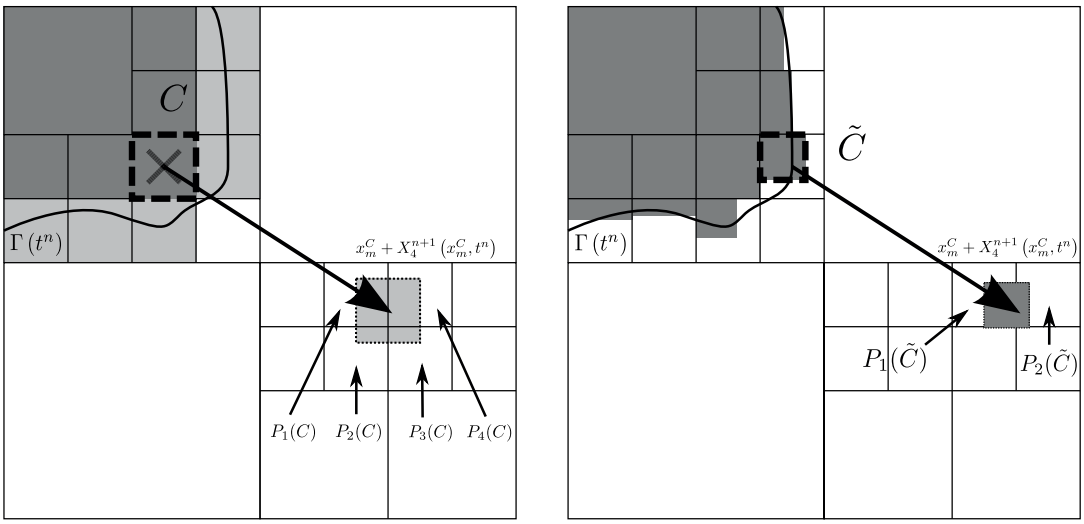
We can apply the same algorithm in the octree for cells having only adjacent cells of same level. The "pushing" patterns remain the same. Two more scenarios are possible however. The adjacent cells can either be of lower or of greater level. Note that the latter case can occur even though large cells are not interfacial. Indeed, due to the discretization error and the projection described in this Section, it can happen that $0 < \varphi(C) < 1$ for cells C with level $l < l_{max}$.



(a) SLIC with larger neighbour cells

(b) SLIC with smaller neighbour cells

Figure 1.9: SLIC algorithm on the dashed cell, neighbour cells in bold



(a) Shift a full cell C according to the velocity at its cell center and project its φ value on the receiver cells $P_i(C), i = 1, ..., 4$

(b) Case of a cell C with $\varphi^n(C) < 1$. The cell is treated with SLIC and the smaller cell $\tilde{C}$ is advected

Figure 1.10: Octree advection

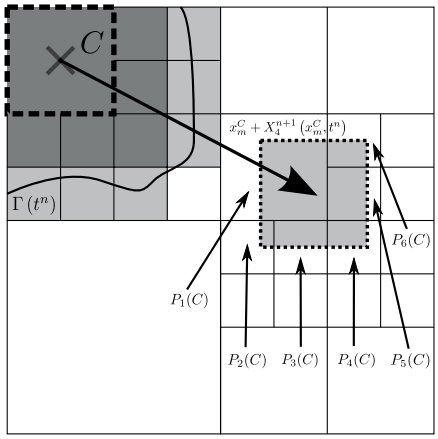


Figure 1.11: Advection of a large cell

If the adjacent cell is of lower level (i.e. the adjacent cell is larger in size), we can use the same procedure as for equal size cells. However, if an adjacent cell is of greater level (i.e. the adjacent cell is smaller in size), we have multiple adjacent cells in that particular direction and it is not clear which pattern to choose. In this case, we choose the pushing pattern as if there were a unique adjacent cell C_u such that $\varphi^n(C_u) = \varphi_u$, where φ_u is as follows.

- If all adjacent cells C_a satisfy $\varphi(C_a) > 1 - \epsilon_{SLIC}$, set $\varphi_u = 1$
- If all adjacent cells C_a satisfy $\varphi(C_a) < \epsilon_{SLIC}$, set $\varphi_u = 0$
- Otherwise set $\varphi_u = 0.5$

where $\epsilon_{SLIC} = 0.05$.

We now introduce the core of the advection scheme that solves (1.3) on the octree. The translation of the cells and projection on the octree are analogous to the scheme on the uniform structured grid illustrated in Figure 1.1.

Let us call $X_4^{n+1}(C, t^n)$ the translation of cell $C \in \mathscr{D}^n$ by vector $X_4^{n+1}(x_m^C, t^n)$ where x_m^C is the center of cell C. $X_4^{n+1}(C, t^n)$ shall be the resulting transported hexahedron. For each cell C, let us define $\tilde{C}$ as the fraction of C resulting from applying the SLIC algorithm to the cell C. If C is a full cell, we simply have $\tilde{C} = C$ as shown in Figure 1.10a. Instead of advecting cell C, we instead advect $\tilde{C}$ as shown in Figure 1.10b and denote by $X_4^{n+1}(\tilde{C}, t^n)$ the translation of $\tilde{C}$ with vector $X_4^{n+1}(x_m^C, t^n)$. In the case of small fluid volumes, a translation of $\tilde{C}$ by $X^{n+1}(x_m^{\tilde{C}}, t^n)$ could also be considered since x_m^C can fail to belong to $\tilde{C}$.

To project the hexahedron $X_4^{n+1}(\tilde{C}, t^n)$ onto the octree, we look for all cells in $\mathscr{D}^{n+1/2}$ intersecting it. Let us denote $P_i(\tilde{C}) \in \mathscr{D}^{n+1/2}$ the i-th cell intersecting $X_4^{n+1}(\tilde{C}, t^n)$ where i ranges from 1 to the number of cells in the intersection.

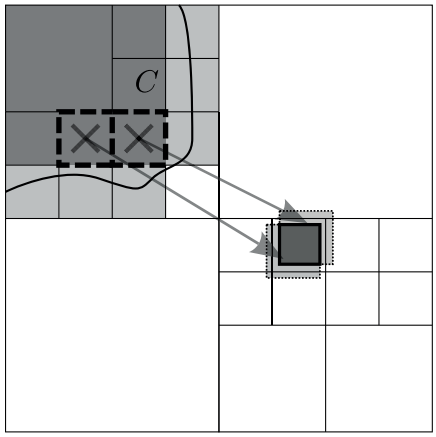


Figure 1.12: When the timestep is large, numerical compression can occur and a cell gets a φ value greater than 1.

For each non-empty cell C, a contribution of

$$\frac{volume(P_i(\tilde{C}) \cap X_4^{n+1}(\tilde{C}, t^n))}{volume(P_i(\tilde{C}))}$$

will be added to $\varphi^{n+1}(P_i(\tilde{C}))$. An illustration of this process is shown in Figure 1.10. Projecting large cells is done exactly the same way as illustrated in Figure 1.11.

Another equivalent formulation arises when we consider the receiver cell instead of the cell to be projected. For each cell $D \in \mathscr{D}^{n+1/2}$, $\varphi^{n+1}(D)$ is given by

$$\varphi^{n+1}(D) = \sum_{C \in \mathscr{D}^n} \frac{volume(D \cap X_4^{n+1}(\tilde{C}, t^n))}{volume(D)}.$$

If fluid is projected outside the computational domain, it is stored in a buffer and redistributed later in the cavity using the decompression algorithm. φ^{n+1} has now been defined on $\mathscr{D}^{n+1/2}$ and will be post-processed to ensure $0 \leq \varphi^{n+1} \leq 1$.

1.4.4 Decompression

When the timestep is large, a cell $D \in \mathscr{D}^{n+1/2}$ can have a VOF value $\varphi^{n+1}(D) > 1$ after the advection step as illustrated in Figure 1.12. This is meaningless since φ is a marker quantity designed to define the fluid domain with $\varphi = 0$ in the absence of fluid or $\varphi = 1$ in its presence. We will therefore set $\varphi^{n+1}(D) = 1$ for those cells and redistribute the excess fluid in the cavity in order to enforce mass conservation. Heuristic decompression algorithms such as in [36] are proposed to accomplish this task. These heuristic algorithms consist in choosing a priority function $\theta : \mathscr{D}^{n+1/2} \to \mathbb{R}$ and redistributing the total excess fluid to the cells in $\mathscr{D}^{n+1/2}$ in

decreasing order of their θ values. Algorithm 3 describes this process more formally. $V(C)$ indicates the volume of C where $C \in \mathscr{D}^{n+1/2}$. The decompression algorithm ends when the buffer is empty or when there are no more cells to decompress to. In the latter case, we keep the buffer for the next iteration so as to conserve mass.

Algorithm 3 Decompression algorithm with priority function θ

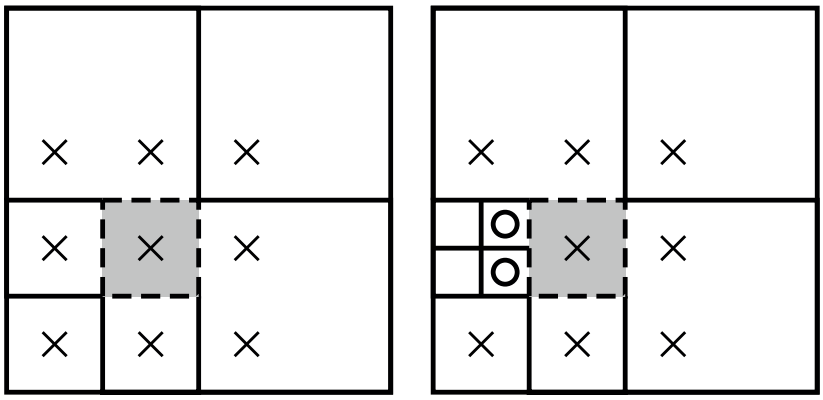
```
b ← 0                                          ▷ b is a buffer storing the excess fluid
for all cell C ∈ 𝒟^{n+1/2} do
    if φ^{n+1}(C) > 1 then
        b ← b + (φ^{n+1}(C) − 1) V(C)
        φ^{n+1}(C) ← 1
    end if
end for
A_h ← {C ∈ 𝒟^{n+1/2} | 0 < φ^{n+1}(C) < 1}
while b > 0 and |A_h| > 0 do
    choose C in A_h which maximizes θ(C)
    remove C from A_h
    s ← 1 − φ^{n+1}(C)                         ▷ Fraction of non-liquid space in cell
    φ^{n+1}(C) ← min(1, φ^{n+1}(C) + b / V(C))
    b ← max(0, b − s V(C))
end while
```

Classic decompression We call the classic decompression the one used in [36] which consists in choosing $\theta = \varphi^{n+1}$. This means that we redistribute the fluid first to cells with highest φ^{n+1} values.

Median filter smoothing decompression This algorithm is similar to the classic decompression but instead of choosing $\theta = \varphi^{n+1}$, we choose for θ a smoothing of φ^{n+1}. The motivation behind this is to redistribute excess fluid on the cells which have highest φ^{n+1} values and whose neighbours also have high φ^{n+1} values. Thus, we hope to first redistribute excess fluid in the bulk of the fluid and only then on the interface. We choose the median filter [72] as a smoothing operator for two main reasons. First, the filter only requires the φ values of the adjacent cells to the one we want to apply the smoothing to, which makes the overhead for neighbour search in the octree relatively low. Second, the filter has some desirable noise-suppressing properties (see e.g. [73]).

The median filter works by retrieving the median value of a compact stencil. Suppose that we have a two-dimensional structured grid G_s of cells $C_{i,j}$ and a field $\varphi : G_s \to \mathbb{R}$ that we want to filter. Let $\varphi_s : G_s \to \mathbb{R}$ be the filtered field. It is defined as the median of a multiset as follows

$$\varphi_s(C_{i,j}) = median\left\{\varphi(C_{i-s,j-t}),\ s,t \in \{-1,0,1\}\right\}.$$



(a) Median filter stencil with larger neighbours

(b) We average the φ value of the circled cells before computing the median

Figure 1.13: Stencil used for the median filter applied to the octree.

In three dimensions, the median filter thus requires getting the φ values of 26 neighbouring cells and of the cell to be filtered. To apply this filter on the octree, we can apply the filter as is but we need to treat the cases where the cell has larger neighbours or smaller neighbours.

Suppose now that we want to get the filtered value of φ^{n+1} at cell $C \in \mathscr{D}^{n+1/2}$ where C has neighbours of same size or larger, h_C is the edge length of C (we suppose it is a cube here) and x_m^C is the center-point of C. We use the 27 point compact stencil centered at x_m^C of distance between adjacent nodes h_C which we note S_C illustrated in Figure 1.13.a. The filtered value of φ^{n+1} is the median of the multiset $\{\varphi^{n+1}(D),\ D \in \mathscr{D}^{n+1/2} \text{ containing } p,\ p \in S_C\}$, allowing duplicate values. If a cell contains two nodes of the stencil, its φ^{n+1} value should appear twice in the expression of the multiset.

Now suppose that we want to get the filtered value of φ^{n+1} at cell $C \in \mathscr{D}^{n+1/2}$ where C also has neighbours of smaller size. For a point of the stencil $p \in S_C$, if the cell C has smaller neighbours in the direction $p - x_m^C$, the point p will be a common vertex of 8 cells and we do not get a unique value of φ. We choose instead to take the average value of $\varphi^{n+1}(C_p^i)$ where C_p^i are the cells sharing a face or an edge with cell C in direction $p - x_m^C$. We add this value to the multiset instead. The Figure 1.13.b illustrates this case; the cells sharing an edge or face are marked with a circle.

We choose to use the median filter smoothing decompression for our simulations. It results in less spurious holes than the classic decompression since the bulk of the fluid is prioritized during redistribution due to the smoothing. The redistribution does not follow the physics of the problem but is considered to be a post-processing which corrects spurious fluid compressions. Redistribution algorithms based on the physics of the problem could be considered but we have found the median filter decompression to yield satisfactory results in little computational time. It should be noted that the amount of numerical compression tends towards zero as cell

size decreases.

We compute the smoothing of the φ^{n+1} field and apply Algorithm 3 to enforce $0 \leq \varphi^{n+1}(C) \leq 1$ $\forall C \in \mathscr{D}^{n+1/2}$.

We present a comparison of Median filter smoothing decompression and Classic decompression methods on a free surface flow problem. Free surface flows are introduced in Section 1.6 but we anticipate in order to compare the decompression schemes on a real test case. A 2D-3D jet buckling test case documented in [42] is simulated on a structured grid for both decompression algorithms and results are shown in Figure 1.14. It can be seen that the median filter decompression method redistributes the fluid in the bulk instead of the boundary and as a consequence, hinders the appearance of spurious bubbles as can be seen in the classic decompression case. It can be noted that the median filter decompression results in a more stable jet, which buckles at a later time than for a classic decompression scheme. This particular simulation is however an extreme case where the slightest perturbations of the jet equilibrium can drastically change the timing and direction of the jet buckling. In simulations that we have run and that will be analyzed later, the choice between classic decompression and median filter decompression algorithms does not drastically affect the evolution of the liquid region. It is however of particular interest for the coarsening of the octree mesh to prevent the appearance of these spurious bubbles in the first place. Indeed, we will see in Section 1.4.5 how we require cells to be filled to be able to coarsen the octree cells in the bulk of the fluid. A decompression algorithm hindering the appearance of the spurious bubbles will therefore allow for more cells to be coarsened and therefore leads to a faster and less memory-consuming algorithm. These spurious bubbles are however expected to disappear when timestep and cell size tend to zero.

1.4.5 Coarsening

After the advection and decompression steps, we coarsen non-interfacial cells for memory and speed gains. From the point of view of the octree graph, coarsening means removing 8 children nodes in the octree graph belonging to the same parent. From the point of view of the octree mesh, this means joining 8 cells belonging to the same parent cell. The values of φ^{n+1} of the parent cell will be given by the average of the corresponding values of its 8 children cells.

The goal is to coarsen the cells as much as possible while satisfying the following requirements.

- Liquid cells should not be coarsened further than the level l_{liquid}
- Interfacial cells should not be coarsened
- Eight cells C_i can only be coarsened if they all satisfy either of the conditions $\varphi^{n+1}(C_i) \geq 1 - \epsilon_{liquid}$, $i = 0, ..., 7$ or $\varphi^{n+1}(C_i) = 0$, $i = 0, ..., 7$.

The first requirement is present to preserve an accurate representation of the velocity field

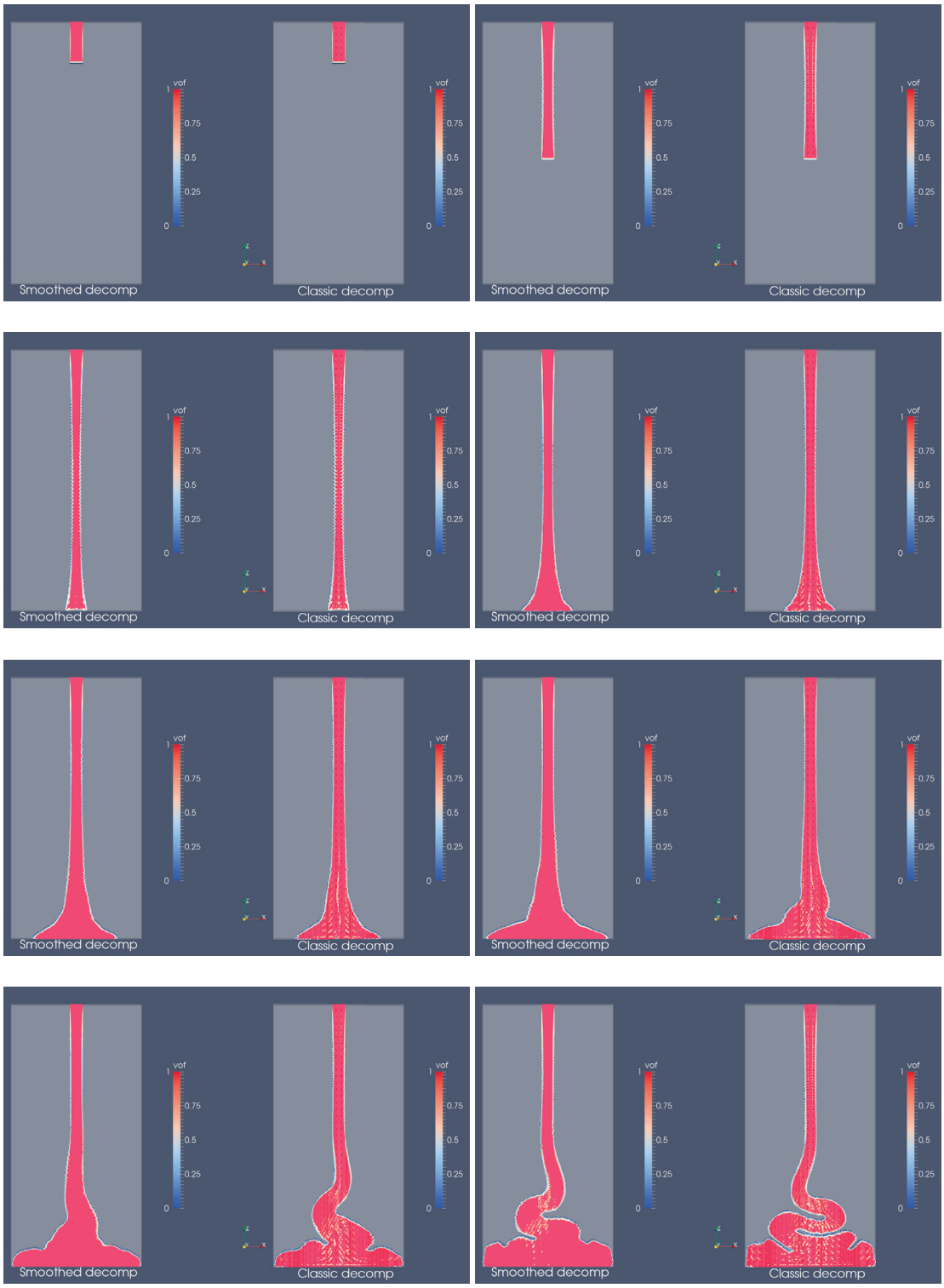
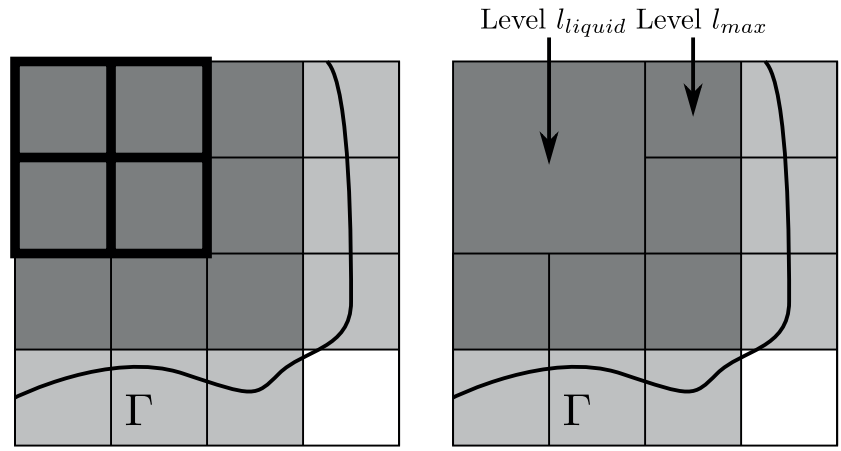


Figure 1.14: Decompression algorithm comparison. Median filter smoothing decompression on the left and classic decompression on the right. Median filter smoothing redistributes fluid better in the bulk of the fluid rather than on the boundary.



(a) Cells sharing the same parent satisfy coarsening requirements

(b) The cells are replaced by their parent cell

Figure 1.15: Coarsening process.

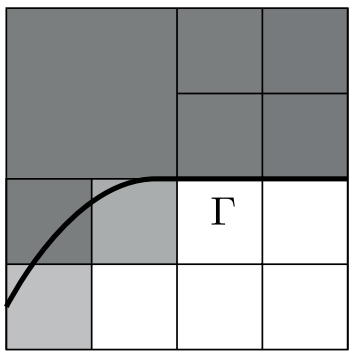


Figure 1.16: Coarsening is not done if one of the 8 cells has a neighbour in a different state (liquid/non-liquid).

when evaluating the velocity field at cell centers. The second requirement is needed to keep interfacial cells at level l_{max} so as to preserve an accurate representation of the interface. The last requirement is needed because when the timestep is large, it can happen that a cell C in the bulk of the fluid with level $l < l_{max}$ takes a value of $\varphi^{n+1}(C) < 1$. The latter should in principle not happen since a coarse cell C in the fluid bulk is supposed to satisfy $\varphi^{n+1}(C) = 1$. For these reasons, we also allow a relaxed coarsening if the cell is not fully liquid but $\varphi^{n+1}(C)$ is close enough to 1. In that case, we set $\varphi^{n+1}(C) = 1$ for the coarsened cell C and the buffer containing the fluid to redistribute in the decompression algorithm at the next timestep is decreased so as to enforce mass conservation. The buffer will then temporarily be negative until compression occurs at the next step. It has never been observed that the buffer would stay negative throughout simulations. This additional post-processing is justified in a similar way to the decompression algorithm. In the decompression algorithm, we correct spurious compressions and in this step we correct spurious diffusions in the bulk of the fluid. With the relaxation, we obtain a more efficient coarsening and a sharper definition of the fluid bulk. Numerical results in Section 2 confirm that this does not prevent accurate tracking of the interface and in fact can even improve the results in difficult test cases where spurious bubbles appear in the non-adaptive algorithm. We have found $\epsilon_{liquid} = 0.1$ to yield satisfying results. The coarsening process is illustrated in Figure 1.15. We apply the coarsening procedure on the octree $\mathscr{D}^{n+1/2}$ and obtain a coarsened octree $\mathscr{D}^{n+1}$.

1.5 Octree implementation

An efficient octree implementation has to be adapted to the numerical scheme described above, therefore the following features are desirable. Low memory overhead, efficient iteration over the octree cells and retrieval of parent, child, neighbour cells and of leaf cells containing a specified point are all features that need to be satisfied by our octree implementation.

Early implementations of the quadtree, which is the 2D equivalent of the octree, such as [74] used to store a reference to the parent and children cells for each cell of the quadtree. Coordinates and level information could be stored in cells or computed from the path to the root node depending on the desired tradeoff between memory usage and CPU time. The minimal storage version of the octree implementation has a structural memory overhead of 9 memory words per cell (one reference to the parent and 8 to the children). Retrieval of neighbour cells required ascending the octree and computing a path to descend.

An improvement to this implementation is called the Fully Threaded Tree (FTT) and was proposed in [69]. It introduced the notion of oct which aggregates positional information of 8 cells belonging to the same parent. The oct stores its level, coordinates and references to parent cell and parent cells of neighbouring octs, along with the 8 cells belonging to that oct. Each cell contains the physical quantities associated with it along with a pointer to its associated oct. This implementation reduces memory overhead and makes neighbour retrieval more efficient and simpler. Indeed, the structural memory overhead of 19 words per

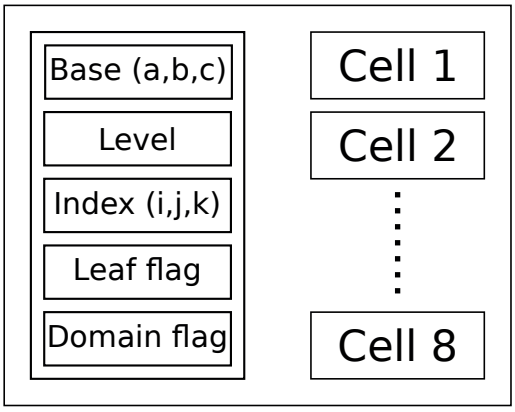


Figure 1.17: Memory map of an oct

oct which corresponds to $2\frac{3}{8}$ words per cell is lower than the previous 9 words per cell.

Recently, a pointerless implementation of the octree was proposed in [75]. An integer indexing system identifies uniquely each oct and retrieving parent, children and neighbour cells amounts to simple arithmetic operations. This eliminates the need for references to other cells and yields a memory overhead of $\frac{5}{8}$ words per cell. The tree-like structure is replaced with a simple hash map whose keys are integers identifying each oct which makes the structure easier to handle and slightly more efficient.

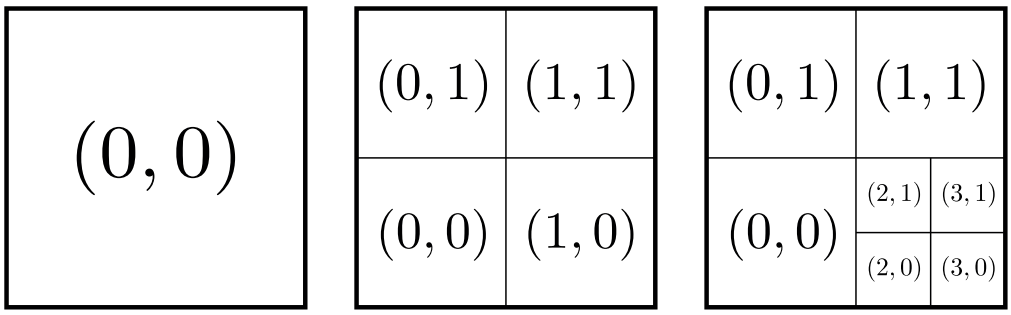
We take advantage of the strategy used in [75] and extend it to allow cells of arbitrary aspect ratio. This is done by replacing the single base hexahedron by an array of base hexahedra which can be chosen so as to get the desired cell aspect ratio. We also allow for more complex domains with the use of a flag indicating whether a cell is inside or outside the domain.

Let us drop the assumption that the computational domain is a hexahedron and let the axis-aligned bounding box of the computational domain be of dimensions $(L_x, L_y, L_z) \in \mathbb{R}^3$.

As illustrated in Figure 1.17, an oct stores in memory a *base* integer triplet (a, b, c), the refinement *level*, an *index* integer triplet (i, j, k), a *leaf* flag indicating which cells are leaves and a *domain* flag indicating which cells are in the computational domain. It also stores the physical quantities associated with its 8 cells.

Recall that in order to avoid having the bounding box dictate the anisotropy of cells, we subdivide it into an array of octs with respectively B_x, B_y, B_z octs in directions x, y, z as illustrated in Figure 1.4. Such octs are set to level 0 and will be called *base octs*. Any oct in the octree belongs to a unique base oct and the *base* integer triplet (a, b, c) represents the coordinates of that base oct in the array.

The *index* (i, j, k) of an oct is recursively defined as follows. Let the index of a base oct be $(0, 0, 0)$. Suppose that the index of an oct is (i, j, k), then its 8 children octs have indices defined



(a) Index of base oct

(b) Indices of children given by $(2\cdot 0+\delta_i, 2\cdot 0+\delta_j)$, $\delta_i,\delta_j \in \{0,1\}$.

(c) Indices of further children given by $(2\cdot 1+\delta_i, 2\cdot 1+\delta_j)$, $\delta_i,\delta_j \in \{0,1\}$.

Figure 1.18: Oct indices.

by

$$(2i+\delta_i,\ 2j+\delta_j,\ 2k+\delta_k), \quad \delta_i,\delta_j,\delta_k \in \{0,1\}.$$

A 2D example illustration is provided in Figure 1.18.

The simple recursive definition of the index makes it easy to retrieve the parent oct's index (i_p, j_p, k_p) given a child oct's index (i, j, k). It is given by

$$(i_p, j_p, k_p) = \left(\left\lfloor \frac{i}{2} \right\rfloor, \left\lfloor \frac{j}{2} \right\rfloor, \left\lfloor \frac{k}{2} \right\rfloor\right).$$

Assume for simplicity that there is a single base oct in the octree, i.e. $B_x = B_y = B_z = 1$. Given an oct of level l, another way of computing the index is simply to count its coordinates in an array of octs of level l. . This means that if an oct with index (i, j, k) has a neighbour oct of same level of index (i_n, j_n, k_n), we have

$$\begin{aligned}(i_n, j_n, k_n) = (i+\gamma_i,\ j+\gamma_j,\ k+\gamma_k),\ |\gamma_i|+|\gamma_j|+|\gamma_k| = 1,\\ \gamma_i,\gamma_j,\gamma_k \in \{-1,0,1\}.\end{aligned}$$

As a consequence, an oct is uniquely defined by its level, base and index and its coordinates can be computed. Let (x_0, y_0, z_0) be the vertex of the bounding box which has minimal (x, y, z) coordinates and recall that the bounding box has dimensions (L_x, L_y, L_z). The dimensions of an oct of level l, base (a, b, c) and index (i, j, k) are $\left(\frac{L_x}{B_x\, 2^l}, \frac{L_y}{B_y\, 2^l}, \frac{L_z}{B_z\, 2^l}\right)$ and its vertex which has minimal coordinates is given by

$$(x_v, y_v, z_v) = \left(x_0 + \frac{L_x}{B_x}(a+\frac{i}{2^l}),\ y_0 + \frac{L_y}{B_y}(b+\frac{j}{2^l}),\ z_0 + \frac{L_z}{B_z}(c+\frac{k}{2^l})\right).$$

The oct structures are indexed in a hash table using an *ID* which uniquely encodes level, base and index of an oct. A hash table (or hash map) is a data structure which allows mapping of *keys* (here the *ID*) to *values* (here the *oct*) with the advantage of an average number of $O(1)$ operations to retrieve, insert and delete elements in the structure. Here we modify the definition of *ID* given in [75] to allow arrays of base octs which permit arbitrary cell aspect ratios. Given an oct of level l, base (a, b, c) and index (i, j, k) we define the *ID* as

$$ID = a + b\,B_x + c\,B_xB_y + B_xB_yB_z\left(\sum_{n=0}^{l-1} 2^{3n} + i + j\,2^l + k\,2^{2l}\right).$$

Note that if the number of base octs in each direction are powers of two, the information encoded into the *ID* can be restored using fast binary operations. With this strategy, it is easy retrieve parent octs and children octs from the octree. Indeed, to retrieve the parent oct of an oct with index (i, j, k), we simply compute the parent index $\left(\left\lfloor\frac{i}{2}\right\rfloor, \left\lfloor\frac{j}{2}\right\rfloor, \left\lfloor\frac{k}{2}\right\rfloor\right)$ which allows us to compute the parent ID and query the hash map with it.

To retrieve a leaf oct containing a point (x, y, z) we first compute its base

$$(a, b, c) = \left(\left\lfloor\frac{x - x_0}{L_x}B_x\right\rfloor, \left\lfloor\frac{y - y_0}{L_y}B_y\right\rfloor, \left\lfloor\frac{z - z_0}{L_z}B_z\right\rfloor\right).$$

Assuming that a leaf oct of level l exists in the octree, we can compute its index

$$(i_l, j_l, k_l) = \left(\left\lfloor 2^l\left(\frac{B_x}{L_x}(x - x_0) - a\right)\right\rfloor, \left\lfloor 2^l\left(\frac{B_y}{L_y}(y - y_0) - b\right)\right\rfloor, \left\lfloor 2^l\left(\frac{B_z}{L_z}(z - z_0) - c\right)\right\rfloor\right).$$

We then compute the ID and query the hash map. Usually we do not know the level l of the leaf oct containing the point in advance, so a simple strategy is to start from level l_{max} and query the hash map with octs of decreasing level until the query is successful. In the numerical experiments we have conducted, l_{max} rarely exceeded 6. There is therefore probably little to gain in terms of algorithm speed in looking for a better heuristic to predict which level the query will be successful at since the number of queries is bounded by $l_{max} + 1$.

Since hash maps are also equipped with algorithms for efficient traversal of all elements, we have satisfied all the desired features of our octree implementation.

The handling of complex domains is supported via the *domain* flag and will be explained in detail in section 1.6.5.

1.6 Octree-based scheme for free surface flows governed by the Navier Stokes equations

1.6.1 Splitting scheme for the Navier Stokes equations

We introduce a splitting scheme to simulate free surface flows governed by the Navier Stokes equations. Recall that $T > 0$ is the final simulation time. Let Q_T be the space-time domain containing the fluid at all times $Q_T = \{(x,t) \in \Lambda \times (0,T);\ \varphi(x,t) = 1\}$. Let $v : Q_T \to \mathbb{R}^3$ be the velocity field and $p : Q_T \to \mathbb{R}$ the pressure field. Note that the velocity field is now defined only in the liquid domain and not in the whole cavity Λ as in Section 1.1. We will be solving the time-dependent incompressible Navier-Stokes equations

$$\rho \frac{\partial v}{\partial t} + \rho (v \cdot \nabla) v - 2\mu \nabla \cdot \epsilon(v) + \nabla p = \rho g \qquad \text{in } Q_T, \tag{1.15}$$

$$\nabla \cdot v = 0 \qquad \text{in } Q_T, \tag{1.16}$$

where ρ is the density, μ the dynamic viscosity of the fluid, g the gravitational field and $\epsilon(v) = (\nabla v + \nabla v^T)/2$ the rate-of-strain tensor. The Navier Stokes equations are coupled with the advection equation (1.1) which translates the fact that the interface moves with the liquid. In order to solve the full Navier-Stokes equations, we will use an implicit splitting algorithm of order one. The splitting scheme described in [40] allows decoupling the advection part from other phenomena, such as diffusion, incompressibility and effect of external forces in our case. This time splitting scheme has been successfully used with a structured cartesian mesh in [41, 36, 42, 37, 38, 43, 44] and we describe how it can be used with the octree.

Let $\Delta t = \frac{T}{N}$ be the timestep where $N > 0$ is an integer. The time discretization points are then $t_n = n\Delta t$, $n = 0, 1, ..., N$ and $\Lambda \subset \mathbb{R}^3$ is the bounded computational domain. Let us assume that at timestep t_n, an approximation $\varphi^n : \Lambda \to \mathbb{R}$ of the liquid characteristic function is known which defines the domain Ω^n, an approximation of the liquid domain at time t_n.

Due to the splitting, the velocity satisfies a transport equation. We assume that an approximation $v^n : \Omega^n \to \mathbb{R}^3$ of the velocity field at time t_n is available.

In the advection step, we first transport φ^n along with the velocity v^n to obtain the new liquid characteristic function φ^{n+1} which defines the new liquid region Ω^{n+1} and a first estimate of the velocity field $v^{n+1/2}$. The second step of the splitting algorithm then yields a Stokes problem to solve which gives the final estimate v^{n+1} of the velocity at timestep t_{n+1}.

The advection step consists in solving

$$\frac{\partial \varphi}{\partial t} + v \cdot \nabla \varphi = 0, \tag{1.17}$$

$$\frac{\partial v}{\partial t} + v \cdot \nabla v = 0, \tag{1.18}$$

between t_n and t_{n+1} where the initial conditions are given by

$$\varphi(t_n) = \varphi^n,$$
$$v(t_n) = v^n.$$

From equation (1.18) we deduce that velocity is constant along characteristics and therefore the characteristics are straight lines. Indeed, similarly to (1.2), the characteristics $X : [t_n, t_{n+1}] \to \mathbb{R}^3$ are here defined as

$$\dot{X}(t) = v(X(t), t) \quad \forall t \in [t_n, t_{n+1}]. \tag{1.19}$$

Equation (1.18) then implies that

$$\frac{d}{dt} v(X(t), t) = 0 \quad \forall t \in [t_n, t_{n+1}]. \tag{1.20}$$

Since the characteristics are straight lines, they are computed exactly with a first order method and it is therefore sufficient to use an explicit Euler method which gives

$$\varphi^{n+1}(X_1^{n+1}(x, t^n)) = \varphi^n(x). \tag{1.21}$$
$$v^{n+1/2}(X_1^{n+1}(x, t^n)) = v^n(x). \tag{1.22}$$

for $x \in \Omega^n$. φ^{n+1} defines the new liquid region $\Omega^{n+1} = \{x \in \Lambda : \varphi^{n+1}(x) = 1\}$ on which an estimate of the new velocity field $v^{n+1/2}$ after the first step of the splitting is computed.

Given $v^{n+1/2}$ and Ω^{n+1}, the second step of the splitting algorithm consists in computing the new velocity v^{n+1}. To do this, we solve the following Stokes problem

$$\rho \frac{v^{n+1} - v^{n+1/2}}{\Delta t} - 2\mu \nabla \cdot \epsilon(v^{n+1}) + \nabla p^{n+1} = \rho g \qquad \text{in } \Omega^{n+1}, \tag{1.23}$$
$$\nabla \cdot v^{n+1} = 0 \qquad \text{in } \Omega^{n+1}. \tag{1.24}$$

In Section 1.6.3, we detail the space discretization and algorithms used to perform both splitting steps in practice.

1.6.2 Initial conditions and boundary conditions

We briefly recall the initial and boundary conditions given for instance in [36]. Given an initial liquid characteristic function $\varphi(\cdot, 0)$, the initial liquid domain is then $\Omega(0) = \{x \in \Lambda : \varphi(x, 0) = 1\}$. The initial velocity then has to be given in $\Omega(0)$.

The boundary conditions for the Stokes problem (1.23)-(1.24) are then as follows. Neglecting forces due to pressure from the gas outside the liquid region and any capillary forces, we assume that the interface is stress-free. Therefore, the following boundary condition holds at

the boundary of the initial liquid region $\Omega(0)$ that is not in contact with the domain boundary

$$2\mu\epsilon(v)\cdot n - pn = 0$$

where n is the outward unit normal to the free surface.

We describe two options for the boundary condition in the region where the liquid is in contact with the domain boundary. The first option is a Dirichlet condition which imposes all three components of the velocity. This is the so-called inflow condition or noslip condition in the case of homogenous Dirichlet boundary conditions. The second option is the Signorini boundary condition, which imposes zero normal velocity and tangential stress on the boundary if the liquid pushes against the boundary wall. This translates as imposing

$$v.n = 0 \quad \text{and} \quad \big(2\mu\epsilon(v)\cdot n - pn\big)\cdot t_i = 0,\ i = 1,2 \quad \text{if} \quad \big(2\mu\epsilon(v)\cdot n - pn\big)\cdot n < 0$$

where $t_i,\ i = 1,2$ are two orthogonal unit vectors of the plane tangent to the boundary wall. On the other hand, if the fluid does not push against the boundary wall, a zero stress condition is imposed as follows

$$2\mu\epsilon(v)\cdot n - pn = 0 \quad \text{if} \quad \big(2\mu\epsilon(v)\cdot n - pn\big)\cdot n \geq 0.$$

1.6.3 Space discretization of the splitting scheme

The splitting scheme allows the use of different grids for the advection problem and Stokes problem. The advection step is done on the octree and the Stokes problem is solved on a tetrahedral grid. As stated in the first part, the choice of the octree was motivated by the capture of fine detail at the interface while keeping coarser cells in the bulk of the fluid for efficiency. The Stokes problem is solved on a tetrahedral grid for several reasons.

First of all, finite elements on tetrahedral grids have been the object of extensive research and are therefore relatively well known, see e.g. the references in [46]. Tetrahedral meshes allow accurate capture of even complex domain boundaries and boundary normals. The latter is useful for imposition of the slip boundary condition for instance. This would be a challenge

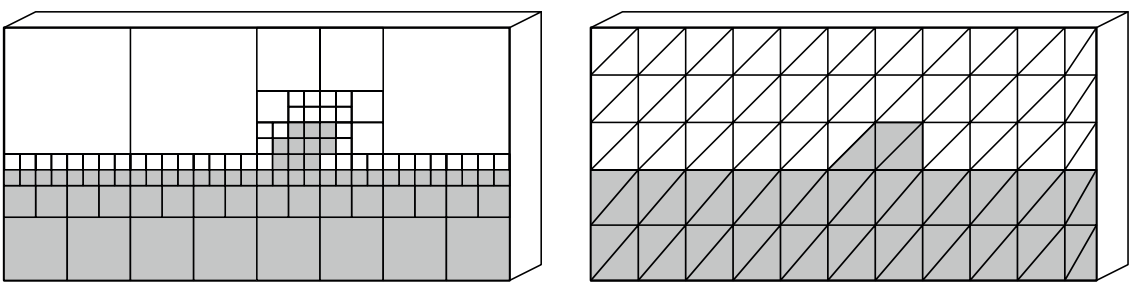


Figure 1.19: Octree grid for advection and tetrahedral mesh for Stokes problem.

if we wanted to solve the Stokes problem on the octree as well. Solving the Stokes problem on the octree would create other problems, such as dealing with the hanging nodes in the octree. Also, the large number of fine cells at the boundary might lead to a prohibitively high cost. We assume that the continuity of the velocity allows us to solve the Stokes problem on a tetrahedral mesh coarser than the fine octree cells as illustrated in Figure 1.19. Let us call the tetrahedral mesh $\mathscr{T}_H$ where H is the maximal diameter of all tetrahedra in $\mathscr{T}_H$. We will choose H to be approximately of the same size as h_{liquid}.

Advection of the liquid characteristic function φ has been described in previous Sections in detail. Advection of the velocity happens in an analogous way. Since we now use an explicit Euler scheme to follow the characteristics, for each cell $D \in \mathscr{D}^{n+1/2}$, $\varphi^{n+1}(D)$ is given by

$$\varphi^{n+1}(D) = \sum_{C \in \mathscr{D}^n} \frac{volume(D \cap X_1^{n+1}(\tilde{C}, t^n))}{volume(D)}$$

where $\tilde{C}$ is the cell C treated with the SLIC algorithm.

Analogously for the velocity projection, each contributing cell $C \in \mathscr{D}^n$ yields a weighted contribution of $v^{n+1/2}(C)$ to the receiver cell $D \in \mathscr{D}^{n+1/2}$. That weight is precisely the normalized contribution of cell C to $\varphi^{n+1}(D)$. For each cell $D \in \mathscr{D}^{n+1/2}$ such that $\varphi^{n+1}(D) > 0$, $v^{n+1/2}(D)$ is then given by

$$v^{n+1/2}(D) = \sum_{C \in \mathscr{D}^n} v^n(C) \frac{\left(\dfrac{volume(D \cap X_1^{n+1}(\tilde{C}, t^n))}{volume(D)} \right)}{\varphi^{n+1}(D)}.$$

We will explain in Section 1.6.4 how we interpolate φ^{n+1} and $v^{n+1/2}$ onto the tetrahedral mesh $\mathscr{T}_H$ to define which tetrahedra are considered liquid and also to define the velocity on those tetrahedra.

The Stokes problem is then solved on the union of all liquid elements using $\mathbb{P}_1 - \mathbb{P}_1$ finite elements, an implicit Euler time discretization and boundary conditions given in Section 1.6.2. Several possible stabilizations are discussed in Chapter 3. As will be seen, an efficient choice which does not require a stabilization parameter is the bubble reconstruction method described in Section 3.1.5.

Closing the loop, Section 1.6.4 describes how velocity is interpolated back onto the octree from the tetrahedral mesh for the next iteration.

1.6.4 Interpolations between meshes

Dynamic mapping between meshes In order to compute interpolations between octree cells and tetrahedra, we need an efficient way of mapping each cell to its neighbouring

tetrahedra and vice versa. We say that a cell $C \in \mathscr{D}_h^{n+1}$ is the **neighbour** of a tetrahedron $K \in \mathscr{T}_H$ and vice-versa if their axis-aligned bounding boxes have a non-empty intersection.

At initialization, for each octree cell we compute and store a list of its tetrahedral neighbours and for each tetrahedron we store the list of its neighbour cells. When a cell C of the octree is split, each of its eight children stores a copy of the neighbour list of C and we simply remove the tetrahedra from the eight lists that are not neighbours with each respective child. The list of neighbours of C is then deleted. The neighbour lists of the tetrahedra involved in the update are also updated accordingly. When eight cells are coarsened, we set the neighbour list of the parent cell to be the union of the eight cells' neighbour lists and update the neighbour lists of the involved tetrahedra accordingly.

Interpolation from the octree to the tetrahedral mesh We wish to interpolate the fields φ^{n+1} and $v^{n+1/2}$ from the octree $\mathscr{D}_h^{n+1}$ to the tetrahedral mesh $\mathscr{T}_H$. φ^{n+1} is interpolated in order to determine which tetrahedra are considered liquid, that is on which tetrahedra we will solve the Stokes problem. We seek to determine an interpolated value φ_K^{n+1} for each tetrahedron K in $\mathscr{T}_H$. If $\varphi_K^{n+1} \geq 0.5$, the tetrahedron is considered liquid. Suppose that B_K is the axis-aligned bounding box of K in $\mathscr{T}_H$. We define φ_K^{n+1} as a weighted average of values $\varphi^{n+1}(C)$ where C are neighbours of K. Weights are given by the volume of the intersection between cell C and the axis-aligned bounding box of K. We have

$$\varphi_K^{n+1} = \frac{\sum\limits_{C \in \mathscr{D}_h^{n+1}} \varphi^{n+1}(C)\ volume(B_K \cap C)}{\sum\limits_{C \in \mathscr{D}_h^{n+1}} volume(B_K \cap C)}.$$

Recall that we are using $\mathbb{P}_1 - \mathbb{P}_1$ finite elements so we also need to define $v^{n+1/2}$ on the nodes of the tetrahedral mesh. In order to do that, we will use a variation of the inverse distance weighting interpolation, see e.g. [76]. Let $v_P^{n+1/2}$ be an approximation of $v^{n+1/2}$ at node P of mesh $\mathscr{T}_H$ given by the interpolation. Let B_P be the axis-aligned bounding box of the union of the tetrahedra K in $\mathscr{T}_H$ such that $P \in K$. To get a local interpolation and to handle different sizes of octree cells, we weight the contribution of each cell C not only by the inverse distance but also by the volume of the intersection of B_P and C. If there exists a cell C whose center c coincides with P, we set $v_P^{n+1/2} = v^{n+1/2}(C)$. Otherwise, we have the following expression

$$v_P^{n+1/2} = \frac{\sum\limits_{C \in \mathscr{D}_h^{n+1}} v^{n+1/2}(C)\ \dfrac{volume(B_P \cap C)}{\|c - P\|_2^2}}{\sum\limits_{C \in \mathscr{D}_h^{n+1}} \dfrac{volume(B_P \cap C)}{\|c - P\|_2^2}}$$

where $\|\|_2$ is the Euclidean norm and c is the cell center of C. Figure 1.20 shows a vertex P, the patch of tetrahedra containing it, the bounding box B_P and all the cells involved in the interpolation.

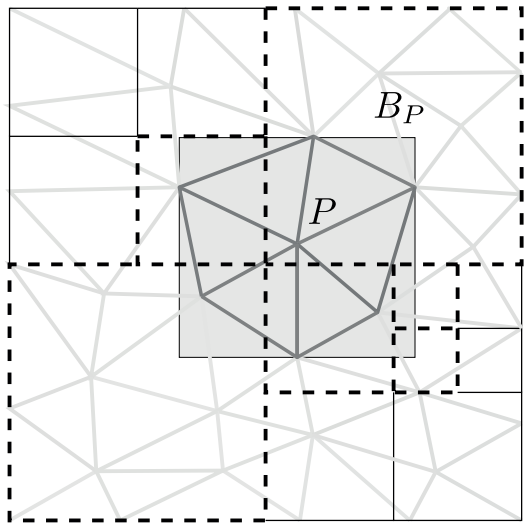


Figure 1.20: All the tetrahedra (here triangles) in $\mathcal{T}_H$ containing the vertex P are shown. The bounding box B_P of the union of those tetrahedra is shown in light grey. Cells in dashed lines are involved in the interpolation.

The interpolation of φ^{n+1} defines the liquid elements and therefore the subset of $\mathcal{T}_H$ on which the Stokes problem is solved. Note that it is possible for a cell to end up with only non-liquid tetrahedral neighbours, which means that all its neighbours are excluded from the Stokes' problem solving. In this case, move the fluid contained in the cell to the decompression buffer in order to redistribute it at the next decompression step. Given $v^{n+1/2}$ on the mesh $\mathcal{T}_H$, we solve the Stokes problem to obtain v^{n+1} and interpolate it back onto the octree as follows.

Interpolation from the tetrahedral mesh to the octree We have computed the piecewise linear field v^{n+1} on the mesh $\mathcal{T}_H$ by solving the Stokes problem. v^{n+1} is interpolated onto a cell $C \in \mathcal{D}_h^{n+1}$ of the octree by inverse distance weighting of the values $v^{n+1}(P)$ where P is a node of the neighbouring tetrahedron K of the cell C. Again, if the cell center c of C coincides with a node P of a tetrahedron $K \in \mathcal{T}_H$, we set $v^{n+1}(C) = v^{n+1}(P)$. Otherwise, set

$$v^{n+1}(C) = \frac{\displaystyle\sum_{K\in\mathcal{T}_H}\sum_{P\in K}\frac{v^{n+1}(P)}{\|c-P\|_2^2}}{\displaystyle\sum_{K\in\mathcal{T}_H}\sum_{P\in K}\frac{1}{\|c-P\|_2^2}}.$$

The interpolation is illustrated in Figure 1.21 With this interpolation, v^{n+1} is defined on the octree $\mathcal{D}_h^{n+1}$ and we proceed to the next iteration.

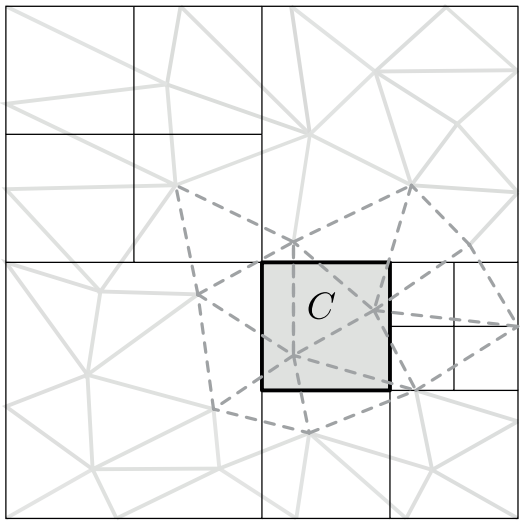


Figure 1.21: Tetrahedra (here triangles) in $\mathcal{T}_H$ shown in dashed lines are involved in the interpolation on the cell C.

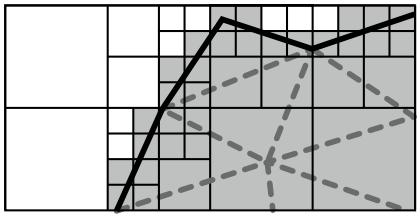


Figure 1.22: Cells with domain flag 1 are in gray and white cells have domain flag 0.

1.6.5 Capturing of complex domains

In Section 1.5, we mentioned that octs store a domain flag to indicate which cells belonging to that oct are inside the domain. More precisely, the domain flag is 1 for a cell which should be advected and advected into and it is 0 otherwise.

Given the tetrahedral mesh $\mathcal{T}_H$, the octree is initially refined up to level l_{max} for all cells intersecting the triangles in the boundary of $\mathcal{T}_H$, although for axis-aligned triangles we can choose whether to do so or not. This initially refined octree is the octree with minimal refinement allowed; we do not allow coarsening further than it to preserve accurate capture of the boundary throughout the simulation. For each non-leaf cell that has been split in this initial phase in order to capture the boundary, the domain flag is set to 0. Thus, when coarsening eight cells during the simulation, if the domain flag of the parent is 0, we do not allow the coarsening. In the initially refined octree, we set the domain flag of cells of level l_{max} that intersect the boundary or are fully within the domain to 1. Cells of level l_{max} that are outside of the domain have their domain flag set to 0. Up to now, all the cells intersecting

the boundary of $\mathscr{T}_H$ have had their domain flags set. All remaining cells that intersect any tetrahedron in the mesh have their domain flag set to 1 and then all the remaining cells are outside of the domain and have their domain flag set to 0. An illustration is provided in Figure 1.22. During the advection phase, the cells to be projected are given by iterating over all liquid cells $\mathscr{D}_h^n$ that have a domain flag of 1. If a receiver cell in $\mathscr{D}_h^{n+1/2}$ has a domain flag of 0, the fluid contribution is instead stored in the decompression buffer.

2 Numerical results

The octree convection scheme was implemented in *C++* using standard *STL* containers and was benchmarked on a system equipped with a 3.30GHz Intel Xeon E5-2643 processor and 32GB RAM. The scheme was benchmarked on a single core but could be parallelized by decomposing the domain and implementing a local version of the decompression algorithm on each subdomain. Parallelizing would require special care in decomposing the domain such that the load is balanced as equally as possible over all processors. A Hilbert space-filling curve such as proposed in [75] could be adapted to our octree implementation to do so.

The reported memory usage is the total physical memory allocation for our program performed by the operating system and therefore includes an overhead which becomes negligible as memory usage increases. All of the presented test cases have the property that $\varphi(\cdot,0) = \varphi(\cdot,T)$ where φ is the liquid characteristic function introduced in Section 1.1 and $T > 0$ final simulation time. The L^1 error is then the L^1 norm of the difference between fields φ at final time and at time $t = 0$, thus the initial space discretization error is excluded from it.

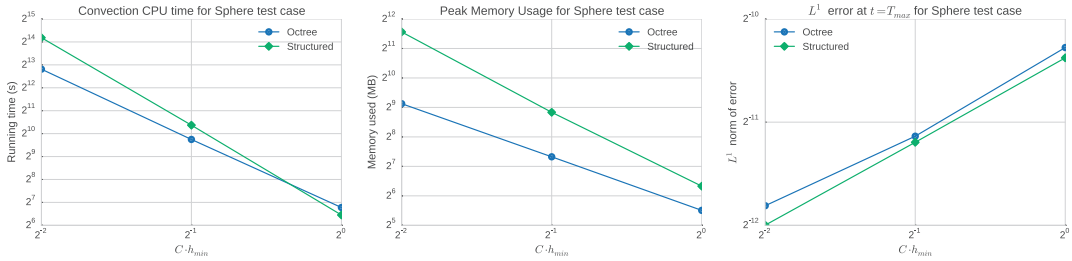
Comparisons are performed against the structured non-adaptive analogue of our convection scheme which is described in [36]. The forward Euler scheme for approximating characteristics in the structured grid solver is however changed to an order 4 Runge Kutta method.

The cell size of the structured non-adaptive grid is defined by h_{min} where h_{min} corresponds to the smallest cell size of the octree.

2.1 3D advection results with the octree-based scheme

2.1.1 Translation of a sphere

A sphere of radius 0.15 initially centered at $(0.2, 0.2, 0.2)$ is translated uniformly with velocity field $(1,1,1)$ from $t = 0$ to $t = 0.6$ and from $t = 0.6$ to $t = 1.2$, the velocity field is reversed such that the final position of the sphere at $T = 1.2$ corresponds to its initial position.



(a) Loglog plot of CPU time

(b) Loglog plot of Memory usage

(c) Loglog plot of L_1 Error

$C \cdot h_{min}$	Octree	Structured
1.0	1.097×10^2	8.743×10^1
0.5	8.588×10^2	1.323×10^3
0.25	7.196×10^3	1.865×10^4

(d) CPU time in seconds

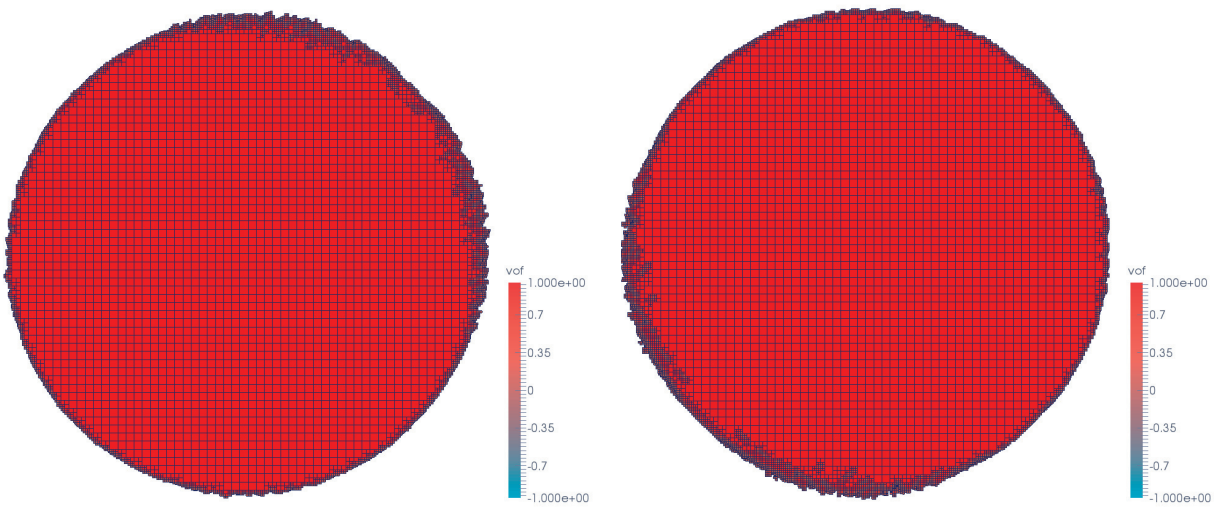
$C \cdot h_{min}$	Octree	Structured
1.0	45.49	80.39
0.5	160.30	456.50
0.25	559.10	3021.00

(e) Memory usage in MB

$C \cdot h_{min}$	Octree	Structured
1.0	8.062×10^{-4}	7.524×10^{-4}
0.5	4.443×10^{-4}	4.265×10^{-4}
0.25	2.785×10^{-4}	2.444×10^{-4}

(f) L_1 Error

Figure 2.1: CPU time, memory usage and error benchmark of the octree scheme for the sphere translation test case. h_{min} is the smallest cell size and we have set $C = 192$ as the reference cell size and $CFL = 6.6$.



(a) t=0.6

(b) t=1.2

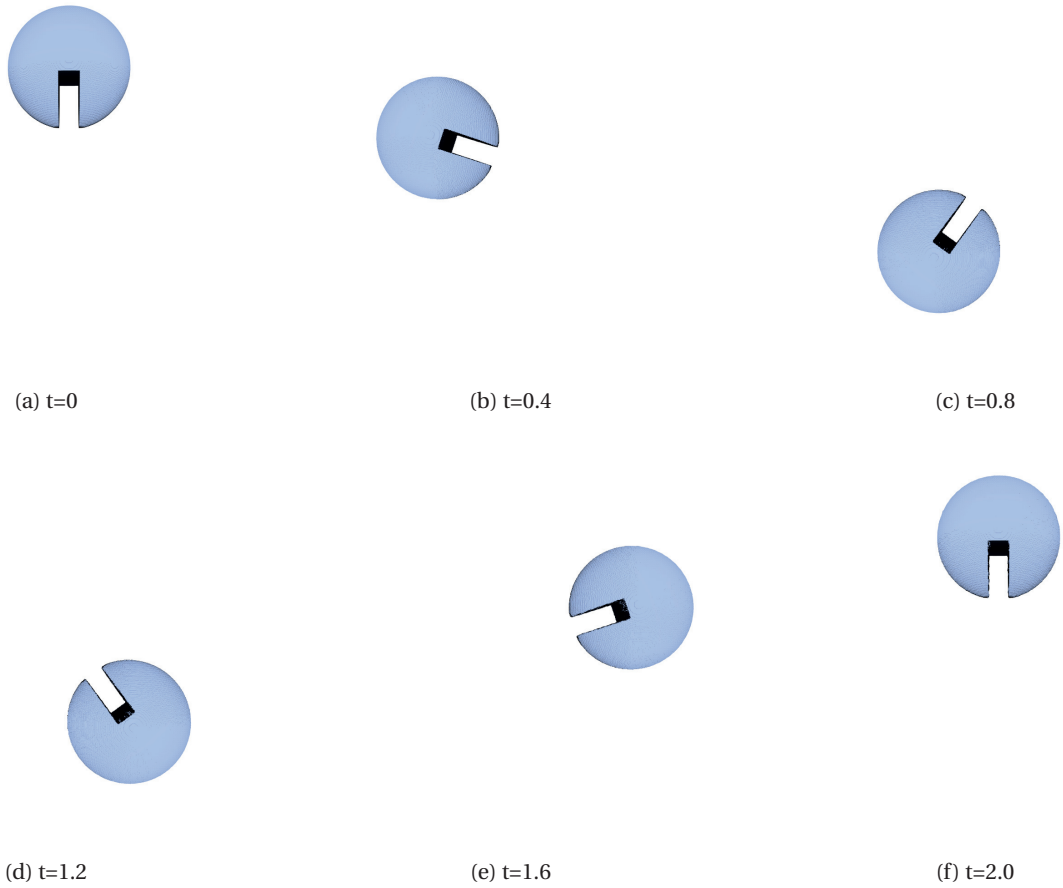
Figure 2.2: Slice of sphere translation test case with $h_{min} = 1.302 \times 10^{-3}$ and $CFL = 6.6$.

The CFL number $CFL = |u|\Delta t / h_{min}$ is an adimensional number representing the maximal number of cells traversed by the fluid in one iteration. Classical VOF methods require the condition $CFL \leq 1$ because they perform advection using an explicit scheme and consider fluxes between adjacent cells. The numerical domain of dependence needs to contain the true domain of dependence for stability [22]. Our method is however unconditionally stable with respect to the CFL and does not require a linear system to be solved. Low CFL values even tend to introduce more numerical diffusion due to the smaller timesteps and hence

more numerous projections. The characteristics method has been shown in [61] to converge in $\mathcal{O}\left(\frac{h^2}{\Delta t} + h + \Delta t\right)$ where the $\frac{h^2}{\Delta t}$ error term stems from the interpolations at each timestep. In order to keep the interpolation error small, the CFL should therefore be kept fairly large. The goal is therefore to choose the CFL number so as to strike a balance between error due to numerical diffusion at low CFL values and error arising from the failure to follow the characteristics at high CFL values.

Restoring the sphere to its initial location could of course be done very accurately by setting the timestep to $\Delta t = 0.6$ since the characteristics are straight lines. This however informs us little about the performance of the scheme when translating free surfaces with timesteps of the same order as would be used in the full Navier Stokes equations resolution. For this benchmark test case we use a CFL of 6.6 which is typically of the order of what we would use for full Navier Stokes equations solving. The results are shown in Figure 2.1.

The interface is restored to its initial position but some bubbles are produced inside the fluid due to the SLIC algorithm as illustrated in Figure 2.2. It can be seen that the transport algorithm produces bubbles inside the fluid bulk close to the free surface in the direction of the flow as shown in Figure 2.2 for $t = 0.6$ in the top right corner of the sphere after the translation in the first direction. It handles rather well the part of the free surface being transported towards the bulk of the fluid. At final time $t = 1.2$ the bubbles which previously prevented coarsening of the fluid have been filled by the decompression algorithm in the top right corner. They however reappeared in the direction of the transport of the fluid.



(a) t=0 (b) t=0.4 (c) t=0.8

(d) t=1.2 (e) t=1.6 (f) t=2.0

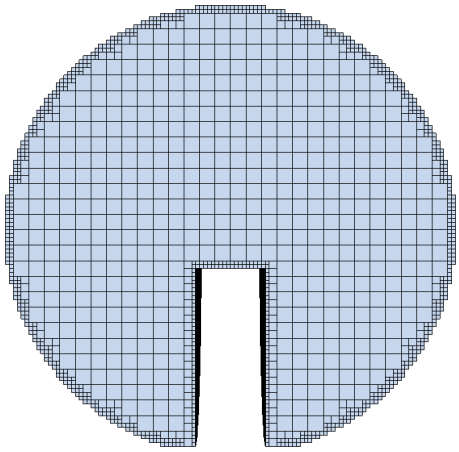
Figure 2.3: Zalesak's sphere test case.

2.1.2 Zalesak's sphere

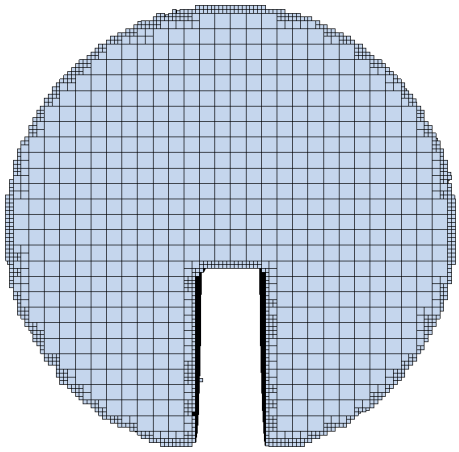
A 3-dimensional version of the classic Zalesak's disk test case is used as a numerical error and performance benchmarking tool for the octree convection scheme [15]. A 3-dimensional slotted sphere of radius 0.15 initially centered at $(0.5, 0.75, 0.5)$ is rotated a full circle around the point at $(0.5, 0.5, 0.5)$ in the $z = 0.5$ plane in a $[0,1] \times [0,1] \times [0,1]$ domain with final time $T = 2$. The slot is defined by the intersection with the sphere and the planes $y = 0.725, x = 0.475$ and $x = 0.525$. The numerical simulation is illustrated in Figure 2.3 and restores the initial slotted sphere at the final time with great accuracy. It illustrates the finest mesh size in the convergence plots in Figure 2.6.

Figure 2.4 shows a cut of the slotted sphere at initial and final times. It illustrates how the octree accurately captures the surface of the slotted sphere and how the interfacial cells remain fine throughout the simulation, while the inner cells remain coarse.

Unlike in the previous test case where the characteristics were straight lines, a crucial point to

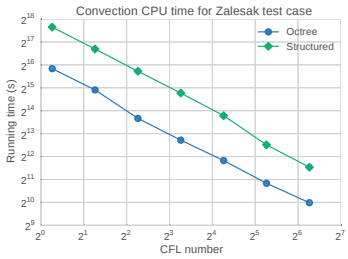


(a) t=0

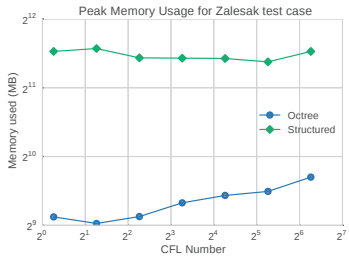


(b) t=2

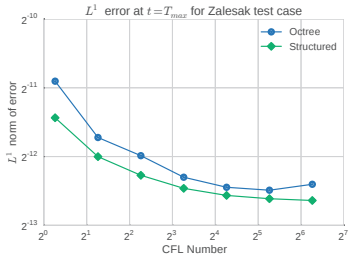
Figure 2.4: Octree mesh cut of Zalesak's sphere at initial and final times with a CFL number of 19.2 and $h_{min} = 2.6 \times 10^{-3}$. Only cells D with $\varphi(D) \geq 0.5$ are displayed. CPU time was 7 minutes. The mesh contains 191506 cells while its structured analog contains 884736 cells.



(a) Loglog plot of CPU time



(b) Loglog plot of Memory usage



(c) Loglog plot of L_1 Error

CFL	Octree	Structured
1.2	5.844×10^4	2.051×10^5
2.4	3.071×10^4	1.057×10^5
4.8	1.299×10^4	5.401×10^4
9.6	6.727×10^3	2.794×10^4
19.2	3.625×10^3	1.409×10^4
38.4	1.825×10^3	5.836×10^3
76.8	1.014×10^3	2.968×10^3

(d) CPU time in seconds

CFL	Octree	Structured
1.2	557.3	2955
2.4	521.6	3042
4.8	559.3	2769
9.6	642.3	2761
19.2	692.0	2752
38.4	720.6	2659
76.8	832.3	2954

(e) Memory usage in MB

CFL	Octree	Structured
1.2	5.218×10^{-4}	3.609×10^{-4}
2.4	2.958×10^{-4}	2.439×10^{-4}
4.8	2.463×10^{-4}	2.021×10^{-4}
9.6	1.982×10^{-4}	1.772×10^{-4}
19.2	1.791×10^{-4}	1.650×10^{-4}
38.4	1.739×10^{-4}	1.596×10^{-4}
76.8	1.846×10^{-4}	1.568×10^{-4}

(f) L_1 Error

Figure 2.5: CPU time, memory usage and error benchmark of the octree scheme for Zalesak's sphere for different CFL numbers and $h_{min} = 2.6 \times 10^{-3}$.

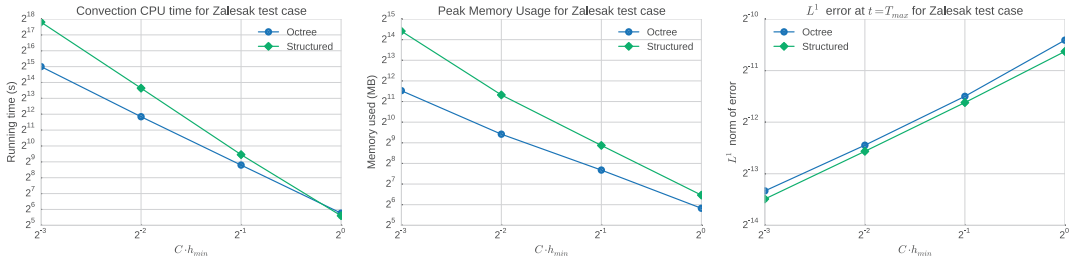
consider in the choice of numerical parameters is the CFL number here.

A comparison of several CFL values is shown in Figure 2.5. We set $h_{min} = 2.6 \times 10^{-3}$ and vary the timestep between 0.00125 and 0.08 to obtain the results. We set $l_{liquid} = l_{max} - 2$, hence $h_{liquid} = 1.02 \times 10^{-2}$ and we will discuss this choice below. As expected, running time decreases with order 1 and memory usage stays approximately constant. L^1 error decreases at first but then attains a minimum for very large CFL numbers. The use of Runge Kutta of order 4 to follow the characteristics allows large CFL numbers in the case of simple stationary velocity fields such as this one. Note that the octree performs about four times better both in terms of running time and memory usage with comparable errors for this fixed cell size. In a second comparison, we study the convergence of the schemes as both cell size and timestep tend to zero while the CFL stays constant. Even though higher CFL values yield lower running time and error, some splitting schemes for solving the full Navier Stokes equations may require a smaller timestep. We choose to study convergence for a CFL of 19.2.

For a fixed $CFL = 19.2$, the benchmarking consists in setting 4 different values of h_{min} for both the octree and fully structured schemes. The baseline value of h_{min} is $1/192$ and the simulation is run using finer meshes with respectively $C \cdot h_{min} = 1, 1/2, 1/4, 1/8$ where $C = 192$. The baseline timestep is set to $\Delta t_{ref} = 0.08$ and in order to keep a constant CFL value, the timestep is chosen such that the ratio $h_{min}/\Delta t$ remains constant.

Recall that for the octree, we have set $l_{liquid} = l_{max} - 2$, i.e. $h_{min} = h_{liquid}/4$.

Results are shown in Figure 2.6. Running time for the structured version grows with order 4



(a) Loglog plot of CPU time

(b) Loglog plot of Memory usage

(c) Loglog plot of L_1 Error

$C \cdot h_{min}$	Octree	Structured
1.0	5.440×10^1	4.869×10^1
0.5	4.446×10^2	7.011×10^2
0.25	3.674×10^3	1.278×10^4
0.125	3.278×10^4	2.285×10^5

(d) CPU time in seconds

$C \cdot h_{min}$	Octree	Structured
1.0	56.76	88.29
0.5	204.70	468.70
0.25	683.70	2556.00
0.125	2951.00	21810.00

(e) Memory usage in MB

$C \cdot h_{min}$	Octree	Structured
1.0	7.350×10^{-4}	6.314×10^{-4}
0.5	3.455×10^{-4}	3.177×10^{-4}
0.25	1.791×10^{-4}	1.650×10^{-4}
0.125	9.693×10^{-5}	8.694×10^{-5}

(f) L_1 Error

Figure 2.6: CPU time, memory usage and error benchmark of the octree scheme for Zalesak's sphere. h_{min} is the smallest cell size, $C = 192$ and $CFL = 19.2$.

compared to order 3 for the octree. Memory growth is order 3 for structured and 2 for the octree. Convergence rates of the L^1 error seem to be of order 0.9 for both structured grid and octree.

Our earlier choice of fixing $l_{liquid} = l_{max} - 2$ is justified as follows. At a relative liquid cell size of $h_{liquid} = 4\ h_{min}$ and for h_{min} of the order of cell sizes benchmarked above, the interfacial cells are responsible for most of the memory usage and CPU time. This is illustrated in Figure 2.7, in which we plot the maximal number of cells through the simulation for the different mesh sizes for both structured grid and octree. Note that for the structured grid we only include liquid cells since only those are stored in memory and for the octree, non-leaf cells are counted as well. We plot the number of cells of level l_{max} and we can clearly see they dominate. Number of cells for the structured grid behaves as $1/h_{min}^3$ compared to $1/h_{min}^2$ for the octree. Taking coarser relative liquid cell sizes of $h_{liquid} = 8\ h_{min}$ or larger produces a larger error and negligible efficiency gains in terms of CPU time and memory usage.

Note that the number of interfacial cells grows in $O\left(1/{h_{min}}^2\right)$ and the constraint $h_{liquid} = 4\ h_{min}$ implies that asymptotically the number of liquid cells grows in $O\left(1/{h_{min}}^3\right)$. However, we have $1/{h_{liquid}}^3 = 1/(64\ {h_{min}}^3)$ which makes the multiplicative constant of the cubic asymptotic growth of liquid cells much smaller than the multiplicative constant of the quadratic growth of interfacial cells. We therefore expect a gain of one order in terms of number of cells for the octree scheme compared to the structured scheme until the cubic growth of liquid cells becomes dominant. In performed experiments, even for the finest levels of refinement, we have not reached the point where growth of number of cells becomes cubic for the octree.

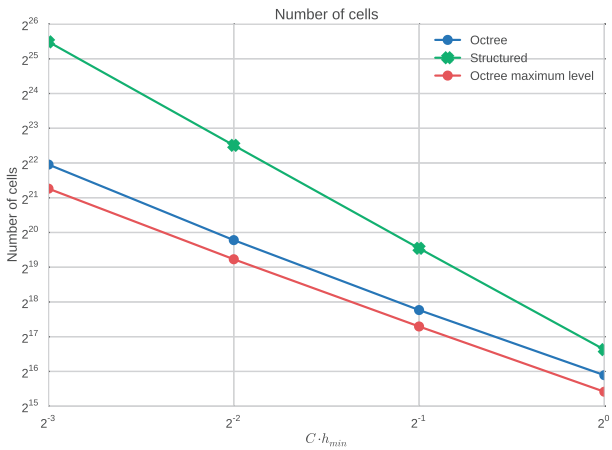


Figure 2.7: Maximal number of cells for Zalesak's sphere test case

2.1.3 Time-dependent vortex

The next test case advects a sphere around a time-dependent vortical velocity field and is described in [77]. A 3-dimensional sphere of radius 0.15 initially centered at $(0.7, 0.5, 0.5)$ is subjected to the following velocity field with final time $T = 2$:

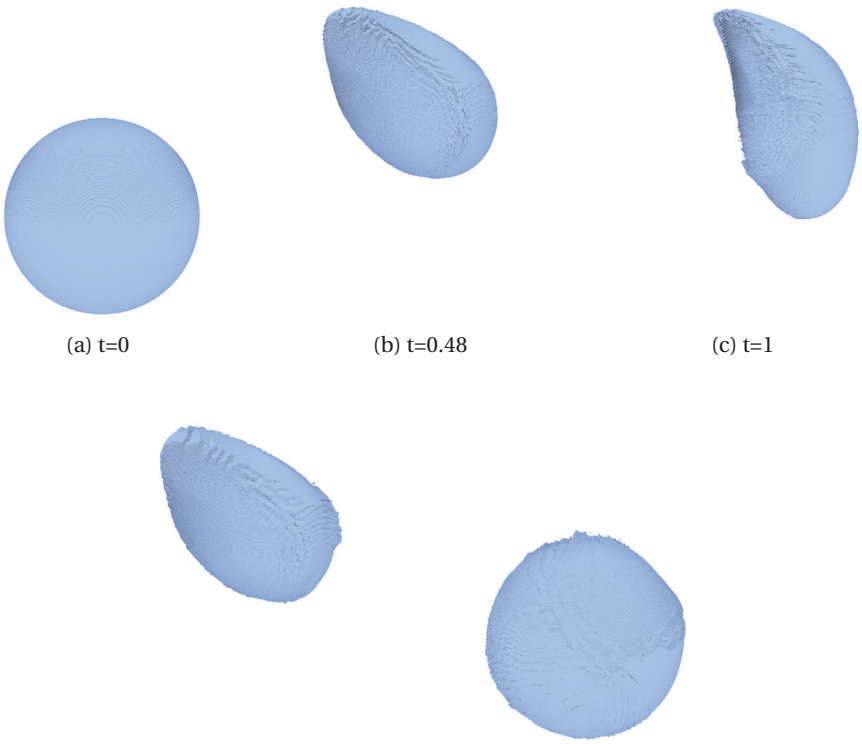
$$u(x,y,z,t) = \begin{pmatrix} \sin^2(\pi x)\cos(\pi t/2)\left(\sin(\pi(y-0.5)) - \sin(\pi(z-0.5))\right) \\ \sin^2(\pi y)\cos(\pi t/2)\left(\sin(\pi(z-0.5)) - \sin(\pi(x-0.5))\right) \\ \sin^2(\pi z)\cos(\pi t/2)\left(\sin(\pi(x-0.5)) - \sin(\pi(y-0.5))\right) \end{pmatrix}$$

The sphere gets deformed from $t = 0$ to $t = 1$ as shown in Figure 2.8 and is returned to its initial position at $t = 2$. A delicate point in this test case is the choice of the CFL since the velocity is zero at $t = 1$ so a range of CFL values are covered throughout the simulation. We will instead set the maximal CFL which is attained at $t = 0$.

Results are shown in Figure 2.9.

This time the L^1 error is lower for the octree scheme than for the structured scheme due to the more complex velocity field. Indeed, the heuristic decompression and relaxed coarsening algorithms which are designed for easier coarsening of liquid octree cells correct some spurious bubbles formed in the bulk of the fluid.

A maximal CFL of 2.7 seems to give best results for the octree structure both in terms of error and memory usage. For lower and higher maximal CFL values, spurious bubbles start to appear and prevent coarsening the bulk of the fluid which results in increase of error and memory usage. This test case in contrast with Zalesak's sphere test case highlights the fact that the optimal CFL number is heavily dependent on the velocity field considered.



(a) t=0 (b) t=0.48 (c) t=1

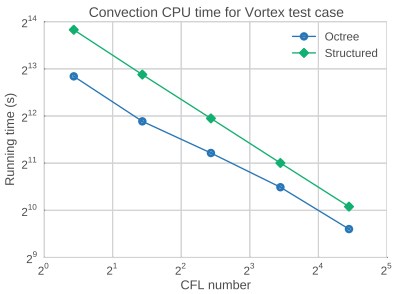
(d) t=1.52 (e) t=2

Figure 2.8: Time-dependent vortex test case.

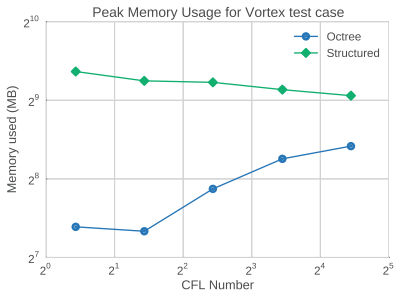
A convergence analysis is performed with $CFL = 2.7$. This time, the baseline value of h_{min} is $1/96$ and the simulation is again run using finer meshes with respectively $C \cdot h_{min} = 1, 1/2, 1/4, 1/8$ where $C = 96$. The baseline timestep is set to $\Delta t_{ref} = 0.02$ and in order to keep a constant CFL value, the timestep is chosen such that the ratio $h_{min}/\Delta t$ remains constant. Results are shown in Figure 2.10.

Running time and memory usage seem to behave as observed in Zalesak's sphere test case but as the octree scheme conserves its approximately first order convergence in terms of error, the convergence for the structured algorithm seems to stall. This behaviour comes from spurious bubbles which are formed in the bulk of the fluid but they can be corrected with adaptive time-stepping which is not discussed here. However, the relaxed coarsening algorithm of the octree seems to correct this problem and allow convergence.

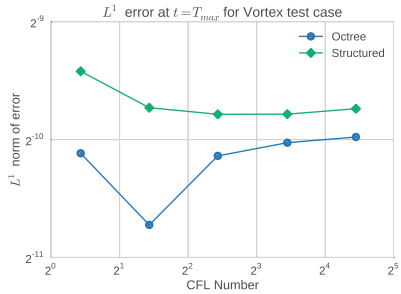
As a conclusion to the analysis of the octree convection scheme, it appears that despite the first order convergence of the numerical scheme, the scheme is able to accurately advect interfaces as shown in the Zalesak's sphere test case. The advantage of being able to choose CFL numbers much larger than 1 speeds up calculations and reduces computation time. CPU time and memory usage growth for the octree are roughly one order lower than for the



(a) Loglog plot of CPU time



(b) Loglog plot of Memory usage



(c) Loglog plot of L_1 Error

CFL	Octree	Structured
1.35	7.339×10^3	1.455×10^4
2.7	3.787×10^3	7.528×10^3
5.4	2.378×10^3	3.954×10^3
10.9	1.438×10^3	2.053×10^3
21.8	7.773×10^2	1.081×10^3

(d) CPU time in seconds

CFL	Octree	Structured
1.35	167.7	659.9
2.7	161.3	607.7
5.4	234.4	599.8
10.9	305.5	562.4
21.8	341.7	533.4

(e) Memory usage in MB

CFL	Octree	Structured
1.35	9.010×10^{-4}	1.459×10^{-3}
2.7	5.912×10^{-4}	1.178×10^{-3}
5.4	8.873×10^{-4}	1.133×10^{-3}
10.9	9.589×10^{-4}	1.134×10^{-3}
21.8	9.914×10^{-4}	1.171×10^{-3}

(f) L_1 Error

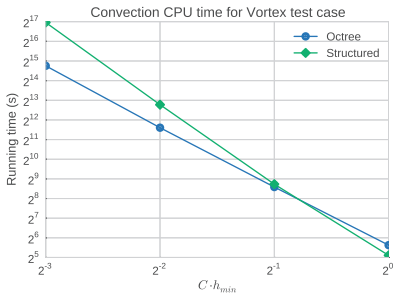
Figure 2.9: CPU time, memory usage and error benchmark of the octree scheme for the time-dependent vortex test case for different values of maximal CFL numbers and $h_{min} = 2.6 \times 10^{-3}$.

structured scheme when refining timestep and cell size. For time-dependent velocity fields, the relaxed coarsening algorithm of the octree scheme can even allow convergence when the structured algorithm does not. The adaptivity of the octree scheme and the usage of fast algorithms allows accurate results in little time and memory usage.

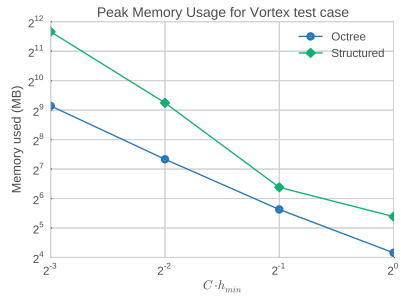
2.1.4 Leveque-Enright's test case

We call this test case the Leveque-Enright test case since it was proposed by Leveque in [78] and popularized by Enright's work on particle level sets [15]. A three-dimensional vortical velocity field deforms a sphere and the opposite velocity field is then applied starting from $t = T/2$ in order to restore the sphere back to its original position at $t = T$. The test case is different from the time-dependent test case in Section 2.1.3 in the sense that a substantial deformation is applied to the sphere which stretches it to the point of being reduced to an elongated filament. Another notable difference is that the deformation velocity field does not vary continuously in time; a constant in time velocity field is applied from $t = 0$ to $t = T/2$ and the opposite velocity field is applied from $t = T/2$ to $t = T$.

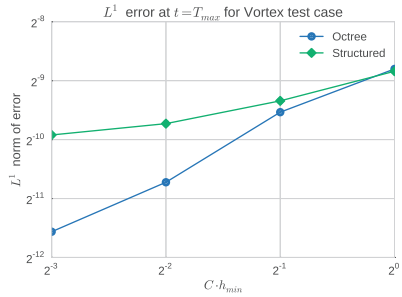
A 3-dimensional sphere of radius 0.15 initially centered at $(0.35, 0.35, 0.35)$ is subjected to the following velocity field from $t = 0$ to $t = T/2$ with final time $T = 2$:



(a) Loglog plot of CPU time



(b) Loglog plot of Memory usage



(c) Loglog plot of L_1 Error

$C \cdot h_{min}$	Octree	Structured
1.0	4.943×10^1	3.447×10^1
0.5	3.841×10^2	4.249×10^2
0.25	3.115×10^3	7.018×10^3
0.125	2.765×10^4	1.275×10^5

(d) CPU time in seconds

$C \cdot h_{min}$	Octree	Structured
1.0	17.86	41.84
0.5	49.49	83.25
0.25	161.20	607.60
0.125	565.00	3233.00

(e) Memory usage in MB

$C \cdot h_{min}$	Octree	Structured
1.0	2.241×10^{-3}	2.181×10^{-3}
0.5	1.348×10^{-3}	1.540×10^{-3}
0.25	5.912×10^{-4}	1.178×10^{-3}
0.125	3.304×10^{-4}	1.031×10^{-3}

(f) L_1 Error

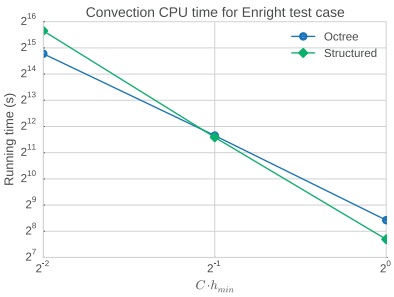
Figure 2.10: CPU time, memory usage and error benchmark of the octree scheme for the time-dependent vortex test case. h_{min} is the smallest cell size, $C = 96$ and $CFL = 2.7$.

$$u(x,y,z,t) = \begin{pmatrix} 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z) \\ -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z) \\ -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z) \end{pmatrix}$$

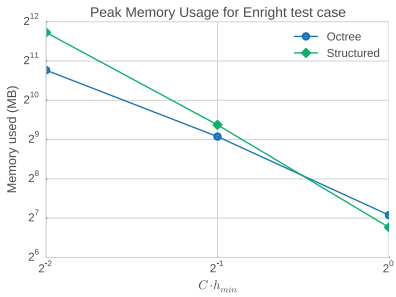
The opposite velocity field is then applied from $t = T/2$ to $t = T$.

Results are shown in Figure 2.11 and pictures of the test case in Figure 2.12. For this somewhat extreme test case, we can see that the octree performs faster than the structured only for the finest mesh. This phenomenon can be explained by the fact that the thin fluid form at $t = 1$ presents a high surface to volume ratio compared to the sphere. When this ratio is high, the overhead of refining the octree on the whole surface is large compared to the structured grid for coarse meshes. The smaller h_{min}, the more cells can be coarsened and eventually the octree is faster. The L^1 error behaves as h_{min} for both octree and structured mesh.

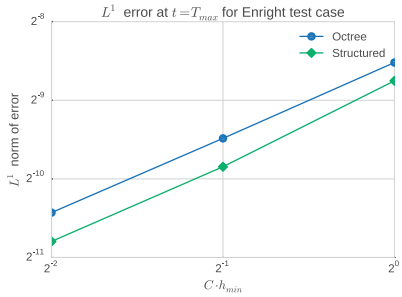
Figure 2.12 shows good agreement between fluid shapes at $t = 0.6$ and $t = 1.4$ where the two should be identical. At final time $t = 2.0$, the sphere is well restored but some extra flotsam is present around the sphere. This type of behaviour is caused by the SLIC algorithm and has been documented for instance in [29].



(a) Loglog plot of CPU time



(b) Loglog plot of Memory usage



(c) Loglog plot of L_1 Error

$C \cdot h_{min}$	Octree	Structured
1.0	3.436×10^2	2.070×10^2
0.5	3.221×10^3	3.077×10^3
0.25	2.806×10^4	5.149×10^4

(d) CPU time in seconds

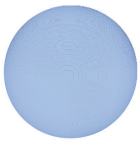
$C \cdot h_{min}$	Octree	Structured
1.0	135.1	108.9
0.5	539.8	663.9
0.25	1742.0	3387.0

(e) Memory usage in MB

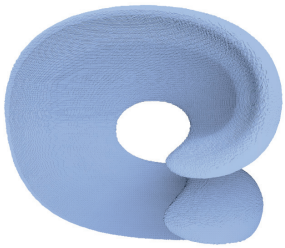
$C \cdot h_{min}$	Octree	Structured
1.0	2.725×10^{-3}	2.319×10^{-3}
0.5	1.396×10^{-3}	1.087×10^{-3}
0.25	7.254×10^{-4}	5.625×10^{-4}

(f) L_1 Error

Figure 2.11: CPU time, memory usage and error benchmark of the octree scheme for the Leveque-Enright test case. h_{min} is the smallest cell size, $C = 192$ and $CFL = 6.6$.

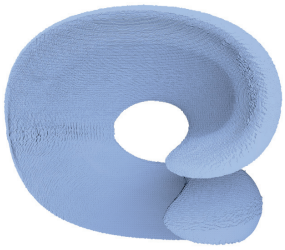


(a) t=0

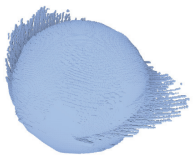


(b) t=0.6

(c) t=1.0



(d) t=1.4



(e) t=2.0

Figure 2.12: Leveque-Enright test case.

2.2 3D Navier Stokes free surface results with the octree-based scheme

2.2.1 Stoker's test case

A first benchmark of the full Navier-Stokes octree scheme is done on the Stoker test case, see for instance [79]. A parallelepipedic cavity $[-50,50] \times [0,2] \times [0,3]$ is initially filled with a water height of $H_l = 2.0$ for $x < 0$ and a height of $H_r = 1.0$ for $x \geq 0$. Under the effect of gravity, a shock wave then propagates along the x-axis towards positive values of x. A second rarefaction wave propagates along the x-axis in the opposite direction and an intermediate constant water height of H_m and velocity u_m then forms in between both waves.

This problem can be solved analytically assuming a simplified model called the inviscid shallow water or Saint-Venant equations. The problem has been solved with the structured analog of our scheme in [44]. Therein can also be found a more precise description of the analytical solution which we just give here. Let $c_l = \sqrt{gH_l}$, $c_r = \sqrt{gH_r}$ and $c_m = \sqrt{gH_m}$ where g is the gravitational acceleration. Let us also define the speed of the hydraulic jump $W = \frac{H_m u_m}{H_m - H_r}$. The analytical water height $\bar{H}$ is then given by

$$\bar{H}(x,t) = \begin{cases} H_l & \text{if } x < -c_l t \\ \dfrac{\left(2c_l - \frac{x}{t}\right)^2}{9g} & \text{if } -c_l t < x < (u_m - c_m)t \\ H_m & \text{if } (u_m - c_m)t < x < Wt \\ H_r & \text{if } Wt < x \end{cases} \tag{2.1}$$

for $x \in [-50,50]$ and $t \in [0,T]$ where $T = 5$. Using the relation $u_m = 2(c_l - c_m)$ and the approximation $H_m \simeq 1.45384$ which stems from the solution of a polynomial equation of degree 6, we compute $\bar{H}$.

We compare the analytical solution to the inviscid shallow water with three different solutions by different refinements of the octree. Physical properties of the water for the Navier-Stokes simulations are taken as $\rho = 10^3 kg/m^3$ and $\mu = 10^{-3} kg/(m\ s)$. Simulation parameters for the coarse octree mesh are $h_{min} = 0.06$ and $\Delta t = 0.05$, for the medium mesh are $h_{min} = 0.03$ and $\Delta t = 0.025$ and for the fine octree mesh are $h_{min} = 0.015$ and $\Delta t = 0.0125$. Like for convection test cases, we choose $l_{liquid} = l_{max} - 2$. Mesh sizes for the uniform tetrahedral mesh are respectively $H = 0.374$, $H = 0.187$ and $H = 0.0936$. Boundary conditions are set to slip conditions. Results plotted for times $t = 0.t, 2.0, 3.5$ and $t = 5.0$ are shown in Figure 2.13.

Results are very similar to those obtained in [44]. The wave profiles given by the Navier-Stokes equation are smoother and present over- and undershoots. They however match closely the analytical curve outside of the rarefaction and shock wave. The point of onset of both the rarefaction wave and shock wave also seem to be in good agreement between both models although the center point of the shocks do not agree as for example in [80]. This difference is expected due to the non-conservative nature of our scheme.

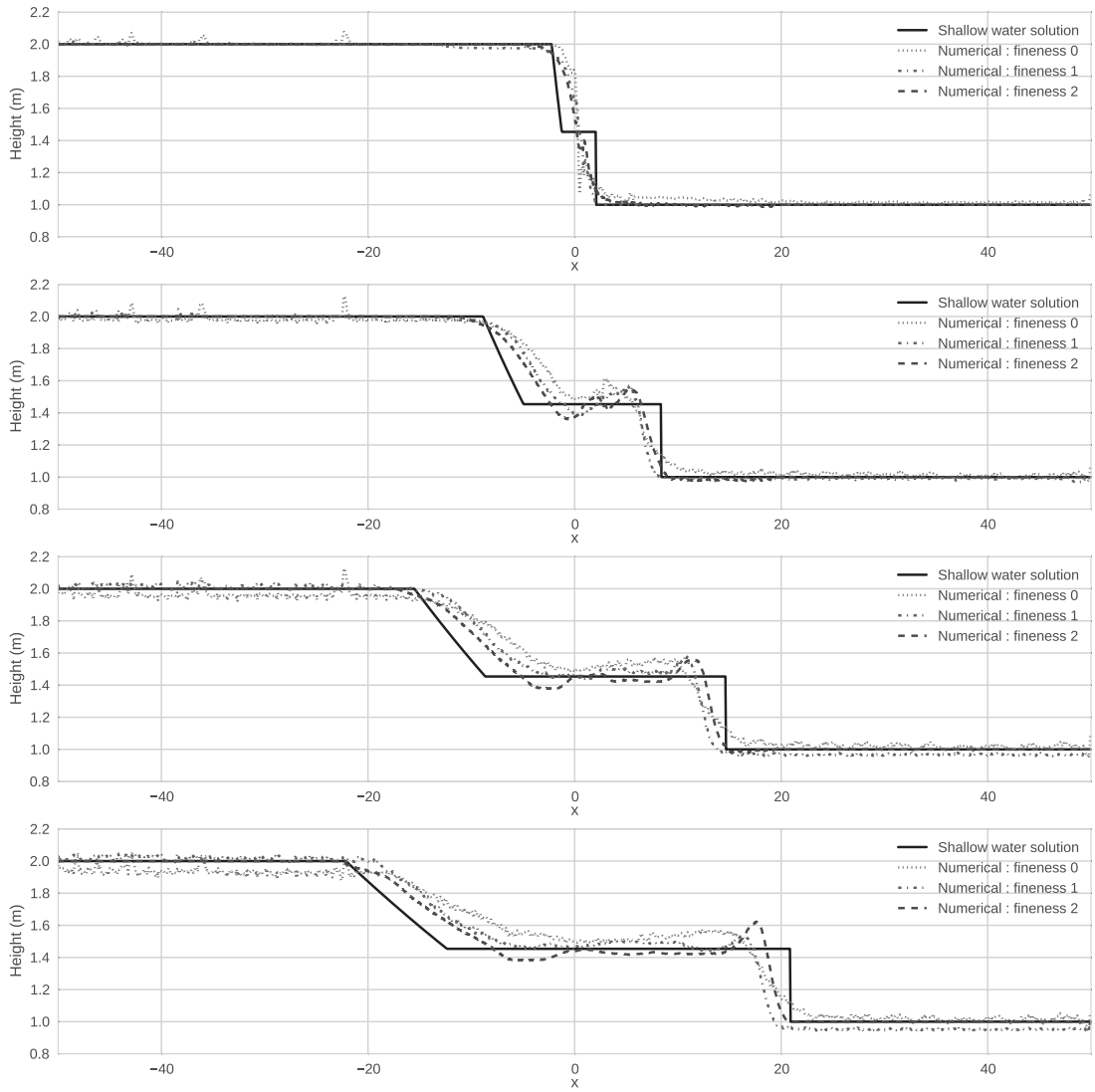


Figure 2.13: Wave profiles for the Stoker test case at times $t = 0.5, 2.0, 3.5$ and $t = 5.0$.

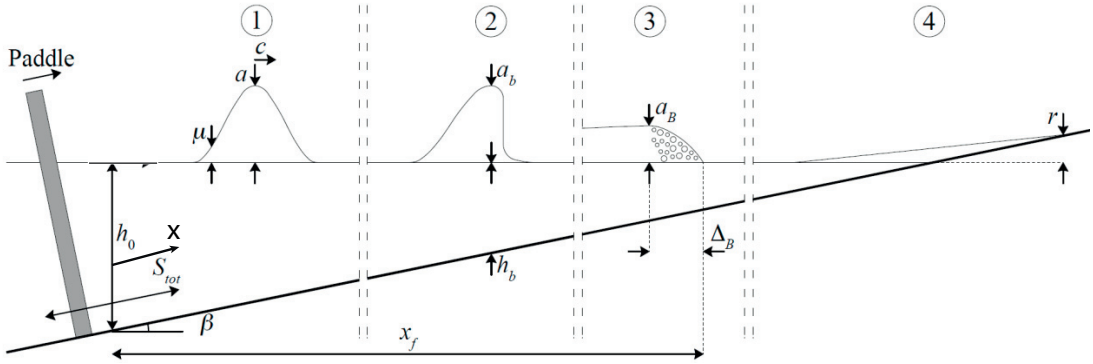


Figure 2.14: Sketch of the experimental setup for the paddle-generated wave

2.2.2 Pseudo-2D paddle-generated wave simulations in a tilted cavity

A paddle-generated wave in a tilted 3D cavity will be used as a benchmark to determine the accuracy of the numerical scheme for solving free surface flows governed by the Navier Stokes equations. Although the wave propagates in a 3D cavity, the cavity is narrow enough such that the wave only exhibits 2D features. The experimental wave profile measurements [45] were kindly provided by the *VAW* at *ETH Zürich*.

The experimental cavity is $14m$ long, $0.5m$ wide and 0.7m high. One of the side walls is lined with observational glass windows and the other wall along with the bottom is lined with smooth PVC. A pneumatic wave-generating piston is mounted at one end of the cavity and is controlled with electrical impulses sent by a computer. The setup has been successfully used in several experiments by the VAW, see [81, 82, 83].

The experimental setup is illustrated in Figure 2.14. The cavity is tilted at an angle β. A plate pushes the water parallel to the bottom of the cavity with a velocity profile designed to generate waves with a given height relative to the still water depth. The relative wave height is noted R_h and the still water depth is always taken as $0.2m$. In order to define more accurately what is meant by still water depth, let us define the x-axis as parallel to the bottom of the cavity, directed from the paddle to the other end of the cavity. The paddle does a sweeping motion from $x = -x_0$ to $x = x_0$. The still water depth is the maximal water depth when the plate is at position $x = 0$ in the middle of the plate sweep. Note that the starting and end coordinates of the plate depend on the desired relative wave height R_h. The relative wave height R_h is then the ratio of the maximal wave height to the still water depth $0.2m$.

We will simulate all combinations of parameters $\beta = 1°$, $\beta = 6°$ and $R_h = 0.3, 0.5, 0.7$.

Two types of sensors have been set up for measuring the wave profiles, UltraSonic Distance Sensors (USDSs) and Capacitance Wave Gauges (CWGs). The USDSs are placed at the top of the cavity and send an acoustic signal downwards towards the water surface. The signal reflects

off the water free surface and the free surface height is deduced from the time taken for the signal to return to the sensor. In [45], an estimate of $3mm$ is given for expected measurement errors. As a downside, this sensor can give spurious values if the signal is reflected away from the sensor. This can happen in particular if the free surface under the sensor is at a steep angle. Since it does happen when $d = 6°$, the less accurate CWG sensors are also used.

CWGs are vertical enamel coated wires attached to the top of the cavity whose capacitance varies linearly with respect to water height. After calibration, they can then be used to determine water height. The obtained wave profiles are then noisy and show jumps of up to $8mm$ between sampling points and the measurement error is estimated in [45] to be also up to $8mm$. It is why they are smoothed with a Savitzky-Golay filter [84]. The Savitzky-Golay filter smoothes the signal by fitting low-level polynomials with the linear least squares method with successive subsets of adjacent data points. The polynomials are also used to derive a smooth velocity which is used for the simulation part to determine the imposed velocity at the paddle. As parameters for the filter we used a data point window size of 71 data points and polynomials of degree 3. Numerical wave profiles were also smoothed for visualization clarity with the Savitzky-Golay filter but to a much lesser extent, with a data point window size of 7 data points and polynomials of degree 3.

Physical properties of the water for the Navier-Stokes simulations are taken as $\rho = 10^3 kg/m^3$ and $\mu = 10^{-3} kg/(m\ s)$. Simulation parameters for the coarse octree mesh are $h_{min} = 3.79e-3$ and $\Delta t = 0.0125$, for the medium mesh are $h_{min} = 1.89e-3$ and $\Delta t = 0.00625$ and for the fine octree mesh are $h_{min} = 9.49e-4$ and $\Delta t = 0.003125$. We choose $l_{liquid} = l_{max} - 2$ to have coarse cells approximately of same size as tetrahedra. Mesh sizes for the uniform tetrahedral mesh are respectively $H = 0.0102$, $H = 0.00510$ and $H = 0.00255$. Since the wave only exhibits two-dimensional features, we have truncated the cavity to a width of 0.125 for lower running times. Boundary conditions are set to no-slip conditions. A generated wave is shown on a coarse mesh in Figure 2.15.

A bubble enrichment stabilization is used for the finite element problem along with a GMRES solver and ILU preconditioner. The paddle movement is simulated in practice by computing at each step which nodes are intersecting the region that has been swept by the paddle and imposing the paddle velocity at those points. The octree is aligned with the water level and therefore not with the cavity. It is refined to level l_{max} at the bottom of the cavity to capture the cavity slope, at the wall in contact with the paddle and at the water free surface but not on the lateral sides which are aligned with the octree and hence captured exactly.

Results are shown in Figures 2.16 to 2.27. For each combination of experimental parameters $d = 1°, 6°$ and $R_h = 0.3, 0.5, 0.7$ we first plot the evolution of the wave height over time at four fixed points in the cavity and then snapshots of the numerical wave overlaid on photographs. Four pairs (USDS and CWG) of sensors placed at coordinates $x_1 = 0.677m$, $x_2 = 1.177m$, $x_3 = 1.677m$ and $x_4 = 2.177m$. The sensors provide water height data from times $t = 0s$ to $t = 4s$ at a sampling rate of $100Hz$. Note that the pneumatic-driven paddle moves with a slight

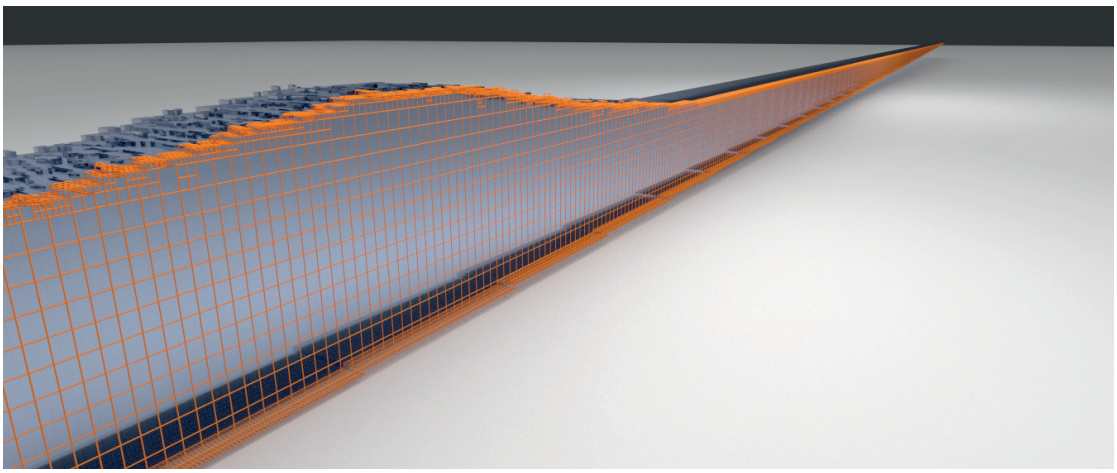


Figure 2.15: Render of a coarse octree wave with $h_{min} = 3.79e-3$

delay compared to the electrical input it receives. Wave profiles have therefore been shifted in order to correct for this fact. Despite this, we have measurements at four different points in the cavity which allows us to see how the wave from numerical simulations compares to the experimental one. Also, wave height profiles are compared.

A high-speed camera was used to capture photographs which are used to compare static in time wave profiles with our simulated wave profiles. The camera is aimed between sensors 3 and 4 and the CWG wires can be seen hanging from the top of the cavity. The photographs have been fitted together by matching initial still water heights and CWG wires with their known positions in the numerical cavity. Note that the CWG wires in the photographs are situated at the lateral wall of the cavity farther away from the camera and the ones placed in the numerical cavity are at the lateral wall closer to the cavity. The parallax then had to be estimated to match the positions of the virtual CWG wires with the experimental ones.

Figures 2.16 to 2.21 show results for $d = 1°$. For these experimental parameters, the wave does not break in the shown data. However, for $R_h = 0.7$, the wave does break soon after sensor 4. We achieve very good agreement between numerical and experimental profiles. It seems that for $R_h = 0.7$ where the wave presents a sharper and higher crest, the mesh needs to be refined more than for lower smoother waves in order for the wave profile to be captured. Note that in Figure 2.20, the USDSs show spurious values on the middle plots and those should be disregarded.

For $d = 6°$, we can see the waves breaking and causing a splash. These types of simulations are typically more difficult since they cannot be simulated by the shallow water model and require more elaborate models. Turbulence, fine-scale drops of water and air bubbles also occur when breaking and these aspects are not captured by our model. Figures 2.22 to 2.27 show results for these parameters. Again, spurious values of the USDS sensors should be disregarded.

H	h_{min}	Δt	Advection	Order	Stokes	Order
0.0102	3.79e-3	0.0125	4.5	×	2	×
0.00510	1.89e-3	0.006125	23	2.35	25	3.64
0.00255	9.49e-4	0.0030625	370	4	794	4.98

Table 2.1: Running times (seconds) per timestep for the wave $d = 1°$ and $R_h = 0.5$

Numerical wave profiles show very good agreement with the experimental wave profile despite the fact that the model does not capture turbulence and the air cushion that is formed beneath the breaking wave as seen in Figure 2.27. Notice that for $R_h = 0.7$, we observe as before that the higher sharper wave crest is more difficult to capture and the slight error in the solitary wave heights causes a more significant in the tongue shape during the breaking. Despite this, water wave profiles quickly match again after the wave breaking.

Running times per timestep for parameters $d = 1°$ and $R_h = 0.5$ are displayed in Table 2.1 for the three different mesh sizes and their associated timesteps. For coarser meshes, an advection step with the octree is slower than solving the Stokes' problem but as meshes get finer, the adaptivity of the octree makes it faster than solving the Stokes problem on a regular grid.

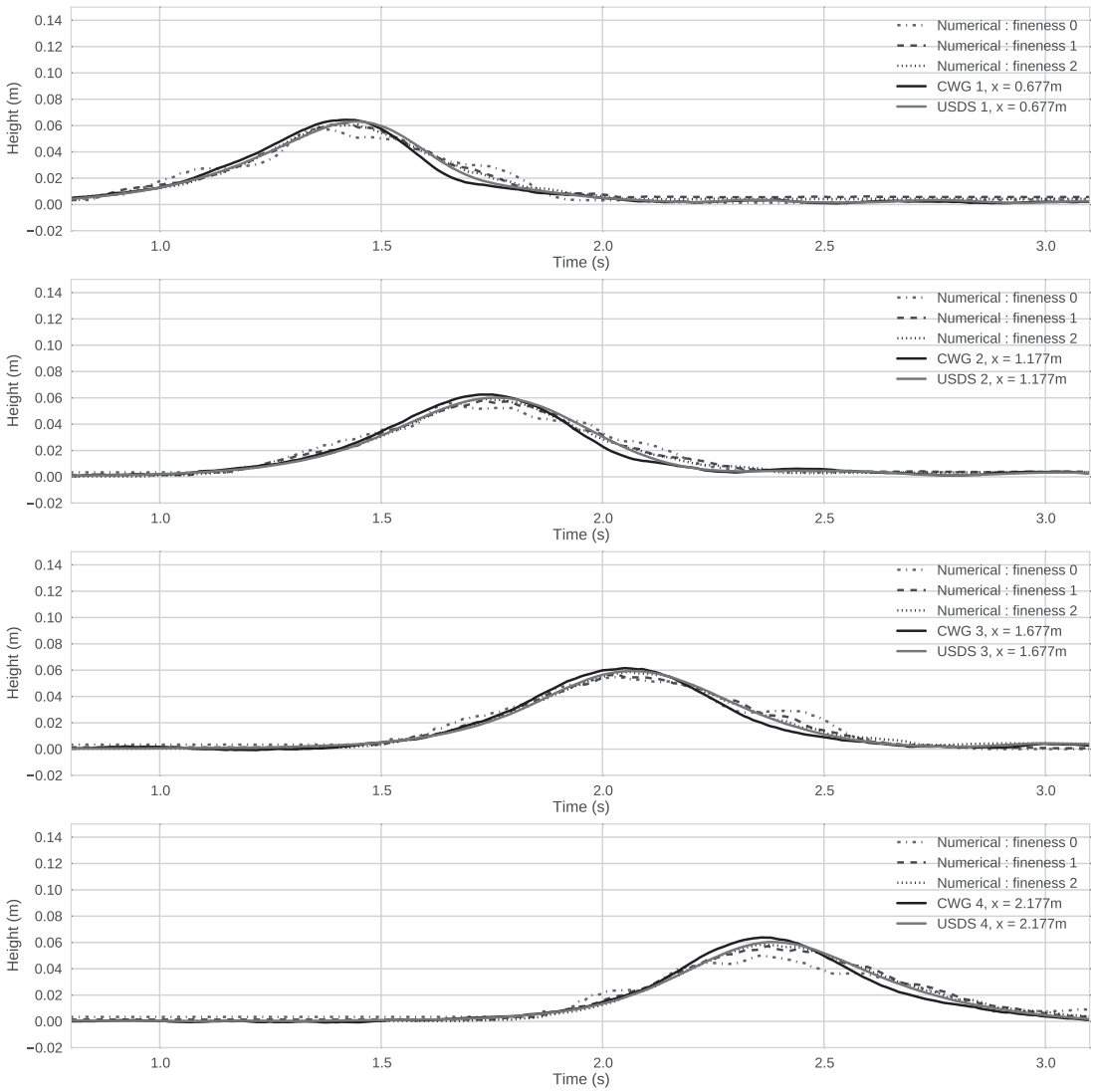
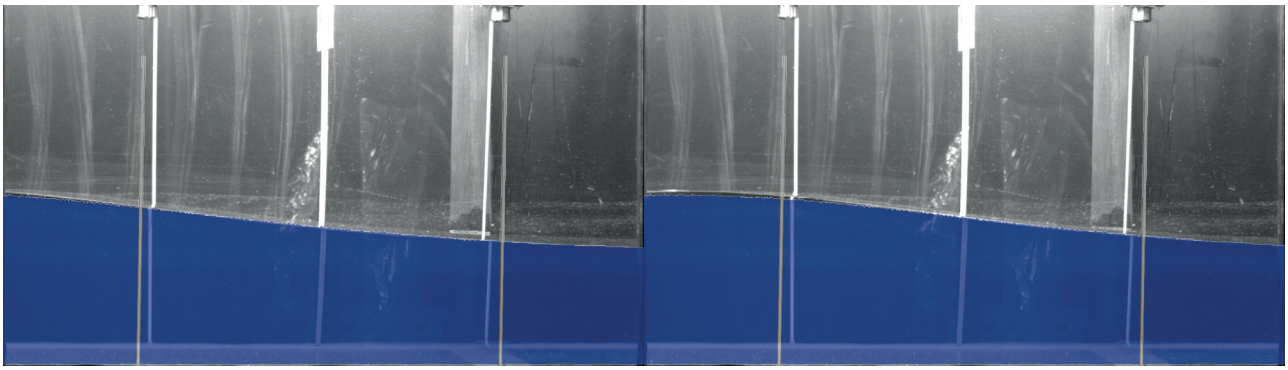
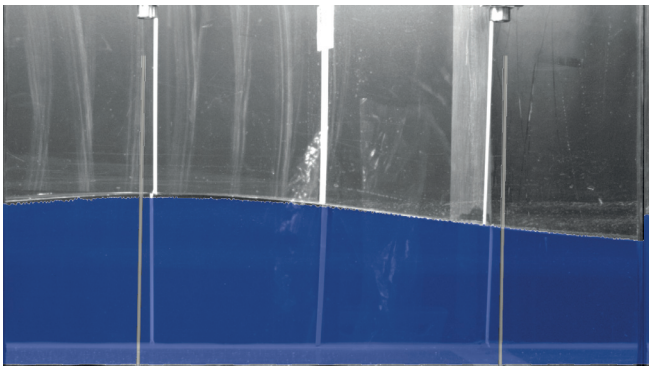


Figure 2.16: Pointwise height of the free surface throughout time on four different sensors for the wave test case for experimental parameters $d = 1°$ and $R_h = 0.3$.

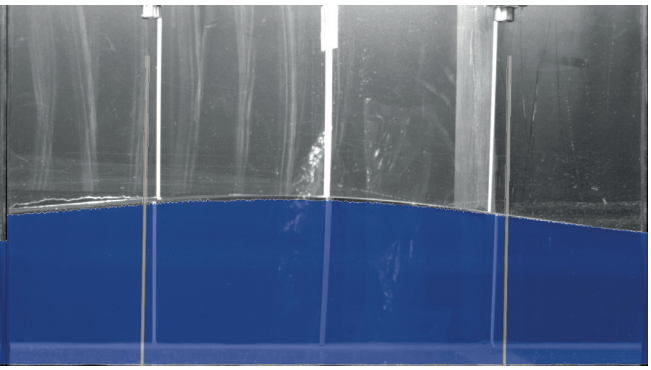


(a) $t = 1.98s$

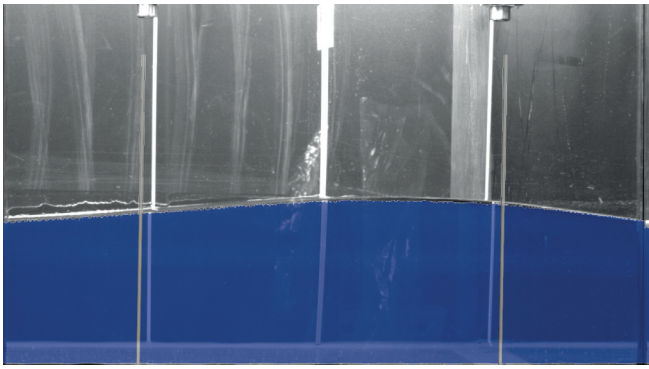
(b) $t = 2.08s$



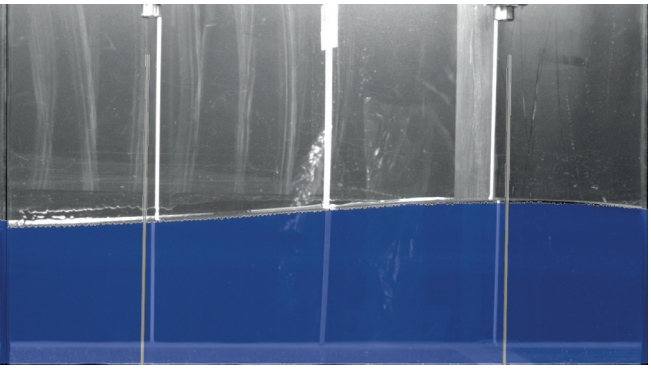
(c) $t = 2.18s$



(d) $t = 2.28s$



(e) $t = 2.38s$



(f) $t = 2.48s$

Figure 2.17: Overlay of the numerical wave profile in blue on top of the high speed pictures of the experimental wave for parameters $d = 1°$ and $R_h = 0.3$.

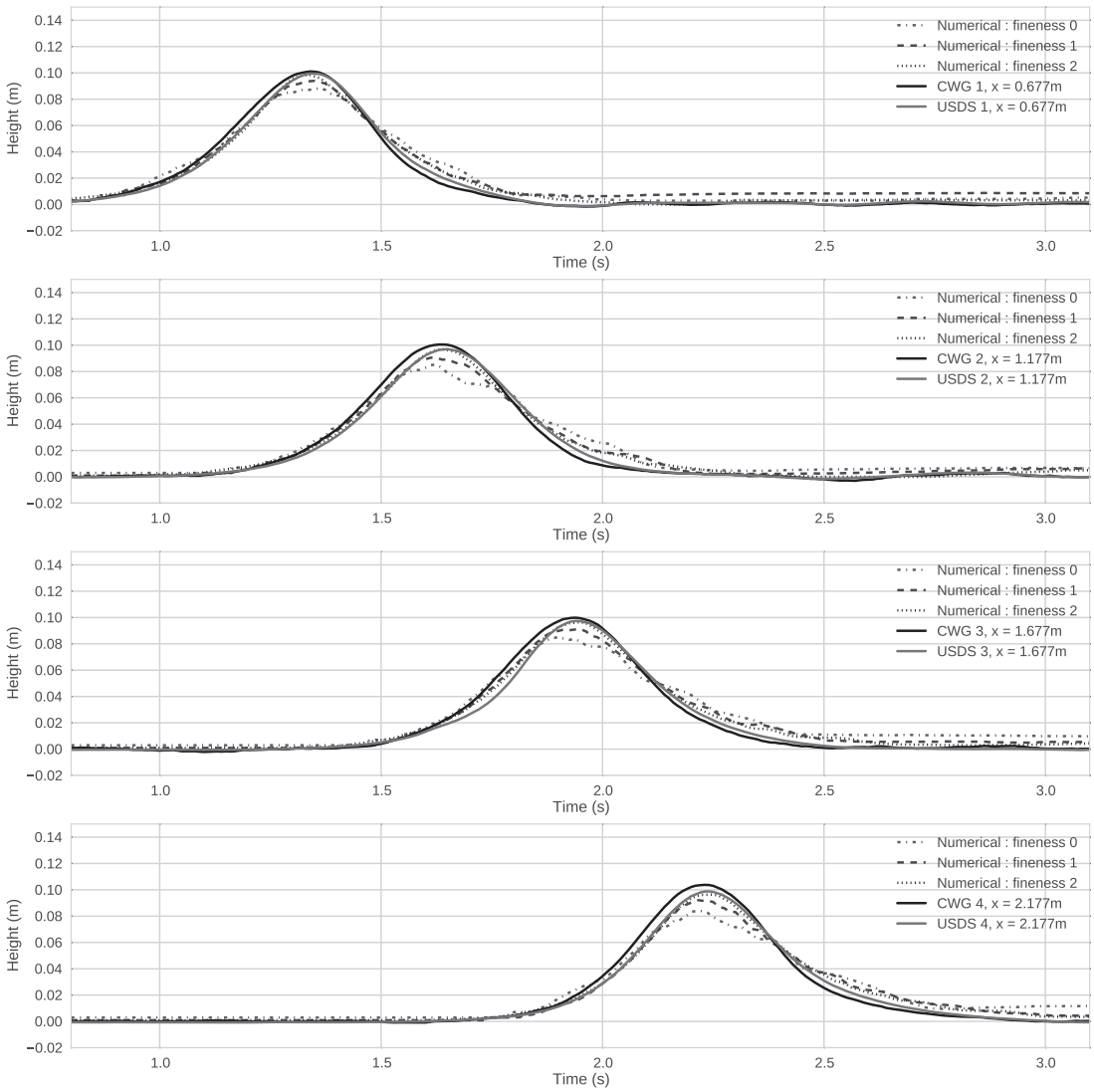
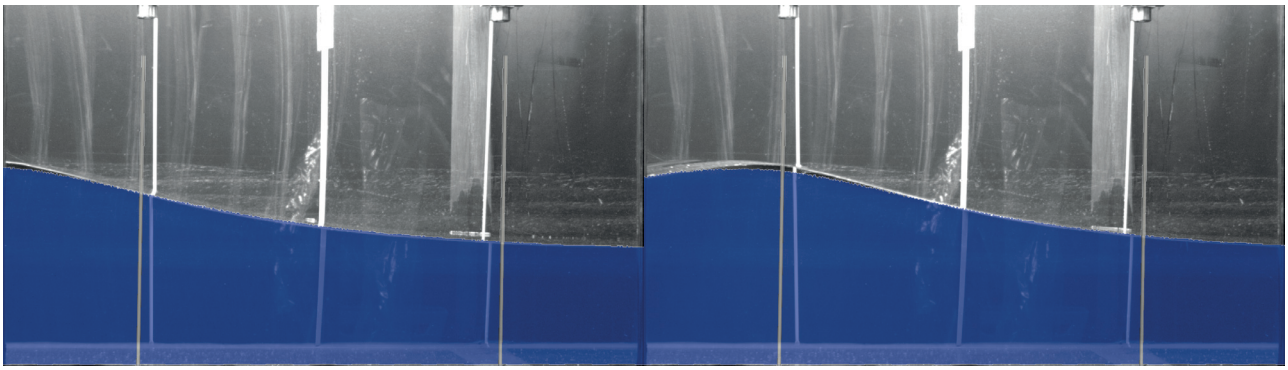
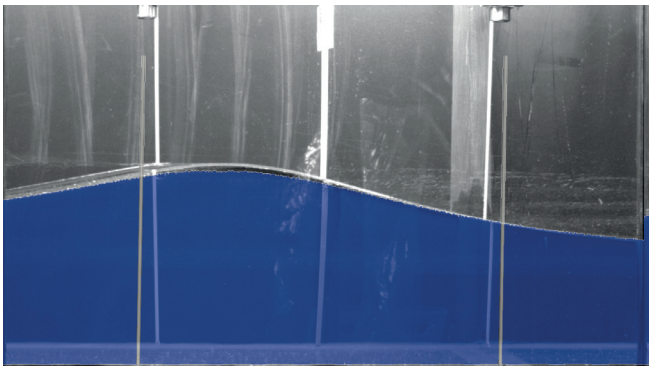


Figure 2.18: Pointwise height of the free surface throughout time on four different sensors for the wave test case for experimental parameters $d = 1°$ and $R_h = 0.5$.

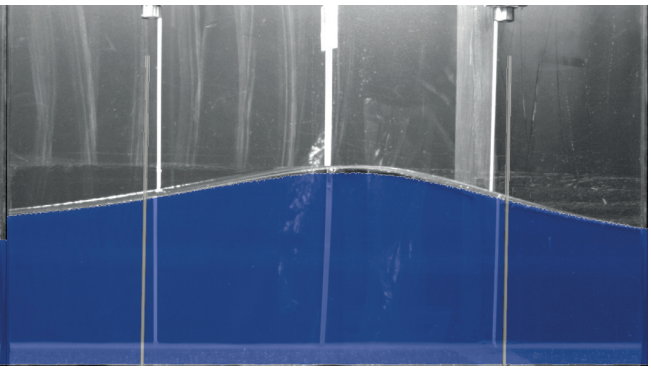


(a) $t = 1.84s$

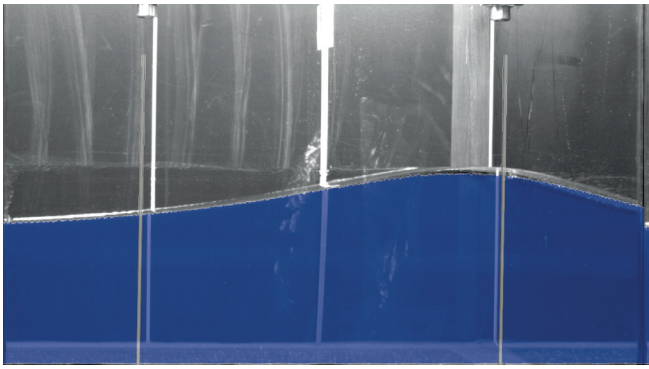
(b) $t = 1.94s$



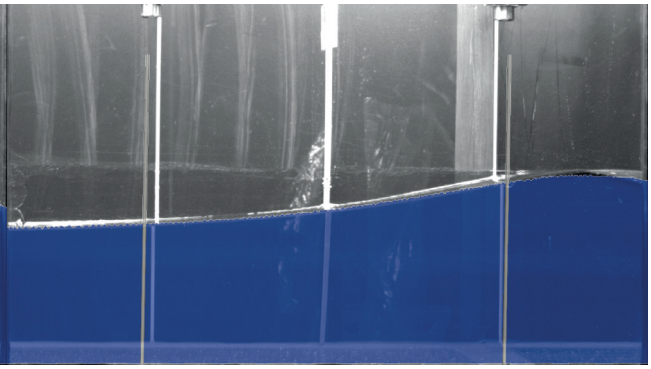
(c) $t = 2.04s$



(d) $t = 2.14s$



(e) $t = 2.24s$



(f) $t = 2.34s$

Figure 2.19: Overlay of the numerical wave profile in blue on top of the high speed pictures of the experimental wave for parameters $d = 1°$ and $R_h = 0.5$.

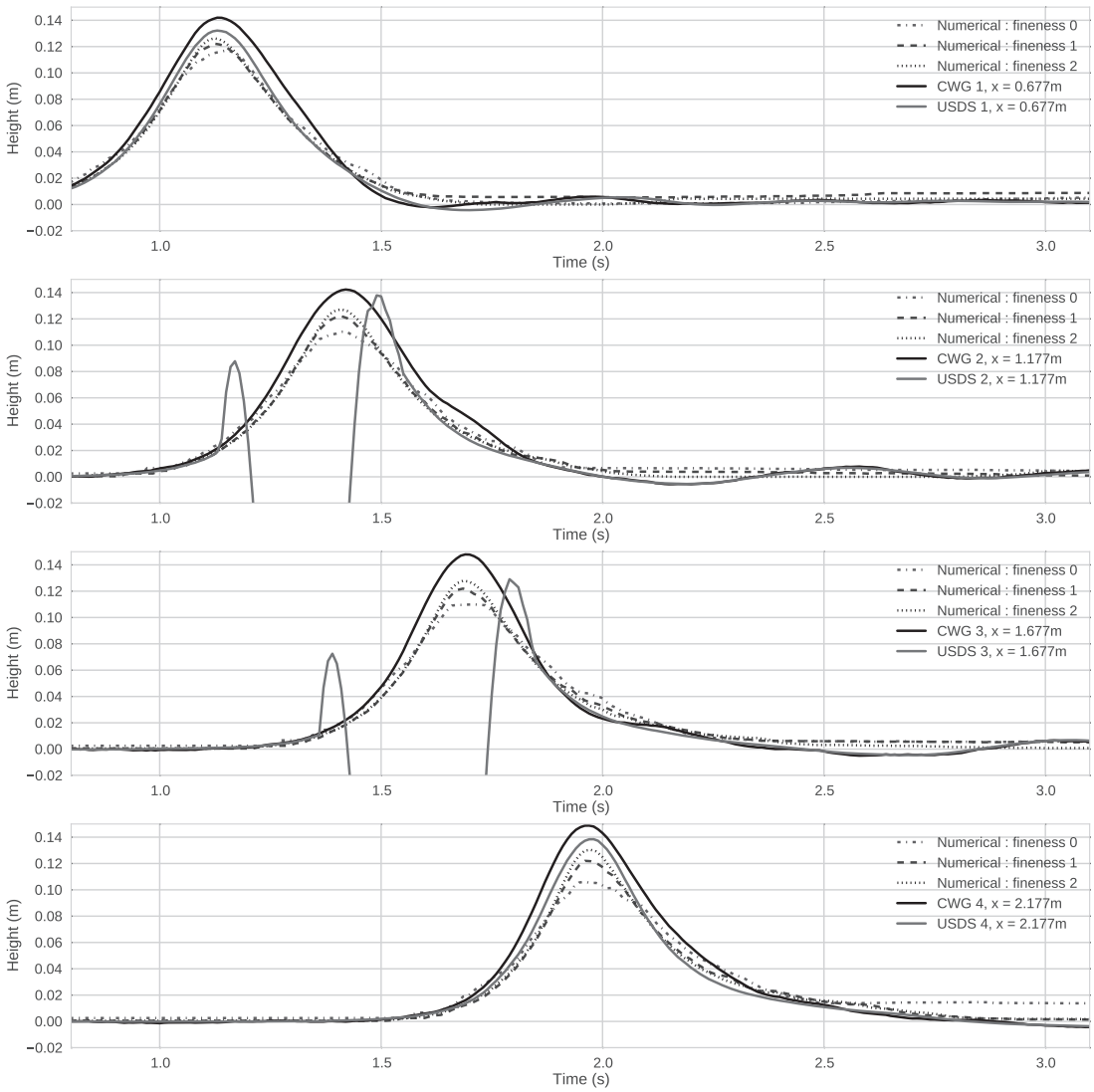
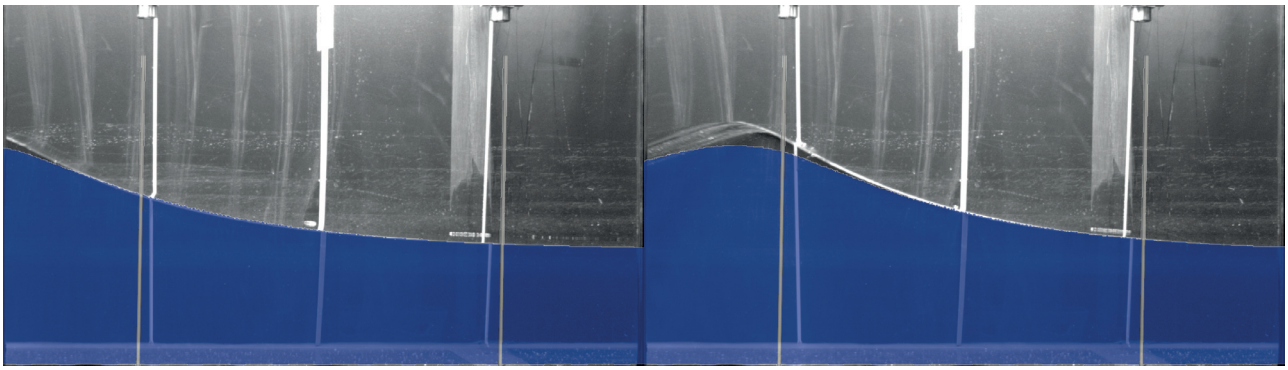
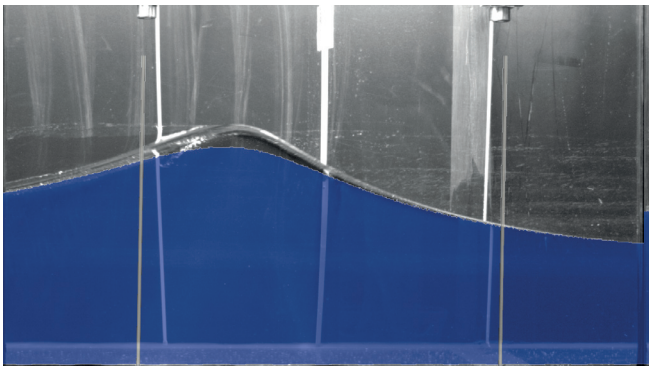


Figure 2.20: Pointwise height of the free surface throughout time on four different sensors for the wave test case for experimental parameters $d = 1°$ and $R_h = 0.7$.

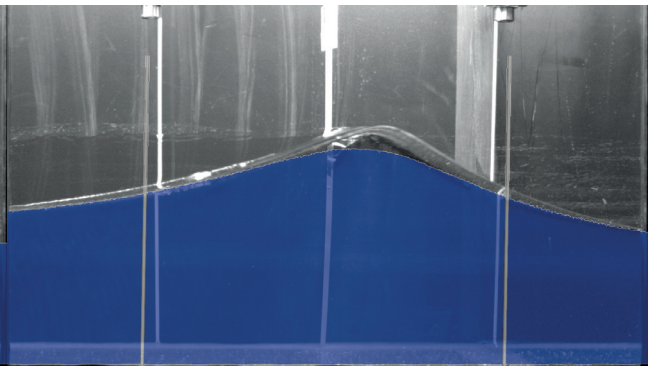


(a) $t = 1.63s$

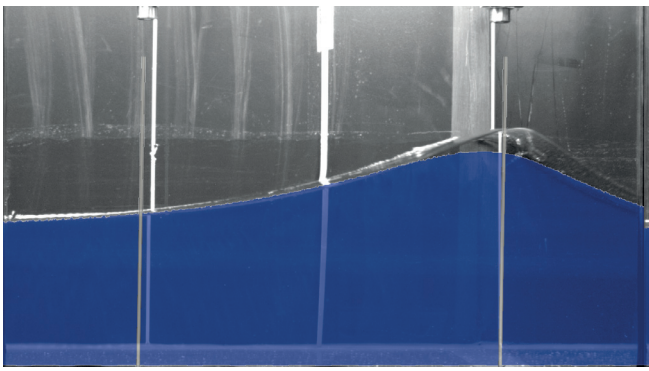
(b) $t = 1.73s$



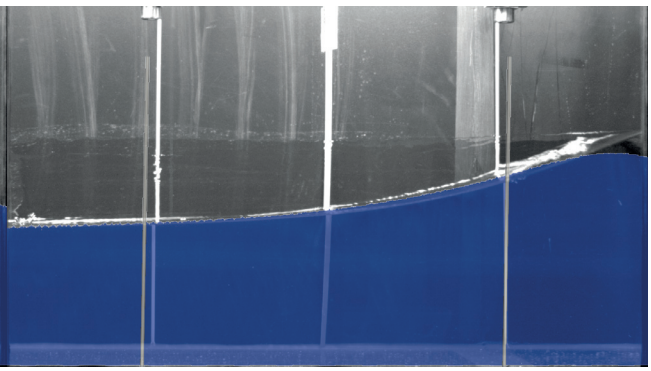
(c) $t = 1.83s$



(d) $t = 1.93s$



(e) $t = 2.03s$



(f) $t = 2.13s$

Figure 2.21: Overlay of the numerical wave profile in blue on top of the high speed pictures of the experimental wave for parameters $d = 1°$ and $R_h = 0.7$.

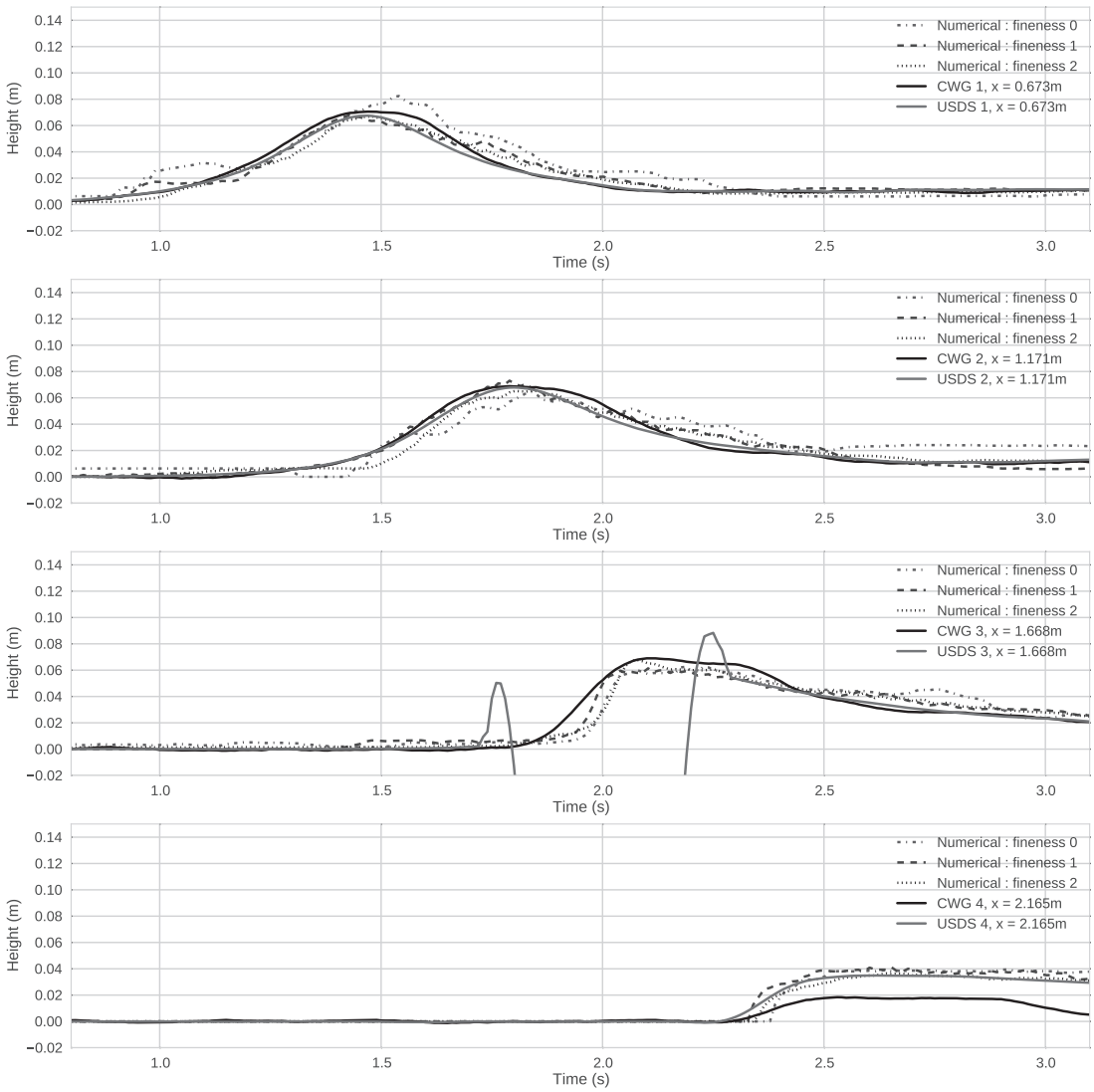
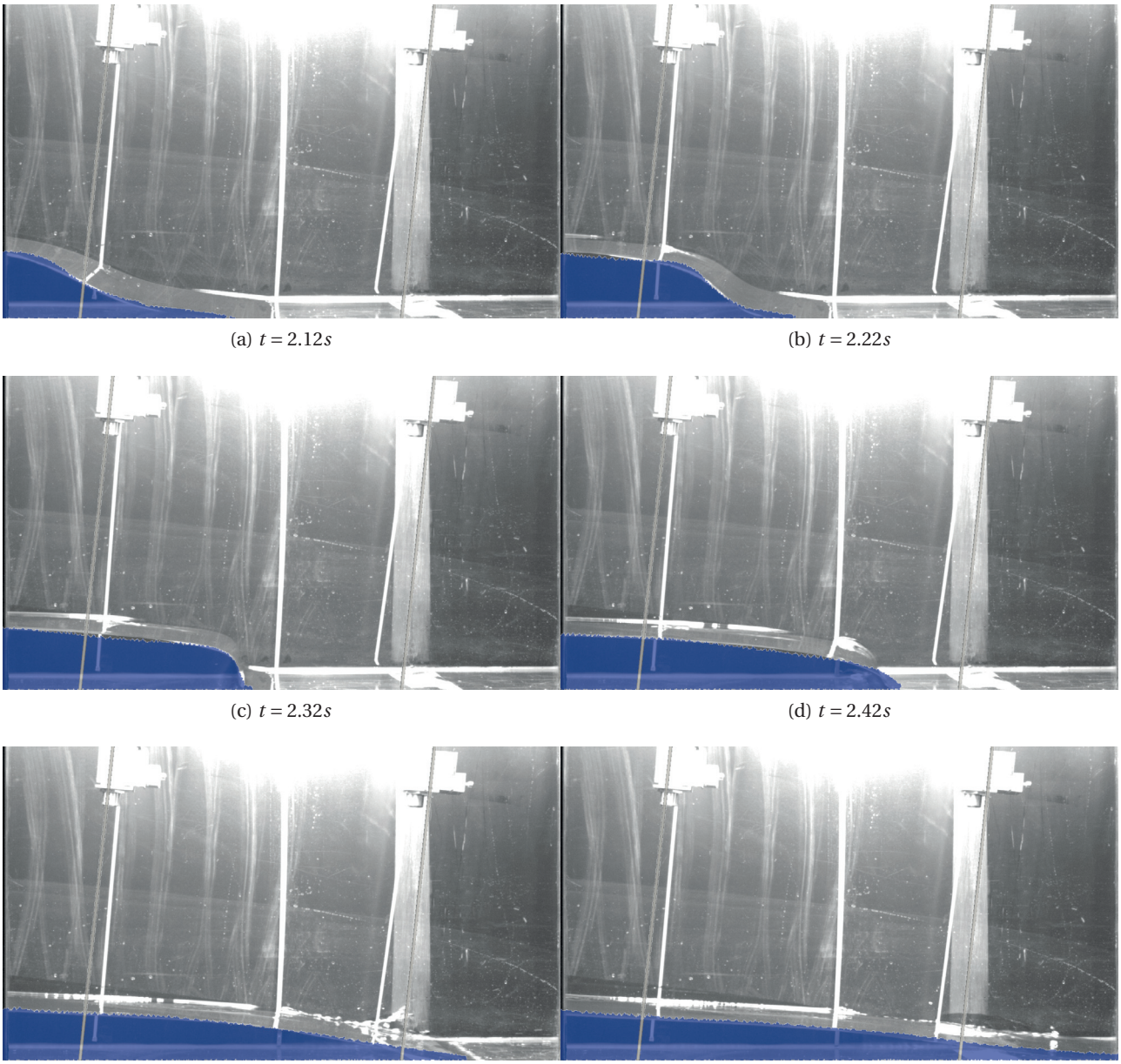


Figure 2.22: Pointwise height of the free surface throughout time on four different sensors for the wave test case for experimental parameters $d = 6°$ and $R_h = 0.3$.



(a) $t = 2.12s$ (b) $t = 2.22s$

(c) $t = 2.32s$ (d) $t = 2.42s$

(e) $t = 2.52s$ (f) $t = 2.62s$

Figure 2.23: Overlay of the numerical wave profile in blue on top of the high speed pictures of the experimental wave for parameters $d = 6°$ and $R_h = 0.3$.

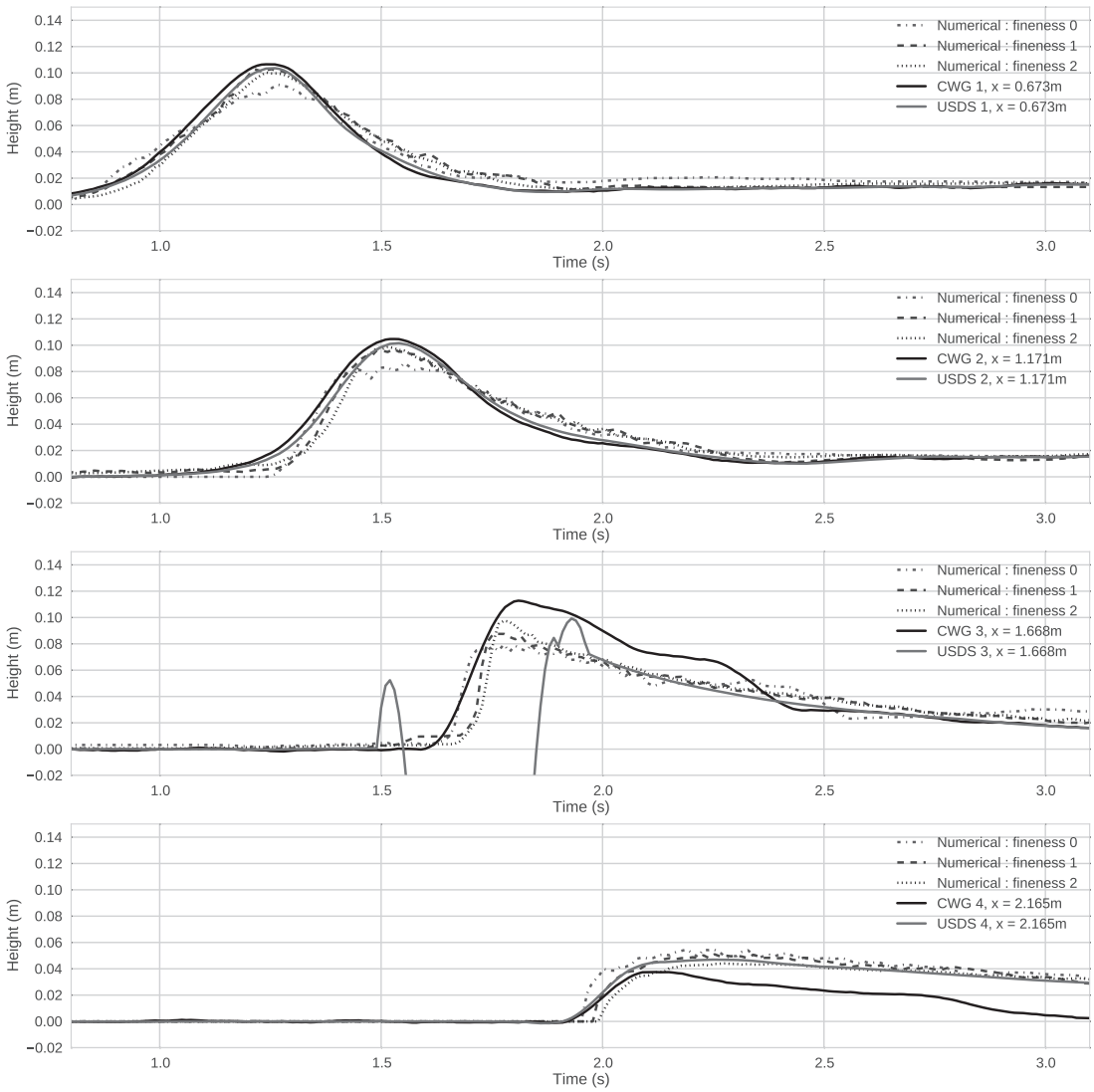
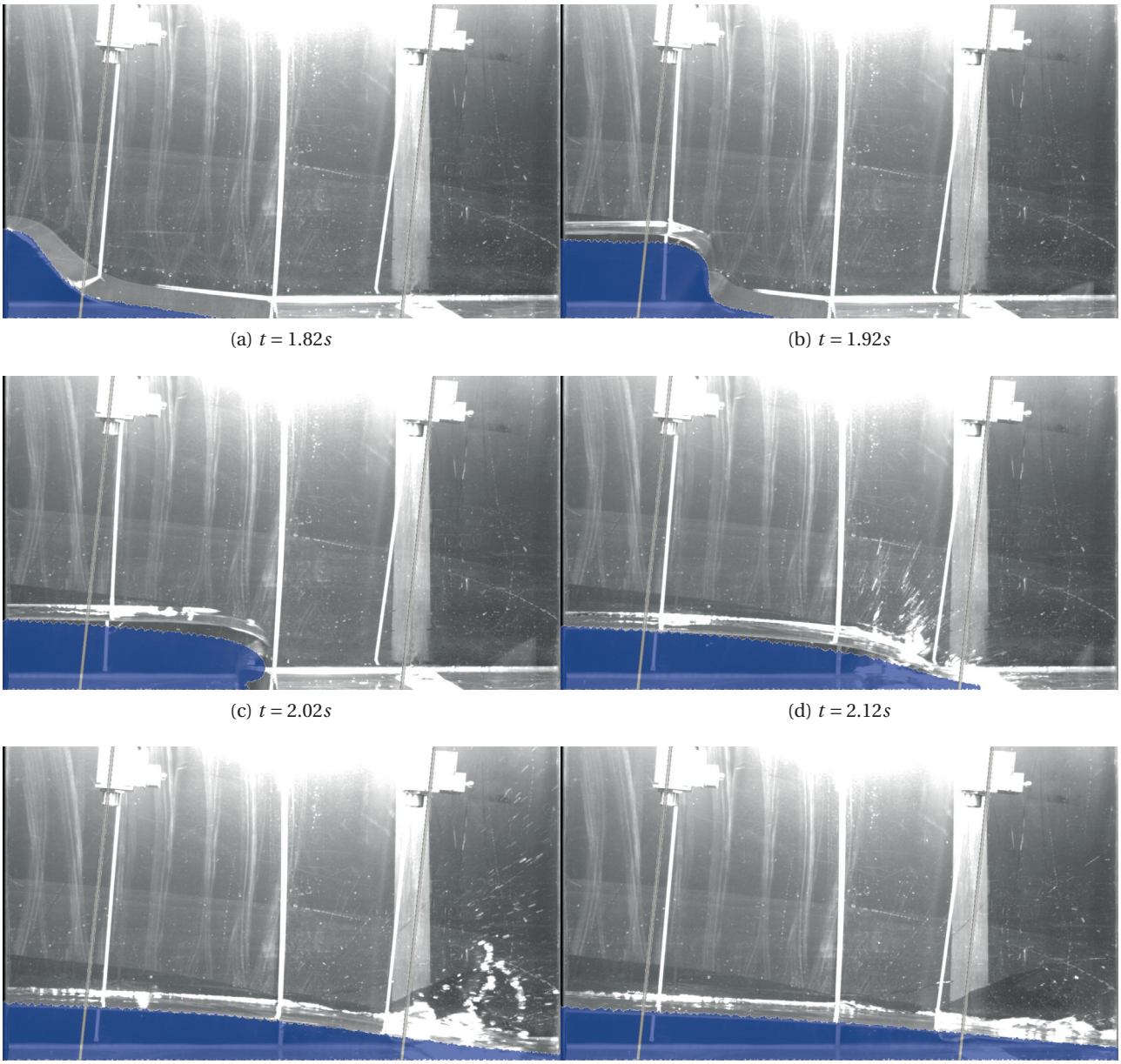


Figure 2.24: Pointwise height of the free surface throughout time on four different sensors for the wave test case for experimental parameters $d = 6°$ and $R_h = 0.5$.



(a) $t = 1.82s$ (b) $t = 1.92s$

(c) $t = 2.02s$ (d) $t = 2.12s$

(e) $t = 2.22s$ (f) $t = 2.32s$

Figure 2.25: Overlay of the numerical wave profile in blue on top of the high speed pictures of the experimental wave for parameters $d = 6°$ and $R_h = 0.5$.

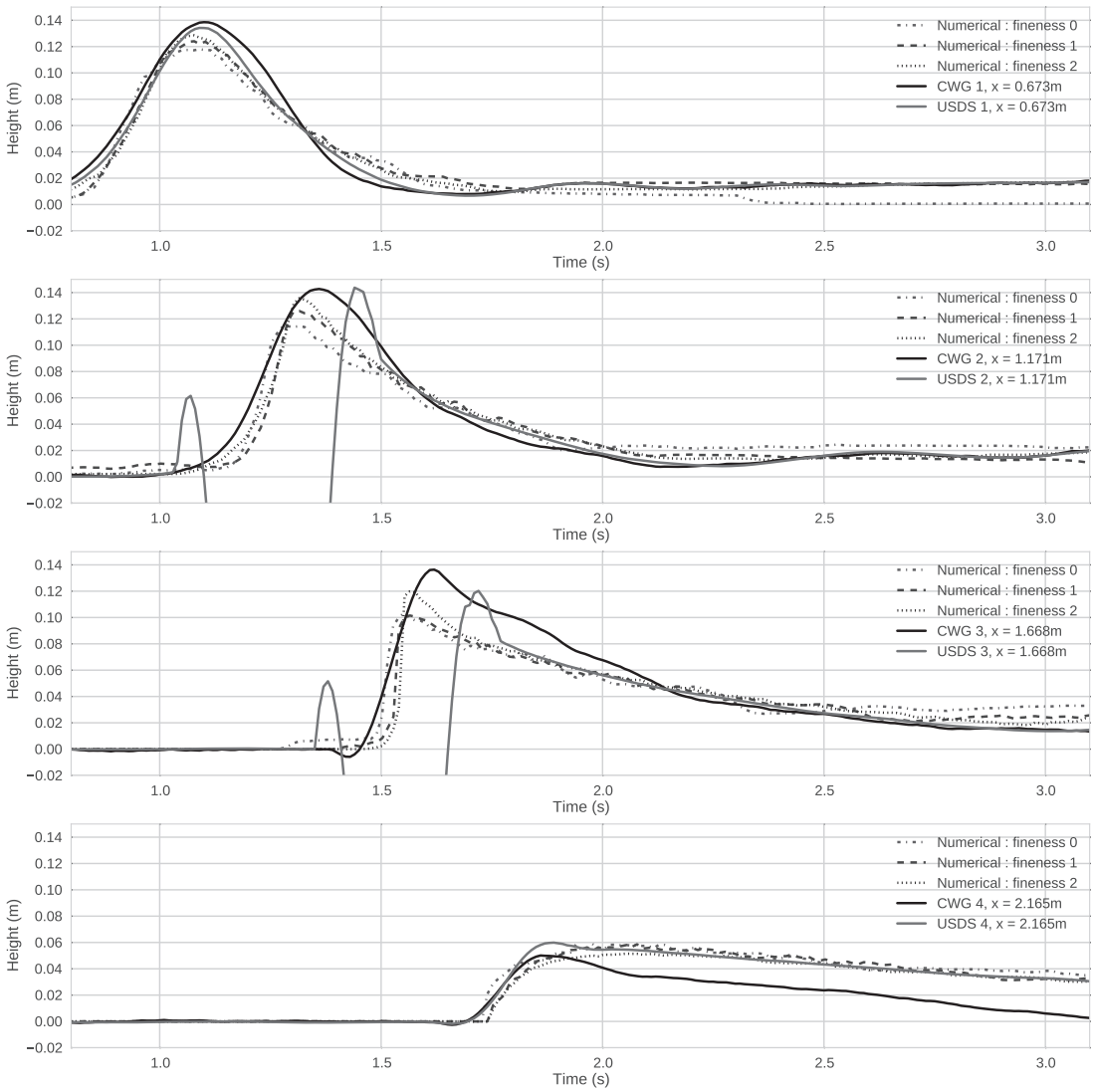
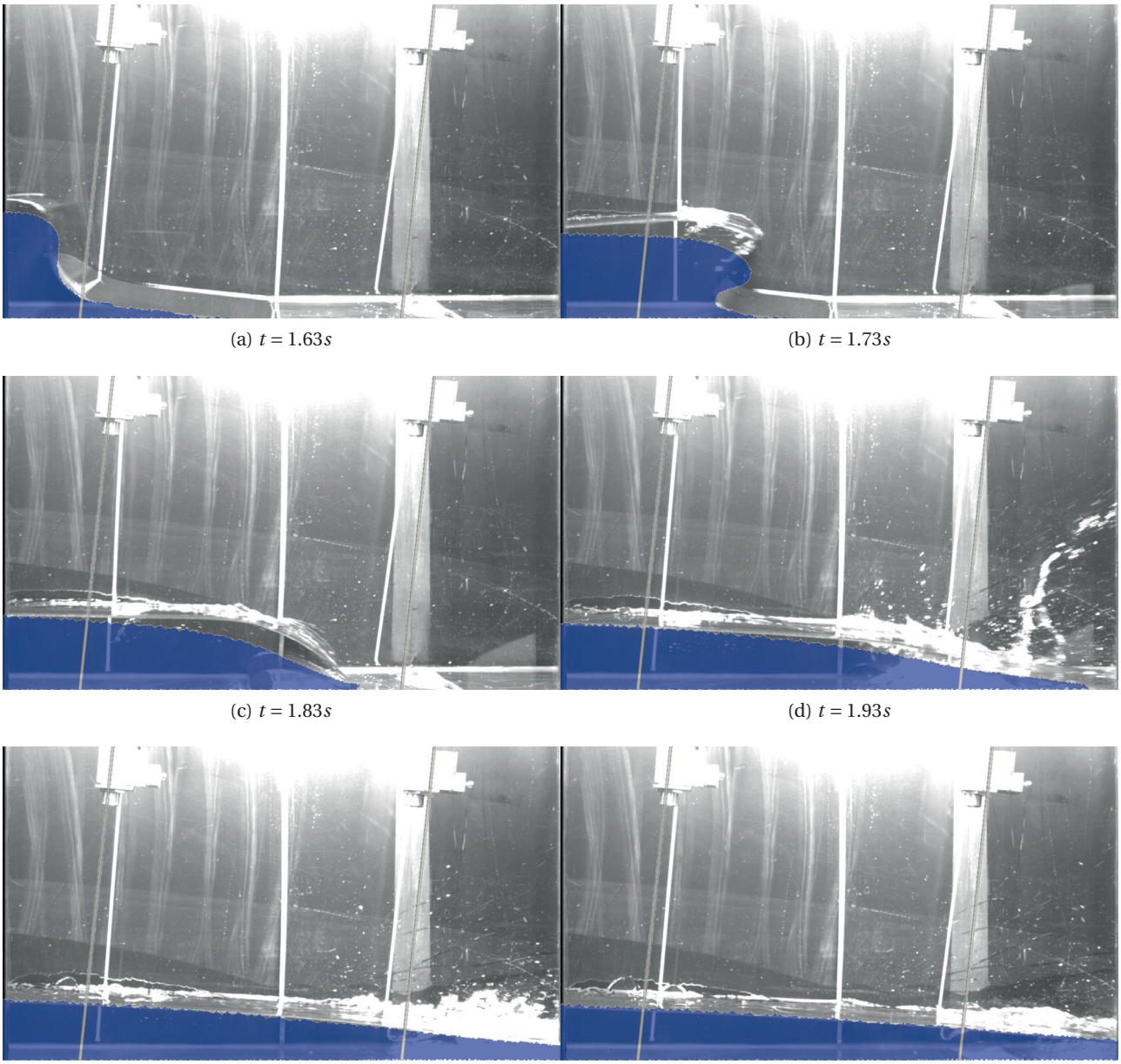


Figure 2.26: Pointwise height of the free surface throughout time on four different sensors for the wave test case for experimental parameters $d = 6°$ and $R_h = 0.7$.



(a) $t = 1.63s$ (b) $t = 1.73s$

(c) $t = 1.83s$ (d) $t = 1.93s$

(e) $t = 2.03s$ (f) $t = 2.13s$

Figure 2.27: Overlay of the numerical wave profile in blue on top of the high speed pictures of the experimental wave for parameters $d = 6°$ and $R_h = 0.7$.

2.2.3 Paddle-generated 3D wave in a large cavity

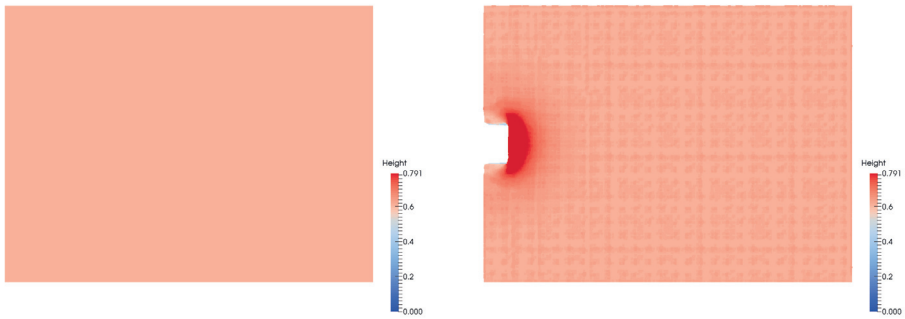
We extend the pseudo-2D paddle-generated water wave to a fully 3D wave. Despite having no measurements to compare with, with this we show that our current scheme is capable of simulating full 3D waves.

The cavity dimensions are $8 \times 6 \times 1.1$. The initial water height is 0.6. A paddle of width 1 moves at uniform velocity 1.52 from $t = 0$ to $t = 1$ and hence displaces a water volume of 0.912. Final time is $T = 6$.

Physical properties of the water for the Navier-Stokes simulations are taken as $\rho = 10^3 kg/m^3$ and $\mu = 10^{-3} kg/(m\ s)$. Simulation parameters for the octree mesh are $h_{min} = 1.5e-2$ and $l_{liquid} = l_{max} - 2$ to have coarse cells approximately of same size as tetrahedra. We have $\Delta t = 0.02$ and mesh size for the uniform tetrahedral mesh is respectively $H = 0.08$. No-slip conditions are imposed on the boundary of the tetrahedral mesh.

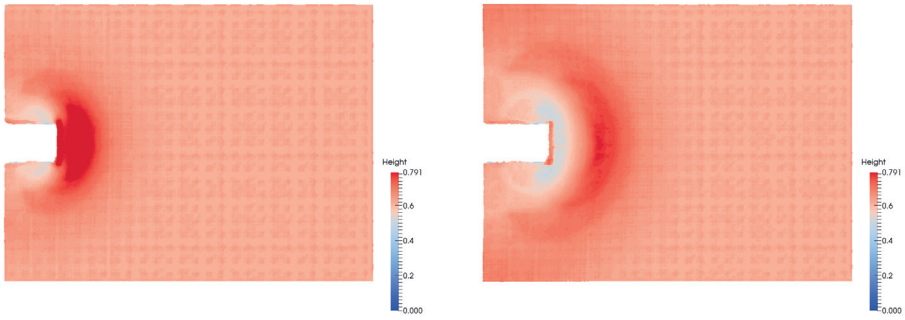
A bubble enrichment stabilization is used for the finite element problem along with a GMRES solver and ILU preconditioner. The paddle movement is simulated in practice by computing at each step which nodes are intersecting the region that has been swept by the paddle and imposing the paddle velocity at those points. The octree is refined to level l_{max} at the wall in contact with the paddle and at the water free surface but not on the other walls or cavity bottom which are aligned with the octree and hence captured exactly.

Wave heights are shown in Figure 2.28 and views of the octree mesh are shown in Figure 2.29 along with a cut of the tetrahedral mesh.



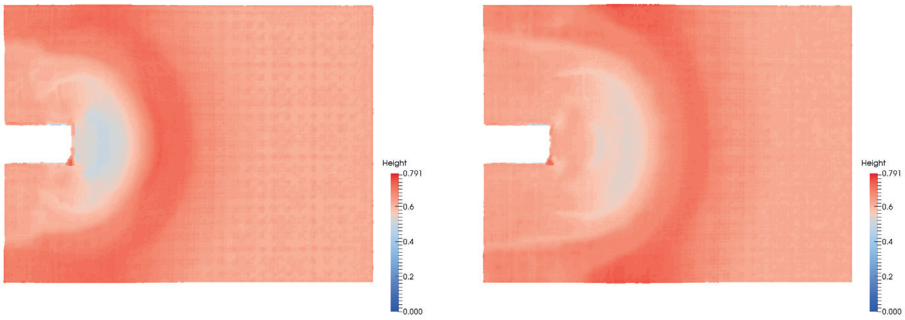
(a) $t = 0.0s$

(b) $t = 0.4s$



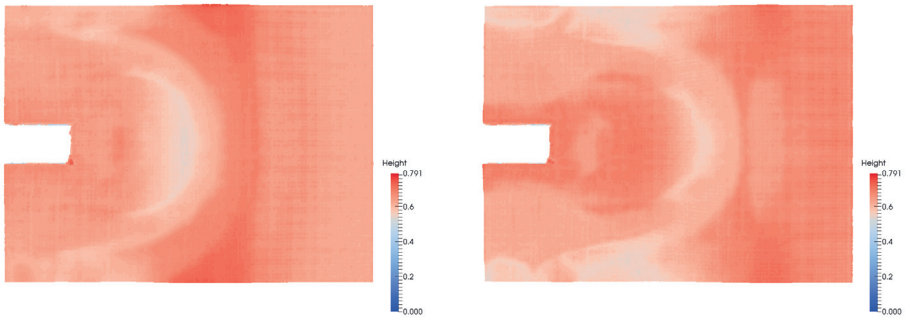
(c) $t = 0.8s$

(d) $t = 1.2s$



(e) $t = 1.6s$

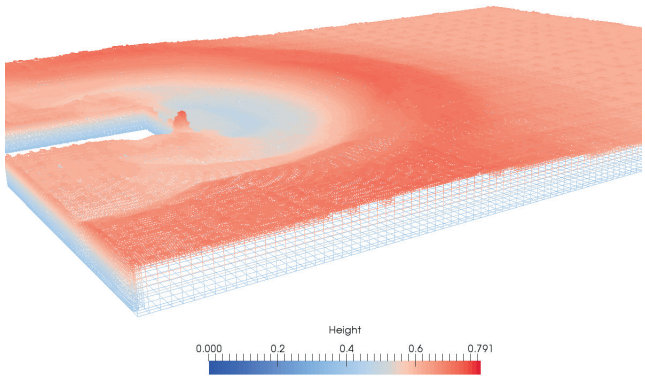
(f) $t = 2.0s$



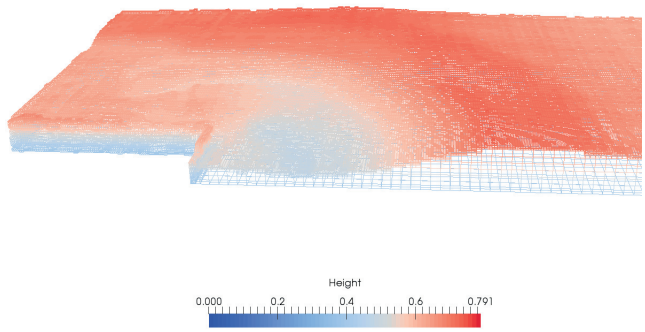
(g) $t = 2.4s$

(h) $t = 2.8s$

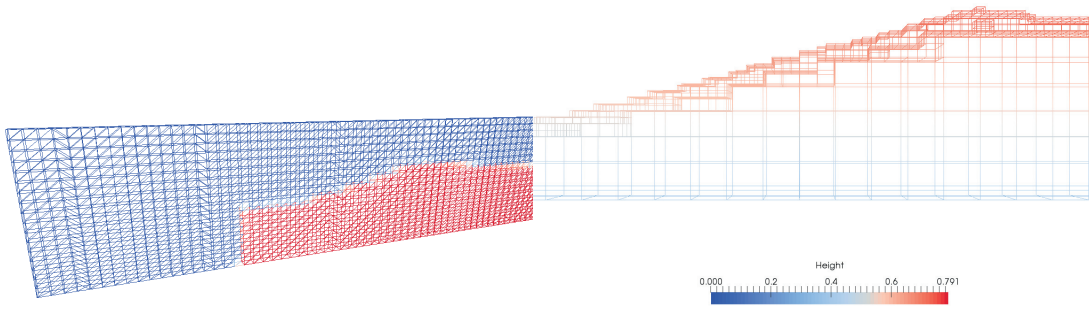
Figure 2.28: 3D wave from the top.



(a) Perspective view of the octree liquid cells at $t = 1.6$



(b) Slice of the octree liquid cells at $t = 1.6$



(c) Cut of the tetrahedral mesh

(d) Cut of the octree liquid cells at $t = 1.6$

Figure 2.29: 3D wave and octree mesh.

3 A study of first order stabilization schemes for the time-dependent Stokes problem

In this Chapter, we will present and compare different first order stabilization schemes for the time-dependent Stokes' equations. Stability and convergence results will be given for consistent and non-consistent PSPG schemes and numerical results will show accuracy and stability of the different schemes.

3.1 Definition of different stabilization schemes

Consider the time-dependent Stokes' equations on a space-time domain $\Omega \times [0, T]$ with velocity $\boldsymbol{u} : \Omega \times [0, T] \to \mathbb{R}^d$ and pressure $p : \Omega \times [0, T] \to \mathbb{R}$ with Ω a bounded open subset of $\mathbb{R}^d$, $d = 2, 3$ and $T > 0$. We first introduce the classical formulation and then provide the functional framework which we will use. Let $\boldsymbol{u}_0 : \Omega \times [0, T] \to \mathbb{R}^d$ be a given initial velocity and $\boldsymbol{f} : \Omega \times [0, T] \to \mathbb{R}^d$ a forcing term.

Assuming sufficient regularity, the classical Stokes' problem is to find velocity $\boldsymbol{u}$ and pressure p that satisfy

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} - \nu \Delta \boldsymbol{u} + \nabla p &= \boldsymbol{f} && \text{in } \Omega \times [0, T] \\ \nabla \cdot \boldsymbol{u} &= 0 && \text{in } \Omega \times [0, T] \\ \boldsymbol{u} &= 0 && \text{on } \partial\Omega \times [0, T] \\ \boldsymbol{u}(\cdot, 0) &= \boldsymbol{u}_0 && \text{in } \Omega. \end{aligned} \tag{3.1}$$

Let us define the spaces $V = [H_0^1(\Omega)]^d = \left\{ \boldsymbol{v} \in [H^1(\Omega)]^d, \boldsymbol{v} = \boldsymbol{0} \text{ on } \Omega \right\}$ and $Q = L_0^2(\Omega)$ to introduce the weak form of (3.1).

Let $\boldsymbol{f} \in L^2(0, T; [L^2(\Omega)]^d)$ and $\boldsymbol{u}_0 \in V$. Following [85], the weak formulation is therefore :

Find $\boldsymbol{u} \in L^2(0,T;[H_0^1(\Omega)]^d) \cap C^0([0,T],[L^2(\Omega)]^d)$ and p such that

$$\left(\frac{\partial \boldsymbol{u}}{\partial t}, \boldsymbol{v}\right) + \nu\,(\nabla \boldsymbol{u}, \nabla \boldsymbol{v}) - \left(p, \nabla\cdot \boldsymbol{v}\right) = \left(\boldsymbol{f}, \boldsymbol{v}\right) \quad \forall \boldsymbol{v} \in V \tag{3.2}$$

$$\left(\nabla\cdot \boldsymbol{u}, q\right) = 0 \qquad \forall q \in Q \tag{3.3}$$

where $\boldsymbol{u}_{|t=0} = \boldsymbol{u}_0$ and $(\cdot,\cdot)$ represents the usual $L^2(\Omega)$ inner product. We do not specify a space for the pressure but a lengthy discussion on it can be found in [86].

Let us now also introduce a conformal triangular or tetrahedral space discretization. For any $h > 0$, let $\mathcal{T}_h$ be a conformal regular mesh [87] of Ω in triangles or tetrahedra K having diameter $h_K \le h$. Let us define the piecewise linear finite element spaces

$$V_h = \left\{\boldsymbol{v}_h \in [\mathscr{C}^0(\overline{\Omega})]^d \mid \quad \boldsymbol{v}_{h|K} \in [\mathbb{P}_1(K)]^d \quad \forall K \in \mathcal{T}_h\right\} \cap [H_0^1(\Omega)]^d \tag{3.4}$$

$$Q_h = \left\{q_h \in \mathscr{C}^0(\overline{\Omega}) \mid \quad q_{h|K} \in \mathbb{P}_1(K) \quad \forall K \in \mathcal{T}_h\right\} \cap Q \tag{3.5}$$

and the space of bubble functions

$$V_{bub} = \left\{\boldsymbol{v}_h \in V \mid \quad \boldsymbol{v}_h = \sum_{K\in\mathcal{T}_h} \boldsymbol{v}_K^b \psi_b^K,\ \boldsymbol{v}_K^b \in \mathbb{R}^d\right\}$$

where we choose ψ_b^K to be the so called conforming bubbles defined as follows. Let ψ_i^K, $i = 1,\dots,d+1$ be the finite element basis functions linear on triangle/tetrahedron K. The conforming bubble ψ_b^K is defined as

$$\psi_b^K = (d+1)^{d+1} \prod_{i=1}^{d+1} \psi_i^K.$$

Using Green's formula, it can be shown that V_h and V_{bub} are orthogonal subspaces of $[H_0^1(\Omega)]^d$ for the H^1 seminorm and we will be using $(V_h \oplus V_{bub})$ as the velocity finite element space for the bubble stabilization.

In what follows, several finite element discretizations with continuous finite elements of first order are considered. Given $N > 0$, $\Delta t = T/N$, let $t_n = n\Delta t,\ n = 0,\dots,N$ be a time discretization of the interval $[0,T]$.

3.1.1 Bubble stabilization

Using continuous $\mathbb{P}_1$ finite elements enriched with bubble functions for velocity and continuous $\mathbb{P}_1$ finite elements for pressure with an implicit Euler time discretization, the weak problem

(3.2)-(3.3) becomes the following semi-discretized problem. Let $\boldsymbol{u}_h^n \simeq \boldsymbol{u}(t_n)$ and $\boldsymbol{u}_h^n$ is known. We take $\boldsymbol{u}_h^0 = \Pi_h^1 \boldsymbol{u}(t_0)$ where Π_h^1 is the L^2 projection onto V_h. $(\boldsymbol{u}_h^{n+1}, p_h^{n+1}) \in (V_h \oplus V_{bub}) \times Q_h$ is then computed from

$$\left(\frac{\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n}{\Delta t}, \boldsymbol{v}_h\right) + \nu\left(\nabla \boldsymbol{u}_h^{n+1}, \nabla \boldsymbol{v}_h\right) - \left(p_h^{n+1}, \nabla \cdot \boldsymbol{v}_h\right) = \left(\boldsymbol{f}(t^{n+1}), \boldsymbol{v}_h\right) \quad \forall \boldsymbol{v}_h \in V_h \oplus V_{bub} \tag{3.6}$$

$$\left(\nabla \cdot \boldsymbol{u}_h^{n+1}, q_h\right) = 0 \qquad \forall q_h \in Q_h \tag{3.7}$$

Taking $\boldsymbol{v}_h = \boldsymbol{u}_h^{n+1}$ and $q_h = p_h^{n+1}$ yields the stability estimate

$$\left\|\boldsymbol{u}_h^{n+1}\right\|_{L^2(\Omega)}^2 + \frac{\Delta t \nu}{2}\left\|\nabla \boldsymbol{u}_h^{n+1}\right\|_{L^2(\Omega)}^2 \leq \left\|\boldsymbol{u}_h^n\right\|_{L^2(\Omega)}^2 + C\Delta t \left\|\boldsymbol{f}(t^{n+1})\right\|_{L^2(\Omega)}^2.$$

The conforming bubble elements satisfy a Babuška-Brezzi or inf-sup condition [88] and we therefore have that : $\exists C,\ \forall h,\ \forall q_h \in Q_h$

$$\left\|q_h\right\|_{L^2(\Omega)} \leq C \sup_{\boldsymbol{v}_h \in (V_h \oplus V_{bub})} \frac{\left(q_h, \nabla \cdot \boldsymbol{v}_h\right)}{\|\nabla \boldsymbol{v}_h\|_{L^2(\Omega)}}. \tag{3.8}$$

For finite element approximations satisfying the inf-sup condition, error estimates for both velocity and pressure have been given for linearized Navier-Stokes equations in [89] with no constraint on the timestep and unconditional stability for the time-dependent Stokes equation is discussed in [90]. Despite being unconditionally stable with respect to the timestep, the scheme requires solving a larger linear system than PSPG-type schemes introduced in Section 3.1.2.

The bubble enrichment method has been linked to a residual-based PSPG stabilization in the context of $\mathbb{P}_1 - \mathbb{P}_1$ finite elements for the stationary Stokes equations in [49, 50]. In [50], both methods were shown to give close results in the sense that the norm of the difference of the velocity-pressure pairs is at most of order h^2 where h is the mesh size. In Section 3.1.5, we discuss this link in the context of the time-dependent Stokes equations.

3.1.2 PSPG stabilizations

Since $\mathbb{P}_1 - \mathbb{P}_1$ finite elements are known not to satisfy the inf-sup condition [46], we introduce several possible stabilizations that stem from a general class of stabilizations known as PSPG (Pressure Stabilized Petrov-Galerkin) stabilizations that circumvent this condition. Analogs of these different stabilizations have been widely studied in the stationary case, see for instance [46] and references therein, but several candidates are available for an extension to the time-dependent case and some of them are presented here.

The parameters $\beta \in \{0,1\}$ and $\gamma \in \left\{0, \pm\frac{1}{\Delta t}\right\}$ define different types of stabilization. Using $\mathbb{P}_1 - \mathbb{P}_1$ finite elements and a consistent stabilization with an implicit Euler time discretization, the weak problem (3.2)-(3.3) becomes the following fully discretized problem. Let $\boldsymbol{u}_h^n \simeq \boldsymbol{u}(t_n)$ and $\boldsymbol{u}_h^n$ is known. We take $\boldsymbol{u}_h^0 = \Pi_h^1 \boldsymbol{u}(t_0)$ where Π_h^1 is the L^2 projection onto V_h. $(\boldsymbol{u}_h^{n+1}, p_h^{n+1})$ is then computed from

$$\begin{aligned}
&\left(\frac{\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n}{\Delta t}, \boldsymbol{v}_h\right) + \nu\left(\nabla \boldsymbol{u}_h^{n+1}, \nabla \boldsymbol{v}_h\right) - \left(p_h^{n+1}, \nabla \cdot \boldsymbol{v}_h\right) + \left(\nabla \cdot \boldsymbol{u}_h^{n+1}, q_h\right) \\
&+ \sum_{K \in \mathcal{T}_h} \alpha_K \left(\beta \frac{\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n}{\Delta t} + \nabla p_h^{n+1} - \boldsymbol{f}(t^{n+1}), \nabla q_h + \gamma \boldsymbol{v}_h\right)_K = \left(\boldsymbol{f}(t^{n+1}), \boldsymbol{v}_h\right) \quad \forall \boldsymbol{v}_h, q_h \in V_h, Q_h
\end{aligned} \tag{3.9}$$

for $n \geq 0$ where $\alpha_K > 0 \; \forall K \in \mathcal{T}_h$ are stabilization parameters associated with element K whose choice will be discussed in Section 3.1.6. Note that we omit the $-\Delta \boldsymbol{u}_h^{n+1}$ term because it is zero for our first order elements. The parameter β selects either a consistent in time or non-consistent stabilization and γ is a parameter which sets the scaling between the stabilization residuals.

Choosing $\gamma = 0$ for $\beta = 0, 1$ yields an analog for the time-dependent Stokes equations of the Streamline Upwind Galerkin (SUPG) stabilization [47] with a respectively non-consistent and consistent in time formulation for $\beta = 0, 1$. The choice $\beta = 0,\ \gamma = 0$ is also reminiscent of the Brezzi-Pitkäranta stabilization [48] in the stationary case, which leaves only a pressure gradient term in the stabilization of the continuity equation but in our case a force term is present as well. When $\gamma \neq 0$, a stabilizing contribution is also added to the momentum equation which can be used to obtain a symmetric linear system as in the case of the Galerkin Least Squares (GLS) stabilization [46].

For the stationary case, a comprehensive comparison of the different variations of PSPG stabilizations has been done in [91], where it is shown that while the solutions for different stabilizations vary little, the matrix properties and performance of different Krylov subspace solvers change significantly.

Scalings $\gamma = \pm\frac{1}{\Delta t}$ will be considered. They correspond to symmetric and anti-symmetric matrix formulations of the stabilization.

Analyses of these PSPG schemes for the time-dependent Stokes equations have been performed individually, most of them recently, in [92, 51, 90, 56, 57, 58]. As far as we know, a stability estimate of the velocity for the $\beta = 1,\ \gamma = 0$ and $\beta = 0,\ \gamma = 0$ stabilizations and a stability condition for the $\beta = 1,\ \gamma = 0$ stabilization were first derived in [92]. In [51], a parallel is made between bubble enrichment of the velocity finite element space and the $\beta = 1,\ \gamma = \frac{1}{\Delta t}$ stabilization. Indeed, assuming that the bubble term behaves like a residual term of the equation, equivalence is proven between the bubble enrichment and the $\beta = 1,\ \gamma = -\frac{1}{\Delta t}$

stabilization. A stability proof for the velocity is then given along with some stability result weaker than L^2 stability for the pressure gradient.

In [56], a proof of stability and convergence of a general class of symmetric stabilizations including the Brezzi-Pitkäranta stabilization which resembles the $\beta = 0,\ \gamma = 0$ stabilization but lacking the force term is given in the fully discrete case. We show in Section 3.1.5 that the scheme with the force term can be derived from the bubble enrichment scheme by making two simplifying assumptions. In [57], stability and convergence for the velocity in the fully discrete case has been proven for the $\beta = 1,\ \gamma = 0$ stabilization. Stability of the pressure is also proven and the L^2 convergence is mentioned as a corollary although without proof. In [58], a semi-discrete analysis proves stability and convergence for both velocity and pressure in the L^2 norm and for the velocity in the fully discrete case.

In Section 3.2 we provide a unified proof of the L^2-stability of the velocity and pressure for both $\beta = 0,\ \gamma = 0$ and $\beta = 1,\ \gamma = 0$ stabilizations in the fully discrete case while keeping track of the influence of the viscosity and stabilization parameter. In fact, we give a result based on the $\beta = 1,\ \gamma = 0$ stabilization expressing the effect of any consistency default on the norm of the solution. A proof for L^2 convergence of both velocity and pressure in the fully discrete case is then given in Section 3.3 while again keeping track of the influence of viscosity and stabilization parameters on the error for further analysis. The convergence result then gives the optimal choice of the stabilization parameter.

3.1.3 Orthogonal Subscales stabilization

Codina and Blasco [52, 53] have introduced the following residual-based stabilization sometimes referred to as Orthogonal Subscales Stabilization (OSS). Let $\boldsymbol{u}_h^n \simeq \boldsymbol{u}(t_n)$ and $\boldsymbol{u}_h^n$ is known. We take $\boldsymbol{u}_h^0 = \Pi_h^1 \boldsymbol{u}(t_0)$ where Π_h^1 is the L^2 projection onto V_h. $(\boldsymbol{u}_h^{n+1}, p_h^{n+1})$ is then computed from

$$\left(\frac{\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n}{\Delta t}, \boldsymbol{v}_h\right) + \nu\left(\nabla \boldsymbol{u}_h^{n+1}, \nabla \boldsymbol{v}_h\right) - \left(p_h^{n+1}, \nabla \cdot \boldsymbol{v}_h\right) = \left(\boldsymbol{f}(t^{n+1}), \boldsymbol{v}_h\right) \qquad \forall \boldsymbol{v}_h \in V_h \tag{3.10}$$

$$\left(\nabla \cdot \boldsymbol{u}_h^{n+1}, q_h\right) + \sum_{K \in \mathcal{T}_h} \alpha_K \left(\nabla p_h^{n+1} - \Pi_h^1(\nabla p_h^{n+1}), \nabla q_h - \Pi_h^1(\nabla q_h)\right)_K = 0 \qquad \forall q_h \in Q_h \tag{3.11}$$

for $n \geq 0$ where Π_h^1 is the L^2 projector onto V_h. This scheme has been proven to be equivalent to a PSPG stabilization with regularized Laplacian by Burman and Fernandez in [57]. The fact that the projector is non-local is however not very convenient and a scheme using local pressure gradient projections has been proposed in [93].

3.1.4 Local pressure projection stabilization

In [54, 55] a new stabilization was introduced based on polynomial pressure projections for the stationary Stokes equations. The same scheme has been used for time-dependent Stokes

problems and we introduce the stabilization used for continuous $\mathbb{P}_1 - \mathbb{P}_1$ finite elements. Let $\boldsymbol{u}_h^n \simeq \boldsymbol{u}(t_n)$ and $\boldsymbol{u}_h^n$ is known. We take $\boldsymbol{u}_h^0 = \Pi_h^1 \boldsymbol{u}(t_0)$ where Π_h^1 is the L^2 projection onto V_h. $(\boldsymbol{u}_h^{n+1}, p_h^{n+1})$ is then computed from

$$\left(\frac{\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n}{\Delta t}, \boldsymbol{v}_h\right) + \nu\left(\nabla \boldsymbol{u}_h^{n+1}, \nabla \boldsymbol{v}_h\right) - \left(p_h^{n+1}, \nabla \cdot \boldsymbol{v}_h\right) = \left(\boldsymbol{f}(t^{n+1}), \boldsymbol{v}_h\right) \qquad \forall \boldsymbol{v}_h \in V_h \tag{3.12}$$

$$\left(\nabla \cdot \boldsymbol{u}_h^{n+1}, q_h\right) + \bar{\alpha} \sum_{K \in \mathcal{T}_h} \left(p_h^{n+1} - \Pi_h^0 p_h^{n+1}, q_h - \Pi_h^0 q_h\right)_K = 0 \qquad \forall q_h \in Q_h \tag{3.13}$$

for $n \geq 0$ where Π_h^0 is a local piecewise constant projector that can be taken as the elementwise mean and $\bar{\alpha} > 0$ a constant stabilization parameter. Stability and convergence of both velocity and pressure have been proved in [94] for the Navier-Stokes equations with $\bar{\alpha} = 1$ and $\mathbb{P}_1 - \mathbb{P}_1$ elements and in [95] for the time-dependent Stokes equations with $\mathbb{P}_1 - \mathbb{P}_1$ elements. Although the natural scaling in the sense that physical units are respected suggests a $\frac{1}{\nu}$ scaling factor in the stabilization term as mentioned in a footnote in [55], stability and convergence results in [94, 95] omit the scaling and prove optimal convergence rate without making the dependence on viscosity explicit. Although the stabilization parameter can be set to $\bar{\alpha} = 1$ for Stokes' equations, solving the full Navier-Stokes equations with a time-splitting method potentially requires different values of the stabilization parameter as seen in [96].

3.1.5 Link between bubble and PSPG stabilizations

The link between the bubble enrichment method and PSPG stabilizations has been reported in [49, 50] in the context of $2D$ stationary Stokes equations. It has been shown in [50] that the bubble unknowns can be eliminated from the linear system after discretization and a formulation similar to PSPG stabilizations can be recovered. This method is sometimes referred to as static condensation. In the context of time-dependent Stokes equations, the bubble unknowns cannot be strictly eliminated but we discuss here some alternative ways of recovering a PSPG-type method. This recovery allows us to explicit terms that are responsible for the unconditional stability of the bubble-enriched numerical scheme which are missing in the PSPG methods.

A method equivalent to the bubble-enrichment method can then be developed by eliminating the bubble unknown and then reconstructing it locally. This method is however cheaper since the linear system is of the same size as PSPG methods. A mention of this method has been noted in [97].

Let us decompose the bubble enriched velocity at timestep n into its piecewise linear part $\boldsymbol{u}_{h,l}^n$ and bubble part $\boldsymbol{u}_{h,b}^n$ such that $\boldsymbol{u}_h^n = \boldsymbol{u}_{h,l}^n + \boldsymbol{u}_{h,b}^n$ with $\boldsymbol{u}_{h,l}^n \in V_h$ and $\boldsymbol{u}_{h,b}^n \in V_{bub}$. We will eliminate $\boldsymbol{u}_{h,b}^{n+1}$ at the elemental level from (3.6)-(3.7) similarly to [50]. We adapt the computations in [50] first of all for a $3D$ setting and also for the time-dependent case.

The 19×19 matrix M_K containing the contributions of an element $K \in \mathcal{T}_h$ in the variational formulation (3.6)-(3.7) is defined by blocks as follows.

$$M_K = |K| \begin{pmatrix} A_K & e_K & 0 & 0 & 0 & 0 & B_{K,x}^T \\ e_K^T & \lambda_K & 0 & 0 & 0 & 0 & w_{K,x}^T \\ 0 & 0 & A_K & e_K & 0 & 0 & B_{K,y}^T \\ 0 & 0 & e_K^T & \lambda_K & 0 & 0 & w_{K,y}^T \\ 0 & 0 & 0 & 0 & A_K & e_K & B_{K,z}^T \\ 0 & 0 & 0 & 0 & e_K^T & \lambda_K & w_{K,z}^T \\ B_{K,x} & w_{K,x} & B_{K,y} & w_{K,y} & B_{K,z} & w_{K,z} & 0 \end{pmatrix}$$

$|K|$ is the volume of tetrahedron K. Exact expressions for each block will be explicited after a summary of where each block comes from is given. The 4×4 A_K matrices stem from contributions from products between velocity linear basis functions, e_K and e_K^T from products between velocity linear and bubble basis functions. λ_K are contributions coming from products of velocity bubble functions. $B_{K,x}$, $B_{K,y}$, $B_{K,z}$ and $B_{K,x}^T$, $B_{K,y}^T$, $B_{K,z}^T$ come from products between divergence of the velocity linear basis functions and pressure (linear) basis functions. $w_{K,x}$, $w_{K,y}$, $w_{K,z}$ and $w_{K,x}^T$, $w_{K,y}^T$, $w_{K,z}^T$ come from products between divergence of the velocity bubble basis functions and pressure (linear) basis functions.

Let us denote ψ_i^K, $i = 1, ..., 4$ the linear nodal basis functions on K and ψ_b^K the bubble basis function on K. The block elements of the matrix M_K are defined as follows.

$$|K|\,(A_K)_{i,j} = \frac{1}{\Delta t}\int_K \psi_i^K \psi_j^K + \nu \int_K \nabla\psi_i^K \cdot \nabla\psi_j^K \qquad i, j = 1, ..., 4 \tag{3.14}$$

$$|K|\left(B_{K,x}\right)_{i,j} = -\int_K \psi_i^K\, \partial_x \psi_j^K, \quad |K|\left(B_{K,y}\right)_{i,j} = -\int_K \psi_i^K\, \partial_y \psi_j^K,$$

$$|K|\left(B_{K,z}\right)_{i,j} = -\int_K \psi_i^K\, \partial_z \psi_j^K \qquad i, j = 1, ..., 4 \tag{3.15}$$

$$|K|\left(w_{K,x}\right)_i = \int_K \psi_b^K\, \partial_x \psi_i^K, \quad |K|\left(w_{K,y}\right)_i = \int_K \psi_b^K\, \partial_y \psi_i^K,$$

$$|K|\left(w_{K,z}\right)_i = \int_K \psi_b^K\, \partial_z \psi_i^K \qquad i = 1, ..., 4 \tag{3.16}$$

$$|K|\,(e_K)_i = \frac{1}{\Delta t}\int_K \psi_i^K\, \psi_b^K \qquad i = 1, ..., 4 \tag{3.17}$$

$$|K|\lambda_K = \frac{1}{\Delta t}\int_K \psi_b^K \psi_b^K + \nu \int_K \nabla\psi_b^K \cdot \nabla\psi_b^K \tag{3.18}$$

Note that we have used Green's formula and the fact that $\psi_{b|\partial K}^K = 0$ to show that $\int_K \nabla\psi_i^K \cdot \nabla\psi_b^K = 0$, $i = 1, ..., 4$. The 19−vector containing the contributions of an element $K \in \mathcal{T}_h$ in the right-

hand side of the variational formulation (3.6)-(3.7) is defined as

$$(r_K)_i = \begin{cases} \int_K f_1\, \psi_i^K + \frac{1}{\Delta t}\sum_{j=1}^{4} u^n_{x,K,j}\int_K \psi_j^K\psi_i^K + \frac{1}{\Delta t}u^n_{x,K,b}\int_K \psi_b^K\psi_i^K\,, & i=1,\dots,4 \\ \int_K f_1\, \psi_b^K + \frac{1}{\Delta t}\sum_{j=1}^{4} u^n_{x,K,j}\int_K \psi_j^K\psi_b^K + \frac{1}{\Delta t}u^n_{x,K,b}\int_K \psi_b^K\psi_b^K\,, & i=5, \\ \int_K f_2\, \psi_{i-5}^K + \frac{1}{\Delta t}\sum_{j=1}^{4} u^n_{y,K,j}\int_K \psi_j^K\psi_{i-5}^K + \frac{1}{\Delta t}u^n_{y,K,b}\int_K \psi_b^K\psi_{i-5}^K\,, & i=6,\dots,9 \\ \int_K f_2\, \psi_b^K + \frac{1}{\Delta t}\sum_{j=1}^{4} u^n_{y,K,j}\int_K \psi_j^K\psi_b^K + \frac{1}{\Delta t}u^n_{y,K,b}\int_K \psi_b^K\psi_b^K\,, & i=10, \\ \int_K f_3\, \psi_{i-10}^K + \frac{1}{\Delta t}\sum_{j=1}^{4} u^n_{z,K,j}\int_K \psi_j^K\psi_{i-10}^K + \frac{1}{\Delta t}u^n_{z,K,b}\int_K \psi_b^K\psi_{i-10}^K\,, & i=11,\dots,14 \\ \int_K f_3\, \psi_b^K + \frac{1}{\Delta t}\sum_{j=1}^{4} u^n_{z,K,j}\int_K \psi_j^K\psi_b^K + \frac{1}{\Delta t}u^n_{z,K,b}\int_K \psi_b^K\psi_b^K\,, & i=15, \\ 0, & i=16,\dots,19. \end{cases}$$

where $f_i,\ i=1,2,3$ are the components of $\boldsymbol{f}(t_{n+1})$. $u^n_{x,K,j}$, $u^n_{y,K,j}$ and $u^n_{z,K,j}$ are respectively the value of the first, second and third component of $\boldsymbol{u}_h^n$ at node j of tetrahedron K. $u^n_{x,K,b}$, $u^n_{y,K,b}$ and $u^n_{z,K,b}$ are respectively the bubble value of the first, second and third component of $\boldsymbol{u}_h^n$ in tetrahedron K.

Keeping in mind that derivatives of the linear basis functions yield constants on the tetrahedron K, and computing integrals, we get

$$(A_K)_{i,j} = \frac{1}{\Delta t}\frac{1}{20}(1+\delta_{ij}) + \nu\nabla\psi_i^K\cdot\nabla\psi_j^K \qquad i,j=1,\dots,4 \tag{3.19}$$

$$\begin{aligned} \left(B_{K,x}\right)_{i,j} &= -\frac{1}{4}\partial_x\psi_j^K, \quad \left(B_{K,y}\right)_{i,j} = -\frac{1}{4}\partial_y\psi_j^K, \\ \left(B_{K,z}\right)_{i,j} &= -\frac{1}{4}\partial_z\psi_j^K \qquad i,j=1,\dots,4 \end{aligned} \tag{3.20}$$

$$\begin{aligned} \left(w_{K,x}\right)_i &= \frac{32}{105}\partial_x\psi_i^K, \quad \left(w_{K,y}\right)_i = \frac{32}{105}\partial_y\psi_i^K, \\ \left(w_{K,z}\right)_i &= \frac{32}{105}\partial_z\psi_i^K \qquad i=1,\dots,4 \end{aligned} \tag{3.21}$$

$$(e_K)_i = \frac{1}{\Delta t}\frac{8}{105} \qquad i=1,\dots,4 \tag{3.22}$$

$$\lambda_K = \frac{1}{\Delta t}\frac{8192}{51975} + \frac{\nu}{|K|}\int_K \nabla\psi_b^K\cdot\nabla\psi_b^K \tag{3.23}$$

where δ_{ij} is the Kronecker delta. Note that unlike in [50] we use conforming bubbles, which are designed such that their maximum is 1. Our bubble basis function contains then a multiplicative factor of 256 with respect to the basis bubble taken simply as the product of

linear basis functions. We also have

$$
(r_K)_i = \begin{cases}
\int_K f_1\, \psi_i^K + \frac{|K|}{\Delta t}\frac{1}{20}\sum_{j=1}^{4}(1+\delta_{ij})u_{x,K,j}^n + \frac{|K|}{\Delta t}\frac{8}{105}u_{x,K,b}^n\,, & i = 1,\dots,4 \\
\int_K f_1\, \psi_b^K + \frac{|K|}{\Delta t}\frac{8}{105}\sum_{j=1}^{4} u_{x,K,j}^n + \frac{|K|}{\Delta t}\frac{8192}{51975}u_{x,K,b}^n\,, & i = 5, \\
\int_K f_2\, \psi_{i-5}^K + \frac{|K|}{\Delta t}\frac{1}{20}\sum_{j=1}^{4}(1+\delta_{ij})u_{y,K,j}^n + \frac{|K|}{\Delta t}\frac{8}{105}u_{y,K,b}^n\,, & i = 6,\dots,9 \\
\int_K f_2\, \psi_b^K + \frac{|K|}{\Delta t}\frac{8}{105}\sum_{j=1}^{4} u_{y,K,j}^n + \frac{|K|}{\Delta t}\frac{8192}{51975}u_{y,K,b}^n\,, & i = 10, \\
\int_K f_3\, \psi_{i-10}^K + \frac{|K|}{\Delta t}\frac{1}{20}\sum_{j=1}^{4}(1+\delta_{ij})u_{z,K,j}^n + \frac{|K|}{\Delta t}\frac{8}{105}u_{z,K,b}^n\,, & i = 11,\dots,14 \\
\int_K f_3\, \psi_b^K + \frac{|K|}{\Delta t}\frac{8}{105}\sum_{j=1}^{4} u_{z,K,j}^n + \frac{|K|}{\Delta t}\frac{8192}{51975}u_{z,K,b}^n\,, & i = 15, \\
0, & i = 16,\dots,19.
\end{cases}
$$

It is important to note that the bubble basis function of tetrahedron K yields no contributions to any other tetrahedra, which allows to work on an elemental level and eliminate the bubble unknown in the matrix M_K. We perform the following line substitutions on the elemental matrix M_K and right-hand side vector r_K

$$
\begin{aligned}
l_{1-4} &\mapsto l_{1-4} - \frac{1}{\lambda_K} l_5 \\
l_{6-9} &\mapsto l_{6-9} - \frac{1}{\lambda_K} l_{10} \\
l_{11-14} &\mapsto l_{11-14} - \frac{1}{\lambda_K} l_{15} \\
l_{16-19} &\mapsto l_{16-19} - \frac{1}{\lambda_K}\left(w_{K,x} l_5 + w_{K,y} l_{10} + w_{K,z} l_{15}\right)
\end{aligned}
$$

where l_i denotes line i of the matrix M_K and right-hand side vector r_K. This yields the 16×16 matrix M'_K

$$
M'_K = |K| \begin{pmatrix} A'_K & 0 & 0 & {B'_{K,x}}^T \\ 0 & A'_K & 0 & {B'_{K,y}}^T \\ 0 & 0 & A'_K & {B'_{K,z}}^T \\ B'_{K,x} & B'_{K,y} & B'_{K,z} & -S_K \end{pmatrix}
$$

with block matrices defined as follow

$$A'_K = A_K - \frac{1}{\lambda_K} e_K e_K^T \tag{3.24}$$

$$B'_{K,x} = B_{K,x} - \frac{1}{\lambda_K} w_{K,x} e_K^T \tag{3.25}$$

$$B'_{K,y} = B_{K,y} - \frac{1}{\lambda_K} w_{K,y} e_K^T \tag{3.26}$$

$$B'_{K,z} = B_{K,z} - \frac{1}{\lambda_K} w_{K,z} e_K^T \tag{3.27}$$

$$S_K = \frac{1}{\lambda_K} \left(w_{K,x} w_{K,x}^T + w_{K,y} w_{K,y}^T + w_{K,z} w_{K,z}^T \right). \tag{3.28}$$

The substitutions also yield the right-hand side 16−vector r'_K defined as

$$\left(r'_K\right)_i = \begin{cases} (r_K)_i - \dfrac{1}{\lambda_K \Delta t} \dfrac{8}{105} (r_K)_5 \ , & i = 1, \dots, 4 \\ (r_K)_{i+1} - \dfrac{1}{\lambda_K \Delta t} \dfrac{8}{105} (r_K)_{10} \ , & i = 5, \dots, 8 \\ (r_K)_{i+2} - \dfrac{1}{\lambda_K \Delta t} \dfrac{8}{105} (r_K)_{15} \ , & i = 9, \dots, 12 \\ -\dfrac{32}{105} \dfrac{1}{\lambda_K} \left(\left(\boldsymbol{f}(t_{n+1}), \psi_b^K \nabla \psi_{i-12}^K \right)_K + \dfrac{1}{\Delta t} \left(\boldsymbol{u}_h^n, \psi_b^K \nabla \psi_{i-12}^K \right)_K \right), & i = 13, \dots, 16. \end{cases}$$

We will now rewrite the substituted system as a variational weak formulation. Remarking as in [50] that the term

$$\frac{1}{\lambda_K} e_K e_K^T = \frac{1}{\lambda_K} \left(\frac{32}{105} \right)^2 \frac{1}{16} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

in (3.24) corresponds to the tetrahedron's barycenter rule, which is exact for linear integrands, the corresponding variational term is $\left(\boldsymbol{u}_{h,l}^{n+1}, \overline{\boldsymbol{v}_h} \right)_K$ for $\boldsymbol{v}_h \in V_h$ where $\overline{\boldsymbol{v}_h}$ is the average value of $\boldsymbol{v}_h$ on the tetrahedron K, here $\frac{1}{4}$. The A'_K matrix blocks then give rise to the following term in the variational formulation

$$-\frac{1}{\lambda_K} \left(\frac{32}{105} \right)^2 \frac{1}{\Delta t^2} \left(\boldsymbol{u}_{h,l}^{n+1}, \overline{\boldsymbol{v}_h} \right).$$

Keeping in mind that $\frac{1}{|K|} \int_K \psi_i^K = \frac{1}{4}$, $i = 1, \dots, 4$, the blocks ${B'_{K,x}}^T$, ${B'_{K,y}}^T$ and ${B'_{K,z}}^T$ in the matrix contribute the following term to the variational form

$$-\frac{1}{\lambda_K} \left(\frac{32}{105} \right)^2 \left(\nabla p_h^{n+1}, \frac{1}{\Delta t} \boldsymbol{v}_h \right).$$

Finally, the matrices $B_{K,x}$, $B_{K,y}$, $B_{K,z}$ and $-S_K$ in M'_K give rise to the following terms

$$+\frac{1}{\lambda_K} \left(\frac{32}{105} \right)^2 \left(\frac{1}{\Delta t} \boldsymbol{u}_{h,l}^{n+1}, \nabla q_h \right) + \frac{1}{\lambda_K} \left(\frac{32}{105} \right)^2 \left(\nabla p_h^{n+1}, \nabla q_h \right).$$

Note that the plus signs are due to the standard choice in the context of Navier-Stokes equations of writing a symmetric matrix formulation and negating the block when writing the variational formulation. To write the right-hand side r'_K as a variational formulation, we note that for all $i = 1, ..., 4$, we have

$$\int_K \psi_i^K = \frac{|K|}{4}$$

and hence $\overline{\psi_i^K} = \frac{1}{4}$. It follows that the integrals of the type

$$\int_K \psi_j^K \psi_b^K$$

can be written as

$$4 \int_K \psi_j^K \psi_b^K \overline{\psi_i^K}.$$

Using this trick, the terms appearing in r'_K but not in r_K can be written in variational formulation as

$$-\frac{1}{\lambda_K} \frac{32}{105} \left(\boldsymbol{f}(t_{n+1}) + \frac{1}{\Delta t} \left(\boldsymbol{u}_{h,l}^n + \boldsymbol{u}_{h,b}^n \right), \psi_b^K \left(\frac{\overline{\boldsymbol{v}_h}}{\Delta t} - \nabla q_h \right) \right).$$

Combining all the terms, the final variational formulation we obtain is

$$\begin{aligned}
&\left(\frac{\boldsymbol{u}_{h,l}^{n+1} - \boldsymbol{u}_{h,l}^n}{\Delta t}, \boldsymbol{v}_h \right) + \nu \left(\nabla \boldsymbol{u}_{h,l}^{n+1}, \nabla \boldsymbol{v}_h \right) - \left(p_h^{n+1}, \nabla \cdot \boldsymbol{v}_h \right) + \left(\nabla \cdot \boldsymbol{u}_{h,l}^{n+1}, q_h \right) \\
&- \sum_{K \in \mathcal{T}_h} \frac{1}{\lambda_K} \left(\frac{32}{105} \right)^2 \left(\frac{\boldsymbol{u}_{h,l}^{n+1}}{\Delta t} + \nabla p_h^{n+1}, \frac{1}{\Delta t} \boldsymbol{v}_h - \nabla q_h \right)_K + \sum_{K \in \mathcal{T}_h} \frac{1}{\lambda_K} \left(\frac{32}{105} \right)^2 \left(\frac{\boldsymbol{u}_{h,l}^{n+1}}{\Delta t}, \frac{\boldsymbol{v}_h - \overline{\boldsymbol{v}_h}}{\Delta t} \right)_K \\
&= \left(\boldsymbol{f}(t^{n+1}) + \frac{1}{\Delta t} \boldsymbol{u}_{h,b}^n, \boldsymbol{v}_h \right) - \sum_{K \in \mathcal{T}_h} \frac{1}{\lambda_K} \frac{32}{105} \left(\boldsymbol{f}(t^{n+1}) + \frac{\boldsymbol{u}_{h,l}^n + \boldsymbol{u}_{h,b}^n}{\Delta t}, \psi_b^K \left(\frac{\overline{\boldsymbol{v}_h}}{\Delta t} - \nabla q_h \right) \right)_K \\
&\forall \boldsymbol{v}_h \in V_h, \forall q_h \in Q_h. \qquad (3.29)
\end{aligned}$$

It is then possible to write the formulation (3.29) as a PSPG formulation with some additional

terms. Writing the PSPG formulation and grouping the remaining terms yields

$$
\begin{aligned}
&\left(\frac{\boldsymbol{u}_{h,l}^{n+1}-\boldsymbol{u}_{h,l}^{n}}{\Delta t},\boldsymbol{v}_h\right)+\nu\left(\nabla\boldsymbol{u}_{h,l}^{n+1},\nabla\boldsymbol{v}_h\right)-\left(p_h^{n+1},\nabla\cdot\boldsymbol{v}_h\right)+\left(\nabla\cdot\boldsymbol{u}_{h,l}^{n+1},q_h\right)\\
&+\sum_{K\in\mathcal{T}_h}\frac{1}{\lambda_K}\left(\frac{32}{105}\right)^2\left(\frac{\boldsymbol{u}_{h,l}^{n+1}-\boldsymbol{u}_{h,l}^{n}}{\Delta t}+\nabla p_h^{n+1}-\boldsymbol{f}(t^{n+1}),\nabla q_h-\frac{1}{\Delta t}\boldsymbol{v}_h\right)_K\\
&=\left(\boldsymbol{f}(t^{n+1}),\boldsymbol{v}_h\right)+\sum_{K\in\mathcal{T}_h}\frac{1}{\lambda_K}\frac{32}{105}\left(\boldsymbol{f}(t^{n+1})+\frac{\boldsymbol{u}_{h,l}^{n}}{\Delta t},\left(\overline{\psi_b^K}-\psi_b^K\right)\left(\frac{\overline{\boldsymbol{v}_h}}{\Delta t}-\nabla q_h\right)\right)_K\\
&-\sum_{K\in\mathcal{T}_h}\frac{1}{\lambda_K}\left(\frac{32}{105}\right)^2\left(\frac{\boldsymbol{u}_{h,l}^{n+1}-\boldsymbol{u}_{h,l}^{n}}{\Delta t}-\boldsymbol{f}(t^{n+1}),\frac{\boldsymbol{v}_h-\overline{\boldsymbol{v}_h}}{\Delta t}\right)_K\\
&-\sum_{K\in\mathcal{T}_h}\frac{1}{\lambda_K}\frac{32}{105}\left(\frac{\boldsymbol{u}_{h,b}^{n}}{\Delta t},\psi_b^K\left(\frac{\overline{\boldsymbol{v}_h}}{\Delta t}-\nabla q_h\right)\right)_K+\left(\frac{1}{\Delta t}\boldsymbol{u}_{h,b}^{n},\boldsymbol{v}_h\right)\quad\forall\boldsymbol{v}_h\in V_h,\forall q_h\in Q_h
\end{aligned}
\tag{3.30}
$$

where $\overline{\psi_b^K}=\frac{32}{105}$ is the average value of the bubble function on K.

We recognize a PSPG variational formulation with parameters $\beta=1$, $\gamma=-\frac{1}{\Delta t}$ and $\alpha_K=\frac{1}{\lambda_K}\left(\frac{32}{105}\right)^2$ and some additional terms. The stabilization parameter is

$$
\frac{1}{\lambda_K}\left(\frac{32}{105}\right)^2=\frac{\left(\frac{32}{105}\right)^2}{\frac{8192}{51975}\frac{1}{\Delta t}+\frac{\nu}{|K|}\int_K\nabla\psi_b^K\cdot\nabla\psi_b^K}.
$$

Since we assumed a regular mesh, we can write $\frac{1}{|K|}\int_K\nabla\psi_b^K\cdot\nabla\psi_b^K=\frac{C_K^b}{h_K^2}$ where the constant C_K^b depends just on the shape and not on the size of K and h_K is the size of K. We can express the stabilization parameter as a weighted harmonic mean between Δt and $\frac{h_K^2}{\nu}$ as

$$
\frac{1}{\lambda_K}\left(\frac{32}{105}\right)^2=\left(\frac{32}{105}\right)^2\frac{\Delta t\frac{h_K^2}{\nu}}{\frac{8192}{51975}\frac{h_K^2}{\nu}+C_K^b\Delta t}
$$

To give an approximate order of the constant C_K^b, for a tetrahedron K similar in the geometrical sense to the reference tetrahedron, we have $C_K^b=\frac{8192}{315}$. This stabilization parameter is discussed further in Section 3.1.6.

In (3.30), we have eliminated the unknown $\boldsymbol{u}_{h,b}^{n+1}$ which means that we also cannot compute the value of $\boldsymbol{u}_{h,b}^{n}$ further than the initial step simply using that equation. However, assuming that we know $\boldsymbol{u}_{h,b}^{n}$, we can compute $\boldsymbol{u}_{h,l}^{n+1}$ and then the bubble value $\boldsymbol{u}_{h,b}^{n+1}$ can be reconstructed using the lines that we have eliminated when getting from M_K to M_K'. Those lines express the

following

$$\left(\frac{(\boldsymbol{u}_{h,l}^{n+1}+\boldsymbol{u}_{h,b}^{n+1})-(\boldsymbol{u}_{h,l}^{n}+\boldsymbol{u}_{h,b}^{n})}{\Delta t},\psi_b^K\right)_K+\nu\left(\nabla\boldsymbol{u}_{h,b}^{n+1},\nabla\psi_b^K\right)_K-\left(p_h^{n+1},\nabla.\psi_b^K\right)_K=\left(\boldsymbol{f}(t^{n+1}),\psi_b^K\right)_K. \tag{3.31}$$

Since the support of the bubble basis function ψ_b^K is restricted to K, we can explicitly and locally reconstruct $\boldsymbol{u}_{h,b}^{n+1}$. We thus have an unconditionally stable method with respect to the timestep while solving a linear system only of the size of a standard $\mathbb{P}_1-\mathbb{P}_1$ method. We will call this method the **bubble reconstruction** method. This method requires storing the $\boldsymbol{u}_{h,b}^{n}$ coefficients from one timestep to another.

We also propose another method where we simply set $\boldsymbol{u}_{h,b}^{n}=0$ and use the remaining terms in (3.30) to compute $\boldsymbol{u}_{h,l}^{n+1}$. We call this method the **bubble elimination** method. It does not require storing any extra coefficients from one timestep to another.

Remark 2. *Although the weak formulation* (3.30) *appears unwieldy to program, in practice the bubble elimination can be implemented efficiently and rather easily by performing a pressure-Schur-like elimination at the elemental level before assembling the elemental matrices together.*

In [51], the assumption that the bubble functions are quasi-static has been made, meaning that

$$\left(\frac{\boldsymbol{u}_{h,b}^{n+1}-\boldsymbol{u}_{h,b}^{n}}{\Delta t},\boldsymbol{v}_h\right)=0,\quad \forall\boldsymbol{v}_h\in V_h\oplus V_{bub}.$$

Let us assume that the bubble functions are quasi-static, we can then do a similar elimination of bubble unknowns as above and (3.29) then becomes

$$\begin{aligned}
&\left(\frac{\boldsymbol{u}_{h,l}^{n+1}-\boldsymbol{u}_{h,l}^{n}}{\Delta t},\boldsymbol{v}_h\right)+\nu\left(\nabla\boldsymbol{u}_{h,l}^{n+1},\nabla\boldsymbol{v}_h\right)-\left(p_h^{n+1},\nabla\cdot\boldsymbol{v}_h\right)+\left(\nabla\cdot\boldsymbol{u}_{h,l}^{n+1},q_h\right)\\
&+\sum_{K\in\mathcal{T}_h}\frac{1}{\lambda_K}\left(\frac{32}{105}\right)^2\left(\frac{\boldsymbol{u}_{h,l}^{n+1}}{\Delta t}+\nabla p_h^{n+1},\nabla q_h\right)_K\\
&=\left(\boldsymbol{f}(t^{n+1}),\boldsymbol{v}_h\right)+\sum_{K\in\mathcal{T}_h}\frac{1}{\lambda_K}\frac{32}{105}\left(\boldsymbol{f}(t^{n+1})+\frac{\boldsymbol{u}_{h,l}^{n}}{\Delta t},\psi_b^K\nabla q_h\right)_K\\
&\forall\boldsymbol{v}_h\in V_h,\forall q_h\in Q_h
\end{aligned} \tag{3.32}$$

where

$$\lambda_K=\frac{\nu}{|K|}\int_K\nabla\psi_b^K\cdot\nabla\psi_b^K=C_K^b\frac{\nu}{h_K^2}.$$

The quasi-static assumption drops the timestep dependency in the stabilization parameter which becomes simply the usual $C\dfrac{h_K^2}{\nu}$.

(3.32) can then be rewritten as

$$\begin{aligned}
&\left(\frac{\boldsymbol{u}_{h,l}^{n+1}-\boldsymbol{u}_{h,l}^{n}}{\Delta t},\boldsymbol{v}_h\right)+\nu\left(\nabla\boldsymbol{u}_{h,l}^{n+1},\nabla\boldsymbol{v}_h\right)-\left(p_h^{n+1},\nabla\cdot\boldsymbol{v}_h\right)+\left(\nabla\cdot\boldsymbol{u}_{h,l}^{n+1},q_h\right)\\
&+\sum_{K\in\mathcal{T}_h}\frac{1}{\lambda_K}\left(\frac{32}{105}\right)^2\left(\nabla p_h^{n+1}-\frac{\psi_b^K}{\overline{\psi_b^K}}\boldsymbol{f}(t^{n+1}),\nabla q_h\right)_K=\left(\boldsymbol{f}(t^{n+1}),\boldsymbol{v}_h\right)\\
&-\sum_{K\in\mathcal{T}_h}\frac{1}{\lambda_K}\frac{32}{105}\left(\frac{\boldsymbol{u}_{h,l}^{n+1}}{\Delta t}\overline{\psi_b^K}-\frac{\boldsymbol{u}_{h,l}^{n}}{\Delta t}\psi_b^K,\nabla q_h\right)_K\qquad\forall\boldsymbol{v}_h\in V_h,\forall q_h\in Q_h.
\end{aligned}\tag{3.33}$$

again where $\overline{\psi_b^K}=\frac{32}{105}$ is the average value of ψ_b^K on element K. With an additional bubble function scaling of the force term in the stabilization and an extra term, we recognize (3.33) as another PSPG scheme with parameters $\beta=0$, $\gamma=0$ and $\alpha_K=\left(\frac{32}{105}\right)^2\frac{1}{C_K^b}\frac{h_K^2}{\nu}$.

The link between bubble enrichment and PSPG type schemes has been established and seems to suggest a stabilization parameter being the harmonic mean of timestep Δt and squared mesh size $\frac{h_K^2}{\nu}$. To our knowledge, this parameter had not previously been derived from the theory but has been motivated by practice for instance in [98]. It is interesting to see that making a quasi-static assumption on the bubbles, the corresponding stabilization parameter is reduced to simply the classic stabilization parameter $\frac{h_K^2}{\nu}$.

3.1.6 Possible choices of the stabilization parameter

A standard and widely used choice for the stabilization parameter α_K has been

$$\alpha_K=\bar{\alpha}\frac{h_K^2}{\nu}$$

at least in the case of the stationary Stokes equations [48, 99, 46] but also for the time-dependent Stokes equations [92, 56]. This choice allows convergence of the pressure in the L^2 norm as proved in Section 3.3. $\bar{\alpha}$ is a dimensionless parameter which has to be chosen and that choice will be discussed. We will call this parameter choice the **spatial** stabilization parameter.

In [100, 101, 98] another stabilization parameter for time-dependent problems is proposed which is a harmonic mean of timestep and squared mesh size h_K^2. Although this choice of parameter was originally motivated by practice, the link with the bubble enrichment in Section 3.1.5 provides a theoretical motivation for this choice of stabilization parameter as well. We will call

$$\alpha_K=\bar{\alpha}\frac{\Delta t\frac{h_K^2}{\nu}}{\frac{h_K^2}{\nu}+\underline{\alpha}\Delta t}$$

the **transient** stabilization parameter along the lines of [90]. $\bar{\alpha}$ and $\underline{\alpha}$ are dimensionless parameters. Note that the physical units of the stabilization parameter reduce to seconds which is the same as for the spatial stabilization parameter. $\underline{\alpha}$ sets the weight of the timestep compared to the squared mesh size in the harmonic mean. While [98] suggests a scaling of $\underline{\alpha} = \frac{1}{2}$, the derivation in Section 3.1.5 assuming a regular mesh suggests scalings of

$$\bar{\alpha} = \frac{33}{56}, \quad \underline{\alpha} = 165.$$

It should be noted that when $\Delta t >> \dfrac{h_K^2}{\nu}$, the transient and spatial stabilizations are equivalent. However, when $\Delta t << \dfrac{h_K^2}{\nu}$, we have for the transient stabilization parameter that $\alpha_K \approx \Delta t$. In the latter case, the pressure is less stabilized and we could not prove L^2 convergence of the pressure.

We prove in Section 3.2 the stability and convergence of the PSPG scheme assuming a spatial stabilization parameter. In the small timestep limit, instabilities have been reported by [90] and it was noted that in the small timestep limit, the stabilization parameter must scale as Δt which is the case for the transient parameter but not for the spatial parameter. However, this condition violates the pressure stability condition $\alpha_K \geq \frac{h_K^2}{\nu}$ in Section 3.2. We will perform numerical experiments to determine performance and stability of spatial and transient parameters.

3.2 Stability of the schemes for velocity and pressure for time-dependent Stokes

To prove a stability result on the pressure for the stabilized scheme, we use the lemma proven in [99], [102] using an argument by Verfürth [103], also called Verfürth's trick or generalized inf-sup condition.

Lemma 1. *Let V_h, Q_h be spaces as defined in* (3.4), (3.5) *and $\boldsymbol{v}_h \in V_h$. Given $p_h \in Q_h$ there exist constants $C_1 > 0$, $C_2 > 0$ such that*

$$\sup_{0 \neq \boldsymbol{v}_h \in V_h} \frac{(p_h, \nabla \cdot \boldsymbol{v}_h)}{\|\boldsymbol{v}_h\|_{H^1(\Omega)}} \geq C_1 \|p_h\|_{L^2(\Omega)} - C_2 \left(\sum_{K \in \mathcal{T}_h} h_K^2 \|\nabla p_h\|_{L^2(K)}^2 \right)^{1/2}. \tag{3.34}$$

Using continuous $\mathbb{P}_1$ finite elements and a consistent stabilization with an implicit Euler time discretization, the weak problem (3.2)-(3.3) becomes the following fully discretized problem. Find $(\boldsymbol{u}_h^{n+1}, p_h^{n+1}) \in V_h \times Q_h$ such that

$$\left(\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t},\boldsymbol{v}_h\right)+\nu\left(\nabla\boldsymbol{u}_h^{n+1},\nabla\boldsymbol{v}_h\right)-\left(p_h^{n+1},\nabla\cdot\boldsymbol{v}_h\right)=\left(\boldsymbol{f}(t^{n+1}),\boldsymbol{v}_h\right)\quad\forall\boldsymbol{v}_h\in V_h \tag{3.35}$$

$$\left(\nabla\cdot\boldsymbol{u}_h^{n+1},q_h\right)+\sum_{K\in\mathcal{T}_h}\alpha_K\left(\beta\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}-\nu\Delta\boldsymbol{u}_h^{n+1}+\nabla p_h^{n+1}-\boldsymbol{g}(t^{n+1}),\nabla q_h\right)_K=0\quad\forall q_h\in Q_h \tag{3.36}$$

for $n\geq 0$ where $\alpha_K>0\ \forall K\in\mathcal{T}_h$ are stabilization parameters associated with element K and $\beta=0$ or 1 depending on the choice of non-consistent or consistent stabilization respectively. Note that in the stabilization term, we consider a function $\boldsymbol{g}\in L^2(0,T;L_0^2(\Omega))$ instead of $\boldsymbol{f}$ for a more general result to be used further. $\boldsymbol{u}_h^0\in V_h$ is an approximation of $\boldsymbol{u}_0$ and we assume that $\boldsymbol{f}\in L^2(0,T;L_0^2(\Omega))$. Note that $\Delta\boldsymbol{u}_h=0$ for $\mathbb{P}_1$ elements. Let $\alpha_{min}=\min\limits_{K\in\mathcal{T}_h}\alpha_K$ and $\alpha_{max}=\max\limits_{K\in\mathcal{T}_h}\alpha_K$ and also $h_{max}=\max\limits_{K\in\mathcal{T}_h}h_K$. The following result gives the stability of the numerical solution.

Theorem 1. *Let $(\boldsymbol{u}_h^n,p_h^n)\in V_h\times Q_h$ be a solution of the above problem for $n=0,\ldots,N$.*

Assume that the stabilization parameters $\alpha_K>0$ are chosen such that there exists a constant $C_\alpha>0$ such that for all K in $\mathcal{T}_h$ and $h_K>0$, α_K satisfies $h_K^2\leq C_\alpha\,\alpha_K$.

Assume also that the timestep $0<\Delta t<1$ satisfies the stability condition $\alpha_{max}\leq\Delta t$ for the consistent scheme $\beta=1$.

Then, the following stability estimates for the velocity and pressure holds

$$\begin{aligned}&\left\|\boldsymbol{u}_h^N\right\|_{L^2(\Omega)}^2+\nu\Delta t\sum_{n=1}^N\left\|\nabla\boldsymbol{u}_h^n\right\|_{L^2(\Omega)}^2+\frac{\Delta t}{2}\sum_{n=1}^N\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^n\right\|_{L^2(K)}^2\\&\leq\left\|\boldsymbol{u}_h^0\right\|_{L^2(\Omega)}^2+\Delta t\sum_{i=1}^N\left(\frac{C_p^2}{\nu}\left\|\boldsymbol{f}(t^i)\right\|_{L^2(\Omega)}^2+\alpha_{max}\left\|\boldsymbol{g}(t^i)\right\|_{L^2(\Omega)}^2\right).\end{aligned}$$

$$\begin{aligned}\Delta t\sum_{n=1}^N\left\|p_h^n\right\|_{L^2(\Omega)}^2\leq C\Bigg[&\frac{1}{\Delta t}\left\|\nabla\cdot\boldsymbol{u}_h^0\right\|_{L^2(\Omega)}^2+\max(1,C_\alpha)\left\|\boldsymbol{u}_h^0\right\|_{L^2(\Omega)}^2+\nu\left\|\nabla\boldsymbol{u}_h^0\right\|_{L^2(\Omega)}^2\\&+\max(1,C_\alpha)\left((1+\frac{C_p^2}{\nu})\,\Delta t\sum_{n=1}^N\left\|\boldsymbol{f}(t^n)\right\|_{L^2(\Omega)}^2+\alpha_{max}\,\Delta t\sum_{n=1}^N\left\|\boldsymbol{g}(t^n)\right\|_{L^2(\Omega)}^2\right)\Bigg].\end{aligned}$$

Remark 3. *Note that in particular $\frac{1}{\Delta t}\left\|\nabla\cdot\boldsymbol{u}_h^0\right\|_{L^2(\Omega)}\leq C\left\|D^2\boldsymbol{u}_0\right\|_{L^2(\Omega)}^2$ holds under the stability condition $\alpha_{max}\leq\Delta t$ if $\boldsymbol{u}_h^0$ is chosen as the Lagrange interpolant in V_h of $\boldsymbol{u}_0$ and $\nabla\cdot\boldsymbol{u}_0=0$ with $\boldsymbol{u}_0\in H^2(\Omega)$.*

Proof. We will first derive the stability result for the velocity. Taking $\boldsymbol{v}_h=\boldsymbol{u}_h^{n+1}$ in (3.35) and

$q_h = p_h^{n+1}$ in (3.36) with $n \in \{0, ..., N\}$ yields

$$\begin{aligned}
&\left(\frac{\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n}{\Delta t}, \boldsymbol{u}_h^{n+1}\right) + \nu\left(\nabla \boldsymbol{u}_h^{n+1}, \nabla \boldsymbol{u}_h^{n+1}\right) \\
&+ \sum_{K\in\mathcal{T}_h} \alpha_K\, \beta \left(\frac{\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n}{\Delta t}, \nabla p_h^{n+1}\right)_K + \sum_{K\in\mathcal{T}_h} \alpha_K \left(\nabla p_h^{n+1} - \boldsymbol{g}(t^{n+1}), \nabla p_h^{n+1}\right)_K \\
&= \left(\boldsymbol{f}(t^{n+1}), \boldsymbol{u}_h^{n+1}\right) \qquad (3.37)
\end{aligned}$$

which leads to the inequality

$$\begin{aligned}
&\frac{1}{2\Delta t}\left\|\boldsymbol{u}_h^{n+1}\right\|^2_{L^2(\Omega)} + \frac{1}{2\Delta t}\left\|\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n\right\|^2_{L^2(\Omega)} + \nu\left\|\nabla \boldsymbol{u}_h^{n+1}\right\|^2_{L^2(\Omega)} \\
&+ \sum_{K\in\mathcal{T}_h} \frac{\alpha_K}{\Delta t}\beta\left(\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n, \nabla p_h^{n+1}\right)_K + \sum_{K\in\mathcal{T}_h} \alpha_K\left(\nabla p_h^{n+1} - \boldsymbol{g}(t^{n+1}), \nabla p_h^{n+1}\right)_K \\
&\leq \frac{1}{2\Delta t}\left\|\boldsymbol{u}_h^n\right\|^2_{L^2(\Omega)} + \left(\boldsymbol{f}(t^{n+1}), \boldsymbol{u}_h^{n+1}\right). \qquad (3.38)
\end{aligned}$$

Using Cauchy-Schwarz and Young's inequalities, we get

$$\begin{aligned}
&\frac{1}{2\Delta t}\left\|\boldsymbol{u}_h^{n+1}\right\|^2_{L^2(\Omega)} + \frac{1-\beta}{2\Delta t}\left\|\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n\right\|^2_{L^2(\Omega)} + \nu\left\|\nabla \boldsymbol{u}_h^{n+1}\right\|^2_{L^2(\Omega)} + \sum_{K\in\mathcal{T}_h} \alpha_K\left\|\nabla p_h^{n+1}\right\|^2_{L^2(K)} \\
&\leq \frac{1}{2\Delta t}\left\|\boldsymbol{u}_h^n\right\|^2_{L^2(\Omega)} + \sum_{K\in\mathcal{T}_h} \alpha_K\left(\boldsymbol{g}(t^{n+1}), \nabla p_h^{n+1}\right)_K + \sum_{K\in\mathcal{T}_h} \frac{\alpha_K^2}{2\Delta t}\beta\left\|\nabla p_h^{n+1}\right\|^2_{L^2(K)} + \left(\boldsymbol{f}(t^{n+1}), \boldsymbol{u}_h^{n+1}\right) \\
&\qquad (3.39)
\end{aligned}$$

which by using the Poincaré inequality with constant C_p leads to

$$\begin{aligned}
&\frac{1}{\Delta t}\left\|\boldsymbol{u}_h^{n+1}\right\|^2_{L^2(\Omega)} + \frac{1-\beta}{\Delta t}\left\|\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n\right\|^2_{L^2(\Omega)} + \nu\left\|\nabla \boldsymbol{u}_h^{n+1}\right\|^2_{L^2(\Omega)} + \sum_{K\in\mathcal{T}_h} \alpha_K\left(\frac{3}{2} - \frac{\alpha_K\,\beta}{\Delta t}\right)\left\|\nabla p_h^{n+1}\right\|^2_{L^2(K)} \\
&\leq \frac{1}{\Delta t}\left\|\boldsymbol{u}_h^n\right\|^2_{L^2(\Omega)} + \sum_{K\in\mathcal{T}_h}\left(\frac{C_p^2}{\nu}\left\|\boldsymbol{f}(t^{n+1})\right\|^2_{L^2(K)} + 2\alpha_K\left\|\boldsymbol{g}(t^{n+1})\right\|^2_{L^2(K)}\right). \qquad (3.40)
\end{aligned}$$

With the assumption $\alpha_{max} \leq \Delta t$ if $\beta = 1$, (3.40) then gives

$$\begin{aligned}
&\frac{1}{\Delta t}\left\|\boldsymbol{u}_h^{n+1}\right\|^2_{L^2(\Omega)} + \frac{1-\beta}{\Delta t}\left\|\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n\right\|^2_{L^2(\Omega)} + \nu\left\|\nabla \boldsymbol{u}_h^{n+1}\right\|^2_{L^2(\Omega)} + \frac{1}{2}\sum_{K\in\mathcal{T}_h} \alpha_K\left\|\nabla p_h^{n+1}\right\|^2_{L^2(K)} \\
&\leq \frac{1}{\Delta t}\left\|\boldsymbol{u}_h^n\right\|^2_{L^2(\Omega)} + \frac{C_p^2}{\nu}\left\|\boldsymbol{f}(t^{n+1})\right\|^2_{L^2(\Omega)} + 2\alpha_{max}\left\|\boldsymbol{g}(t^{n+1})\right\|^2_{L^2(\Omega)} \qquad (3.41)
\end{aligned}$$

which by summing leads to the stability result for the velocity and pressure

$$\begin{aligned}&\left\|\boldsymbol{u}_h^N\right\|_{L^2(\Omega)}^2+\nu\Delta t\sum_{n=1}^{N}\left\|\nabla\boldsymbol{u}_h^n\right\|_{L^2(\Omega)}^2+\frac{\Delta t}{2}\sum_{n=1}^{N}\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^n\right\|_{L^2(K)}^2\\&\leq\left\|\boldsymbol{u}_h^0\right\|_{L^2(\Omega)}^2+\Delta t\sum_{i=1}^{N}\left(\frac{C_p^2}{\nu}\left\|\boldsymbol{f}(t^i)\right\|_{L^2(\Omega)}^2+2\alpha_{max}\left\|\boldsymbol{g}(t^i)\right\|_{L^2(\Omega)}^2\right).\end{aligned}\tag{3.42}$$

Applying Lemma 1 to p_h^{n+1} yields the inequality

$$C_1\left\|p_h^{n+1}\right\|_{L^2(\Omega)}\leq\sup_{0\neq\boldsymbol{v}_h\in V_h}\frac{\left(p_h^{n+1},\nabla\cdot\boldsymbol{v}_h\right)}{\left\|\boldsymbol{v}_h\right\|_{H^1(\Omega)}}+C_2\left(\sum_{K\in\mathcal{T}_h}h_K^2\left\|\nabla p_h^{n+1}\right\|_{L^2(K)}^2\right)^{1/2}.\tag{3.43}$$

Using (3.35) and Cauchy-Schwarz we get $\forall\,\boldsymbol{v}_h\in V_h$

$$\frac{\left(p_h^{n+1},\nabla\cdot\boldsymbol{v}_h\right)}{\left\|\boldsymbol{v}_h\right\|_{H^1(\Omega)}}=\frac{\left(\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t},\boldsymbol{v}_h\right)+\nu\left(\nabla\boldsymbol{u}_h^{n+1},\nabla\boldsymbol{v}_h\right)-\left(\boldsymbol{f}(t^{n+1}),\boldsymbol{v}_h\right)}{\left\|\boldsymbol{v}_h\right\|_{H^1(\Omega)}}\tag{3.44}$$

$$\leq\left\|\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}\right\|_{L^2(\Omega)}+\nu\left\|\nabla\boldsymbol{u}_h^{n+1}\right\|_{L^2(\Omega)}+\left\|\boldsymbol{f}(t^{n+1})\right\|_{L^2(\Omega)}.\tag{3.45}$$

Squaring (3.43) and inserting (3.44) then yields for a constant $C_3>0$ the inequality

$$\begin{aligned}\left\|p_h^{n+1}\right\|_{L^2(\Omega)}^2\leq C_3\Bigg(&\left\|\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}\right\|_{L^2(\Omega)}^2+\nu\left\|\nabla\boldsymbol{u}_h^{n+1}\right\|_{L^2(\Omega)}^2\\&+\left\|\boldsymbol{f}(t^{n+1})\right\|_{L^2(\Omega)}^2+\sum_{K\in\mathcal{T}_h}h_K^2\left\|\nabla p_h^{n+1}\right\|_{L^2(K)}^2\Bigg).\end{aligned}\tag{3.46}$$

We sum (3.46) for $n=0,\ldots,N-1$ and multiply by Δt. Using the assumption $h_K^2\leq C_\alpha\alpha_K\ \ \forall K\in\mathcal{T}_h$ then gives

$$\begin{aligned}\Delta t\sum_{n=1}^{N}\left\|p_h^n\right\|_{L^2(\Omega)}^2\leq C_3\Delta t\sum_{n=0}^{N-1}\Bigg(&\left\|\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}\right\|_{L^2(\Omega)}^2+\nu\left\|\nabla\boldsymbol{u}_h^{n+1}\right\|_{L^2(\Omega)}^2\\&+C_\alpha\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^{n+1}\right\|_{L^2(K)}^2+\left\|\boldsymbol{f}(t^{n+1})\right\|_{L^2(\Omega)}^2\Bigg).\end{aligned}\tag{3.47}$$

It remains only to bound $\left\|\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}\right\|_{L^2(\Omega)}^2$ of the terms on the right-hand side of (3.47) since the rest has been bounded in (3.42). Let us suppose first that $n\geq 1$. We take (3.35) with

$\boldsymbol{v}_h = \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}$. This yields

$$\left\| \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} \right\|_{L^2(\Omega)}^2 + \nu \left(\nabla \boldsymbol{u}_h^{n+1}, \nabla \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} \right) - \left(p_h^{n+1}, \nabla \cdot \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} \right) = \left(\boldsymbol{f}(t^{n+1}), \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} \right). \tag{3.48}$$

We subtract (3.36) for two consecutive timesteps and take $q_h = p_h^{n+1}$. We get

$$\Delta t \left(\nabla \cdot \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}, p_h^{n+1} \right) + \sum_{K\in\mathcal{T}_h} \alpha_K \left(\beta \left(\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} - \frac{\boldsymbol{u}_h^{n}-\boldsymbol{u}_h^{n-1}}{\Delta t} \right) + \nabla(p_h^{n+1}-p_h^n), \nabla p_h^{n+1} \right)_K = \sum_{K\in\mathcal{T}_h} \alpha_K \left(\boldsymbol{g}(t^{n+1}) - \boldsymbol{g}(t^n), \nabla p_h^{n+1} \right)_K. \tag{3.49}$$

Plugging in yields

$$\begin{aligned} &\left\| \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} \right\|_{L^2(\Omega)}^2 + \nu \left(\nabla \boldsymbol{u}_h^{n+1}, \nabla \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} \right) \\ &+ \sum_{K\in\mathcal{T}_h} \frac{\alpha_K}{\Delta t} \left(\beta \left(\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} - \frac{\boldsymbol{u}_h^{n}-\boldsymbol{u}_h^{n-1}}{\Delta t} \right) + \nabla(p_h^{n+1}-p_h^n), \nabla p_h^{n+1} \right)_K \\ &= \left(\boldsymbol{f}(t^{n+1}), \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} \right) + \sum_{K\in\mathcal{T}_h} \frac{\alpha_K}{\Delta t} \left(\boldsymbol{g}(t^{n+1}) - \boldsymbol{g}(t^n), \nabla p_h^{n+1} \right)_K. \end{aligned} \tag{3.50}$$

Using Cauchy Schwarz and Young's inequalities, we get

$$\begin{aligned} &\Delta t \left\| \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} \right\|_{L^2(\Omega)}^2 + \frac{\nu}{2} \left\| \nabla \boldsymbol{u}_h^{n+1} \right\|_{L^2(\Omega)}^2 + \frac{\nu}{2} \left\| \nabla(\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n) \right\|_{L^2(\Omega)}^2 \\ &+ \sum_{K\in\mathcal{T}_h} \frac{\alpha_K}{2} \left\| \nabla p_h^{n+1} \right\|_{L^2(K)}^2 + \sum_{K\in\mathcal{T}_h} \frac{\alpha_K}{2} \left\| \nabla(p_h^{n+1}-p_h^n) \right\|_{L^2(K)}^2 \\ &+ \sum_{K\in\mathcal{T}_h} \alpha_K \beta \left(\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} - \frac{\boldsymbol{u}_h^{n}-\boldsymbol{u}_h^{n-1}}{\Delta t}, \nabla p_h^{n+1} \right)_K \\ &= \frac{\nu}{2} \left\| \nabla \boldsymbol{u}_h^n \right\|_{L^2(\Omega)}^2 + \sum_{K\in\mathcal{T}_h} \frac{\alpha_K}{2} \left\| \nabla p_h^n \right\|_{L^2(K)}^2 + \Delta t \left(\boldsymbol{f}(t^{n+1}), \frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t} \right) \\ &+ \sum_{K\in\mathcal{T}_h} \alpha_K \left(\boldsymbol{g}(t^{n+1}) - \boldsymbol{g}(t^n), \nabla p_h^{n+1} \right)_K. \end{aligned} \tag{3.51}$$

and therefore

$$
\begin{aligned}
&\frac{3\Delta t}{2}\left\|\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}\right\|_{L^2(\Omega)}^2+\nu\left\|\nabla\boldsymbol{u}_h^{n+1}\right\|_{L^2(\Omega)}^2\\
&+\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^{n+1}\right\|_{L^2(K)}^2+\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla(p_h^{n+1}-p_h^n)\right\|_{L^2(K)}^2\\
&+2\sum_{K\in\mathcal{T}_h}\alpha_K\beta\left(\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}-\frac{\boldsymbol{u}_h^n-\boldsymbol{u}_h^{n-1}}{\Delta t},\nabla p_h^{n+1}\right)_K\\
&\leq\nu\left\|\nabla\boldsymbol{u}_h^n\right\|_{L^2(\Omega)}^2+\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^n\right\|_{L^2(K)}^2+2\Delta t\left\|\boldsymbol{f}(t^{n+1})\right\|_{L^2(\Omega)}^2\\
&+2\sum_{K\in\mathcal{T}_h}\alpha_K\left(\boldsymbol{g}(t^{n+1})-\boldsymbol{g}(t^n),\nabla p_h^{n+1}\right)_K.
\end{aligned}
\tag{3.52}
$$

Before summing this inequality for $n=1,\ldots,N-1$, we will write the discrete equivalent of integrating by parts in time the scalar product terms in (3.52). We have

$$
\begin{aligned}
&\sum_{n=1}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\left(\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}-\frac{\boldsymbol{u}_h^n-\boldsymbol{u}_h^{n-1}}{\Delta t},\nabla p_h^{n+1}\right)_K=-\sum_{n=2}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\left(\frac{\boldsymbol{u}_h^n-\boldsymbol{u}_h^{n-1}}{\Delta t},\nabla(p_h^{n+1}-p_h^n)\right)_K\\
&+\sum_{K\in\mathcal{T}_h}\alpha_K\left(\frac{\boldsymbol{u}_h^N-\boldsymbol{u}_h^{N-1}}{\Delta t},\nabla p_h^N\right)_K-\sum_{K\in\mathcal{T}_h}\alpha_K\left(\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t},\nabla p_h^2\right)_K.
\end{aligned}
\tag{3.53}
$$

Similarly,

$$
\begin{aligned}
&\sum_{n=1}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\left(\boldsymbol{g}(t^{n+1})-\boldsymbol{g}(t^n),\nabla p_h^{n+1}\right)_K=-\sum_{n=2}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\left(\boldsymbol{g}(t^n),\nabla(p_h^{n+1}-p_h^n)\right)_K\\
&+\sum_{K\in\mathcal{T}_h}\alpha_K\left(\boldsymbol{g}(t^N),\nabla p_h^N\right)_K-\sum_{K\in\mathcal{T}_h}\alpha_K\left(\boldsymbol{g}(t^1),\nabla p_h^2\right)_K.
\end{aligned}
\tag{3.54}
$$

Now summing (3.52) for $n = 1, ..., N-1$ and using (3.53) and (3.54) yields

$$
\begin{aligned}
&\frac{3\Delta t}{2}\sum_{n=1}^{N-1}\left\|\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}\right\|_{L^2(\Omega)}^2+\nu\sum_{n=1}^{N-1}\left\|\nabla\boldsymbol{u}_h^{n+1}\right\|_{L^2(\Omega)}^2\\
&+\sum_{n=1}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^{n+1}\right\|_{L^2(K)}^2+\sum_{n=1}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla(p_h^{n+1}-p_h^n)\right\|_{L^2(K)}^2\\
&\le\nu\sum_{n=1}^{N-1}\left\|\nabla\boldsymbol{u}_h^n\right\|_{L^2(\Omega)}^2+\sum_{n=1}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^n\right\|_{L^2(K)}^2+2\sum_{n=1}^{N-1}\Delta t\left\|\boldsymbol{f}(t^{n+1})\right\|_{L^2(\Omega)}^2\\
&-\sum_{n=2}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\left(\boldsymbol{g}(t^n),\nabla(p_h^{n+1}-p_h^n)\right)_K\\
&+2\sum_{K\in\mathcal{T}_h}\alpha_K\left(\boldsymbol{g}(t^N),\nabla p_h^N\right)_K-2\sum_{K\in\mathcal{T}_h}\alpha_K\left(\boldsymbol{g}(t^1),\nabla p_h^2\right)_K\\
&+2\sum_{n=2}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\beta\left(\frac{\boldsymbol{u}_h^n-\boldsymbol{u}_h^{n-1}}{\Delta t},\nabla(p_h^{n+1}-p_h^n)\right)_K-2\sum_{K\in\mathcal{T}_h}\alpha_K\beta\left(\frac{\boldsymbol{u}_h^N-\boldsymbol{u}_h^{N-1}}{\Delta t},\nabla p_h^N\right)_K\\
&+2\sum_{K\in\mathcal{T}_h}\alpha_K\beta\left(\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t},\nabla p_h^2\right)_K.
\end{aligned}
\tag{3.55}
$$

Eliminating identical terms on both sides of the inequality and using Cauchy-Schwarz and Young, we get

$$
\begin{aligned}
&\frac{3\Delta t}{2}\sum_{n=1}^{N-1}\left\|\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}\right\|_{L^2(\Omega)}^2+\nu\left\|\nabla\boldsymbol{u}_h^N\right\|_{L^2(\Omega)}^2\\
&+\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^N\right\|_{L^2(K)}^2+\sum_{n=1}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla(p_h^{n+1}-p_h^n)\right\|_{L^2(K)}^2\\
&\le\nu\left\|\nabla\boldsymbol{u}_h^1\right\|_{L^2(\Omega)}^2+\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^1\right\|_{L^2(K)}^2+2\sum_{n=1}^{N-1}\Delta t\left\|\boldsymbol{f}(t^{n+1})\right\|_{L^2(\Omega)}^2\\
&+\sum_{n=2}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\left[5\left\|\boldsymbol{g}(t^n)\right\|_{L^2(K)}^2+\frac{1}{5}\left\|\nabla(p_h^{n+1}-p_h^n)\right\|_{L^2(K)}^2\right]\\
&+\sum_{K\in\mathcal{T}_h}\alpha_K\left[5\left\|\boldsymbol{g}(t^N)\right\|_{L^2(K)}^2+\frac{1}{5}\left\|\nabla p_h^N\right\|_{L^2(K)}^2\right]+\sum_{K\in\mathcal{T}_h}\alpha_K\left[4\left\|\boldsymbol{g}(t^1)\right\|_{L^2(K)}^2+\frac{1}{4}\left\|\nabla p_h^2\right\|_{L^2(K)}^2\right]\\
&+\sum_{n=2}^{N-1}\sum_{K\in\mathcal{T}_h}\alpha_K\beta\left[\frac{5}{4}\left\|\frac{\boldsymbol{u}_h^n-\boldsymbol{u}_h^{n-1}}{\Delta t}\right\|_{L^2(K)}^2+\frac{4}{5}\left\|\nabla(p_h^{n+1}-p_h^n)\right\|_{L^2(K)}^2\right]\\
&+\sum_{K\in\mathcal{T}_h}\alpha_K\beta\left[\frac{5}{4}\left\|\frac{\boldsymbol{u}_h^N-\boldsymbol{u}_h^{N-1}}{\Delta t}\right\|_{L^2(K)}^2+\frac{4}{5}\left\|\nabla p_h^N\right\|_{L^2(K)}^2\right]\\
&+\sum_{K\in\mathcal{T}_h}\alpha_K\beta\left[4\left\|\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right\|_{L^2(K)}^2+\frac{1}{4}\left\|\nabla p_h^2\right\|_{L^2(K)}^2\right].
\end{aligned}
\tag{3.56}
$$

Simplifying the expression and using the assumption $\alpha_K \le \Delta t$ if $\beta = 1$, we obtain

$$\begin{aligned}
&\frac{\Delta t}{4}\sum_{n=1}^{N-1}\left\|\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}\right\|_{L^2(\Omega)}^2 + \nu\left\|\nabla \boldsymbol{u}_h^N\right\|_{L^2(\Omega)}^2 + \sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla(p_h^2-p_h^1)\right\|_{L^2(K)}^2\\
&\le \nu\left\|\nabla \boldsymbol{u}_h^1\right\|_{L^2(\Omega)}^2 + \sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^1\right\|_{L^2(K)}^2 + \Delta t\sum_{n=1}^{N}\left(2\left\|\boldsymbol{f}(t^n)\right\|_{L^2(\Omega)}^2 + 5\left\|\boldsymbol{g}(t^n)\right\|_{L^2(\Omega)}^2\right)\\
&+\sum_{K\in\mathcal{T}_h}4\Delta t\,\beta\left\|\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right\|_{L^2(K)}^2 + \sum_{K\in\mathcal{T}_h}\frac{\alpha_K}{2}\left\|\nabla p_h^2\right\|_{L^2(K)}^2
\end{aligned} \tag{3.57}$$

and finally, writing $p_h^2 = p_h^2 - p_h^1 + p_h^1$ yields

$$\begin{aligned}
&\frac{\Delta t}{2}\sum_{n=1}^{N-1}\left\|\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n}{\Delta t}\right\|_{L^2(\Omega)}^2 + \nu\left\|\nabla \boldsymbol{u}_h^N\right\|_{L^2(\Omega)}^2 \le \nu\left\|\nabla \boldsymbol{u}_h^1\right\|_{L^2(\Omega)}^2 + 2\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla p_h^1\right\|_{L^2(K)}^2\\
&+\Delta t\sum_{n=1}^{N}\left(2\left\|\boldsymbol{f}(t^n)\right\|_{L^2(\Omega)}^2 + 5\left\|\boldsymbol{g}(t^n)\right\|_{L^2(\Omega)}^2\right) + \sum_{K\in\mathcal{T}_h}4\Delta t\,\beta\left\|\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right\|_{L^2(K)}^2.
\end{aligned} \tag{3.58}$$

Now we also need to bound $\Delta t\left\|\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right\|_{L^2(\Omega)}^2$. For this we take $n=1$ and $\boldsymbol{v}_h = \frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}$ in (3.35) which yields

$$\left\|\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right\|_{L^2(\Omega)}^2 + \nu\left(\nabla \boldsymbol{u}_h^1, \nabla\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right) - \frac{1}{\Delta t}\left(p_h^1, \nabla\cdot\boldsymbol{u}_h^1\right) = \left(\boldsymbol{f}(t^{n+1}), \frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right) + \frac{1}{\Delta t}\left(p_h^1, \nabla\cdot\boldsymbol{u}_h^0\right) \tag{3.59}$$

and using (3.36) with $q_h = p_h^1$ then gives

$$\begin{aligned}
&\left\|\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right\|_{L^2(\Omega)}^2 + \nu\left(\nabla \boldsymbol{u}_h^1, \nabla\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right) + \sum_{K\in\mathcal{T}_h}\frac{\alpha_K}{\Delta t}\left(\beta\,\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t} + \nabla p_h^1 - \boldsymbol{g}(t^1), \nabla p_h^1\right)_K\\
&= \left(\boldsymbol{f}(t^1), \frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right) + \frac{1}{\Delta t}\left(p_h^1, \nabla\cdot\boldsymbol{u}_h^0\right).
\end{aligned} \tag{3.60}$$

(3.60) then yields

$$\begin{aligned}
&\left\|\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right\|_{L^2(\Omega)}^2 + \frac{\nu\Delta t}{2}\left\|\nabla\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right\|_{L^2(\Omega)}^2 + \frac{\nu}{2\Delta t}\left\|\nabla \boldsymbol{u}_h^1\right\|_{L^2(\Omega)}^2\\
&+\sum_{K\in\mathcal{T}_h}\frac{\alpha_K}{\Delta t}\left\|\nabla p_h^1\right\|_{L^2(\Omega)}^2 + \sum_{K\in\mathcal{T}_h}\frac{\alpha_K}{\Delta t}\left(\beta\,\frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t} - \boldsymbol{g}(t^1), \nabla p_h^1\right)_K\\
&= \frac{\nu}{2\Delta t}\left\|\nabla \boldsymbol{u}_h^0\right\|_{L^2(\Omega)}^2 + \left(\boldsymbol{f}(t^1), \frac{\boldsymbol{u}_h^1-\boldsymbol{u}_h^0}{\Delta t}\right) + \frac{1}{\Delta t}\left(p_h^1, \nabla\cdot\boldsymbol{u}_h^0\right).
\end{aligned} \tag{3.61}$$

Using the assumption $\alpha_{max} \leq \Delta t$ if $\beta = 1$ with Cauchy Schwarz and Young inequalities, we have for any $\varepsilon > 0$

$$\begin{aligned}
&\Delta t \left\| \frac{\boldsymbol{u}_h^1 - \boldsymbol{u}_h^0}{\Delta t} \right\|_{L^2(\Omega)}^2 + \sum_{K \in \mathcal{T}_h} \alpha_K \left\| \nabla p_h^1 \right\|_{L^2(\Omega)}^2 + 2\nu \Delta t^2 \left\| \nabla \frac{\boldsymbol{u}_h^1 - \boldsymbol{u}_h^0}{\Delta t} \right\|_{L^2(\Omega)}^2 + 2\nu \left\| \nabla \boldsymbol{u}_h^1 \right\|_{L^2(\Omega)}^2 \\
&\leq 2\nu \left\| \nabla \boldsymbol{u}_h^0 \right\|_{L^2(\Omega)}^2 + 4\Delta t \left(\left\| \boldsymbol{f}(t^1) \right\|_{L^2(\Omega)}^2 + \alpha_{max} \left\| \boldsymbol{g}(t^1) \right\|_{L^2(\Omega)}^2 \right) + 2\varepsilon \left\| p_h^1 \right\|_{L^2(\Omega)}^2 + \frac{2}{\varepsilon} \left\| \nabla \cdot \boldsymbol{u}_h^0 \right\|_{L^2(\Omega)}^2 .
\end{aligned} \tag{3.62}$$

Adding (3.62) and (3.58) then yields

$$\begin{aligned}
&\Delta t \sum_{n=0}^{N-1} \left\| \frac{\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n}{\Delta t} \right\|_{L^2(\Omega)}^2 + 2\nu \left\| \nabla \boldsymbol{u}_h^N \right\|_{L^2(\Omega)}^2 + \sum_{K \in \mathcal{T}_h} \alpha_K \left\| \nabla p_h^1 \right\|_{L^2(\Omega)}^2 \leq 9\nu \left\| \nabla \boldsymbol{u}_h^0 \right\|_{L^2(\Omega)}^2 \\
&+ 23\Delta t \sum_{n=1}^{N} \left(\left\| \boldsymbol{f}(t^n) \right\|_{L^2(\Omega)}^2 + \alpha_{max} \left\| \boldsymbol{g}(t^n) \right\|_{L^2(\Omega)}^2 \right) + 9\varepsilon \left\| p_h^1 \right\|_{L^2(\Omega)}^2 + \frac{9}{\varepsilon} \left\| \nabla \cdot \boldsymbol{u}_h^0 \right\|_{L^2(\Omega)}^2 .
\end{aligned} \tag{3.63}$$

We use (3.42) and (3.63) in (3.47) to obtain the inequality

$$\begin{aligned}
\Delta t \sum_{n=1}^{N} \left\| p_h^n \right\|_{L^2(\Omega)}^2 \leq & C_3 \Bigg[9\nu \left\| \nabla \boldsymbol{u}_h^0 \right\|_{L^2(\Omega)}^2 + 9\varepsilon \left\| p_h^1 \right\|_{L^2(\Omega)}^2 + \frac{9}{\varepsilon} \left\| \nabla \cdot \boldsymbol{u}_h^0 \right\|_{L^2(\Omega)}^2 + \max(1, C_\alpha) \left\| \boldsymbol{u}_h^0 \right\|_{L^2(\Omega)}^2 \\
& + 23 \max(1, C_\alpha) \, \Delta t \sum_{n=1}^{N} \left((1 + \frac{C_p^2}{\nu}) \left\| \boldsymbol{f}(t^n) \right\|_{L^2(\Omega)}^2 + \alpha_{max} \left\| \boldsymbol{g}(t^n) \right\|_{L^2(\Omega)}^2 \right) \Bigg] .
\end{aligned} \tag{3.64}$$

Taking $\varepsilon = \frac{\Delta t}{18\, C_3}$ gives the stability estimate for the pressure. □

3.3 Convergence of the schemes for velocity and pressure for time-dependent Stokes

Similarly to [56, 58], we will use the following Ritz projections to prove convergence of the PSPG schemes. Let the Ritz projections $(\mathbf{R}_{h,\beta,\boldsymbol{k}}(\boldsymbol{u}, p), P_{h,\beta,\boldsymbol{k}}(\boldsymbol{u}, p)) \in V_h \times Q_h$ of any $(\boldsymbol{u}, p) \in V \times Q$ satisfy

$$\begin{aligned}
&\nu \left(\nabla \mathbf{R}_{h,\beta,\boldsymbol{k}}(\boldsymbol{u}, p), \nabla \boldsymbol{v}_h \right) - \left(P_{h,\beta,\boldsymbol{k}}(\boldsymbol{u}, p), \nabla \cdot \boldsymbol{v}_h \right) + \left(q_h, \nabla \cdot \mathbf{R}_{h,\beta,\boldsymbol{k}}(\boldsymbol{u}, p) \right) \\
&+ \sum_{K \in \mathcal{T}_h} \alpha_K \left(-\nu \Delta \mathbf{R}_{h,\beta,\boldsymbol{k}}(\boldsymbol{u}, p) + \nabla P_{h,\beta,\boldsymbol{k}}(\boldsymbol{u}, p), \nabla q_h \right)_K \\
&= \nu \left(\nabla \boldsymbol{u}, \nabla \boldsymbol{v}_h \right) - \left(p, \nabla \cdot \boldsymbol{v}_h \right) + \left(q_h, \nabla \cdot \boldsymbol{u} \right) \\
&+ \sum_{K \in \mathcal{T}_h} \alpha_K \left(\beta \left(-\nu \Delta \boldsymbol{u} + \nabla p \right) + (1 - \beta) \, \boldsymbol{k}, \nabla q_h \right)_K \quad \forall (\boldsymbol{v}_h, q_h) \in V_h \times Q_h
\end{aligned} \tag{3.65}$$

where $\boldsymbol{k} \in L^2(\Omega)$ is a given function with $\beta = 0, 1$ corresponding to non-consistent and consistent stabilizations respectively. It is important to note that the terms included in the projection differ depending on β.

We derive a priori error estimates for the Ritz projections as follows.

Lemma 2. *Let $\boldsymbol{u} \in V \cap [H^2(\Omega)]^3$ and $p \in Q \cap H^1(\Omega)$ and $(\boldsymbol{R}_{h,\beta,\boldsymbol{k}}(\boldsymbol{u},p), P_{h,\beta,\boldsymbol{k}}(\boldsymbol{u},p)) \in V_h \times Q_h$ their Ritz projections as defined above. Let $h = \max_{K\in\mathcal{T}_h} h_K$ and $\alpha_K > 0$, $K \in \mathcal{T}_h$. The following estimates hold*

$$\left(\nu \left|\boldsymbol{u} - \boldsymbol{R}_{h,\beta,\boldsymbol{k}}(\boldsymbol{u},p)\right|^2_{H^1(\Omega)} + \sum_{K\in\mathcal{T}_h} \alpha_K \left\|\nabla(p - P_{h,\beta,\boldsymbol{k}}(\boldsymbol{u},p))\right\|^2_{L^2(K)}\right)^{\frac{1}{2}}$$
$$\leq Ch\left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} + \beta\nu\frac{\alpha_{max}^{\frac{1}{2}}}{h}\right)|\boldsymbol{u}|_{H^2(\Omega)} + \left(\frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{h}\right)\left|p\right|_{H^1(\Omega)} + (1-\beta)\frac{\alpha_{max}^{\frac{1}{2}}}{h}\|\boldsymbol{k}\|_{L^2(\Omega)}\right]$$

$$\left\|p - P_{h,\beta,\boldsymbol{k}}(\boldsymbol{u},p)\right\|_{L^2(\Omega)} \leq Ch\max\left(\frac{h}{\alpha_{min}^{\frac{1}{2}}}, \nu^{\frac{1}{2}}\right)\left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} + \beta\nu\frac{\alpha_{max}^{\frac{1}{2}}}{h}\right)|\boldsymbol{u}|_{H^2(\Omega)}\right.$$
$$\left. + \left(\nu^{-\frac{1}{2}} + \frac{h}{\nu^{\frac{1}{2}}} + \alpha_{max}^{\frac{1}{2}}\right)\left|p\right|_{H^1(\Omega)} + (1-\beta)\frac{\alpha_{max}^{\frac{1}{2}}}{h}\|\boldsymbol{k}\|_{L^2(\Omega)}\right]$$

where C is a constant that depends on the domain Ω, but not on ν, $\boldsymbol{u}$, p, $\boldsymbol{k}$ or the stabilization parameter.

Remark 4. *The same estimates can be obtained for time-derivative of the velocity and pressure by differentiating the Ritz error equations* (3.68) *with respect to time and using time-differentiated test functions.*

Remark 5. *Estimates for a similar Ritz projection have been obtained in [104].*

Proof. Let us introduce $I_h\boldsymbol{u}$ the Lagrange interpolant of $\boldsymbol{u}$ in V_h and $J_h p$ the Clément interpolant of p on Q_h. We then define $\boldsymbol{E}_h$ and S_h as follows

$$\boldsymbol{u} - \mathbf{R}_{h,\beta,\boldsymbol{k}}(\boldsymbol{u},p) = \boldsymbol{u} - I_h\boldsymbol{u} + I_h\boldsymbol{u} - \mathbf{R}_{h,\beta,\boldsymbol{k}}(\boldsymbol{u},p) = \boldsymbol{u} - I_h\boldsymbol{u} + \boldsymbol{E}_h \tag{3.66}$$
$$p - P_{h,\beta,\boldsymbol{k}}(\boldsymbol{u},p) = p - J_h p + J_h p - P_{h,\beta,\boldsymbol{k}}(\boldsymbol{u},p) = p - J_h p + S_h \tag{3.67}$$

Injecting the interpolants into (3.65) and choosing $\boldsymbol{v}_h = \boldsymbol{E}_h$, $q_h = S_h$ then yields

$$\begin{aligned}\nu\|\nabla\boldsymbol{E}_h\|^2_{L^2(\Omega)} + \sum_{K\in\mathcal{T}_h}\alpha_K\|\nabla S_h\|^2_{L^2(K)} &= \nu(\nabla(I_h\boldsymbol{u}-\boldsymbol{u}),\nabla\boldsymbol{E}_h) - \left(J_h p - p, \nabla\cdot\boldsymbol{E}_h\right)\\ &+ (S_h, \nabla\cdot(I_h\boldsymbol{u}-\boldsymbol{u})) + \sum_{K\in\mathcal{T}_h}\alpha_K\left(\beta\nu\Delta\boldsymbol{u} + \nabla(J_h p - \beta p), \nabla S_h\right)_K\\ &+ (1-\beta)\sum_{K\in\mathcal{T}_h}\alpha_K(\boldsymbol{k},\nabla S_h)_K\end{aligned} \tag{3.68}$$

Estimating the terms in (3.68) using standard interpolation estimates, Young and Cauchy

Schwarz with one integration by parts, we get

$$\nu\left(\nabla(I_h\boldsymbol{u}-\boldsymbol{u}),\nabla\boldsymbol{E}_h\right) \le C\nu h^2\left|\boldsymbol{u}\right|^2_{H^2(\Omega)} + \frac{\nu}{4}\left|\boldsymbol{E}_h\right|^2_{H^1(\Omega)}$$
$$-\left(J_h p - p, \nabla\cdot\boldsymbol{E}_h\right) \le C\frac{h^2}{\nu}\left|p\right|^2_{H^1(\Omega)} + \frac{\nu}{4}\left|\boldsymbol{E}_h\right|^2_{H^1(\Omega)}$$
$$(S_h,\nabla\cdot(I_h\boldsymbol{u}-\boldsymbol{u})) = -\left(\nabla S_h, I_h\boldsymbol{u}-\boldsymbol{u}\right) \le \sum_{K\in\mathcal{T}_h}\frac{\alpha_K}{4}\left\|\nabla S_h\right\|^2_{L^2(K)} + C\frac{h^4}{\alpha_{min}}\left|\boldsymbol{u}\right|^2_{H^2(\Omega)}.$$

The two last terms in (3.68) gives the following estimate for $\beta = 0$

$$\sum_{K\in\mathcal{T}_h}\alpha_K\left(\nabla J_h p + \boldsymbol{k}, \nabla S_h\right)_K \le \sum_{K\in\mathcal{T}_h}\frac{\alpha_K}{4}\left\|\nabla S_h\right\|^2_{L^2(K)} + C\alpha_{max}\left(\left|p\right|^2_{H^1(\Omega)} + \left\|\boldsymbol{k}\right\|^2_{L^2(\Omega)}\right)$$

and for $\beta = 1$, we get

$$\sum_{K\in\mathcal{T}_h}\alpha_K\left(\nu\Delta\boldsymbol{u} + \nabla(J_h p - p), \nabla S_h\right)_K \le \sum_{K\in\mathcal{T}_h}\frac{\alpha_K}{4}\left\|\nabla S_h\right\|^2_{L^2(K)} + C\alpha_{max}\left(\nu^2\left|\boldsymbol{u}\right|^2_{H^2(\Omega)} + \left|p\right|^2_{H^1(\Omega)}\right).$$

Decomposing the error as above as in (3.66), (3.67), using the triangle inequality and combining the estimates yields the first error estimate.

Applying Lemma 1 to S_h, we get

$$\left\|S_h\right\|_{L^2(\Omega)} \le C\left(\sum_{K\in\mathcal{T}_h} h_K^2\left\|\nabla S_h\right\|^2_{L^2(K)}\right)^{1/2} + C\sup_{0\neq\boldsymbol{v}_h\in V_h}\frac{(S_h,\nabla\cdot\boldsymbol{v}_h)}{\left|\boldsymbol{v}_h\right|_{H^1(\Omega)}}.$$

The first term is estimated using the error estimate derived above and for the second we use the Ritz error equation (3.68) with $q_h = 0$. This yields

$$\frac{(S_h,\nabla\cdot\boldsymbol{v}_h)}{\left|\boldsymbol{v}_h\right|_{H^1(\Omega)}} \le \frac{\left(J_h p - p, \nabla\cdot\boldsymbol{v}_h\right) - \nu\left(\nabla\boldsymbol{E}_h,\nabla\boldsymbol{v}_h\right) + \nu\left(\nabla(I_h\boldsymbol{u}-\boldsymbol{u}),\nabla\boldsymbol{v}_h\right)}{\left|\boldsymbol{v}_h\right|_{H^1(\Omega)}}$$
$$\le C\left[h\left|p\right|_{H^1(\Omega)} + \nu\left|\boldsymbol{E}_h\right|_{H^1(\Omega)} + h\nu\left|\boldsymbol{u}\right|_{H^2(\Omega)}\right].$$

We then get

$$\left\|S_h\right\|_{L^2(\Omega)} \le C\left(\max\left(\frac{h}{\alpha_{min}^{\frac{1}{2}}}, \nu^{\frac{1}{2}}\right)\left(\nu\left|\boldsymbol{E}_h\right|^2_{H^1(\Omega)} + \sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla S_h\right\|^2_{L^2(K)}\right)^{\frac{1}{2}} + h\left|p\right|_{H^1(\Omega)} + h\nu\left|\boldsymbol{u}\right|_{H^2(\Omega)}\right).$$

Decomposing the error as in (3.66), (3.67), using the triangle inequality and the first error estimate yields the pressure error estimate. □

Theorem 2. *Let* $(\boldsymbol{u}, p)$ *be a solution of the time-dependent Stokes' equations* (3.1) *and* $(\boldsymbol{u}_h^n, p_h^n)$

given by (3.9) *for* $n \geq 0$. *Let*

$$\boldsymbol{r}_{h,\beta}(t) = \boldsymbol{R}_{h,\beta,\boldsymbol{f}(t)}(\boldsymbol{u}(t), p(t)) \quad \forall t \geq 0$$
$$s_{h,\beta}(t) = P_{h,\beta,\boldsymbol{f}(t)}(\boldsymbol{u}(t), p(t)) \quad \forall t \geq 0.$$

Assume that

$$\boldsymbol{u}, \frac{\partial \boldsymbol{u}}{\partial t} \in L^\infty(0,T;H^2(\Omega)),$$
$$p, \frac{\partial p}{\partial t} \in L^\infty(0,T;H^1(\Omega)),$$
$$\text{For } \beta = 0, \text{ we have } \boldsymbol{f}, \frac{\partial \boldsymbol{f}}{\partial t} \in L^\infty(0,T;L^2(\Omega)),$$
$$\frac{\partial^2 \boldsymbol{r}_{h,\beta}}{\partial t^2} \in L^2(0,T;L^2(\Omega)).$$

Assume that the stabilization parameters $\alpha_K > 0$ *are chosen such that there exists a constant* $C_\alpha > 0$ *such that for all* K *in* $\mathcal{T}_h$ *and* $h_K > 0$, α_K *satisfies* $h_K^2 \leq C_\alpha\, \alpha_K$.

If we assume also that the timestep Δt *satisfies the stability condition* $\alpha_{max} \leq \Delta t$ *for the consistent scheme* $\beta = 1$ *and that* $\boldsymbol{u}_h^0 = \boldsymbol{r}_{h,\beta}(t_0)$, *then the following a priori error estimate for the discrete in time norm of the velocity holds*

$$\begin{aligned}
&\nu \Delta t \sum_{n=1}^{N} \left|\boldsymbol{u}(t_n) - \boldsymbol{u}_h^n\right|_{H^1(\Omega)}^2 \\
&\leq C \frac{T}{\nu^2} h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} + \beta \nu \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \left(\nu \|\boldsymbol{u}\|_{L^\infty(0,T;H^2(\Omega))}^2 + \left\| \frac{\partial \boldsymbol{u}}{\partial t} \right\|_{L^\infty(0,T;H^2(\Omega))}^2 \right) \right. \\
&+ \left(\frac{h}{\nu^{\frac{1}{2}}} + \alpha_{max}^{\frac{1}{2}} \right)^2 \left(\nu \|p\|_{L^\infty(0,T;H^1(\Omega))}^2 + \left\| \frac{\partial p}{\partial t} \right\|_{L^\infty(0,T;H^1(\Omega))}^2 \right) \\
&\left. + (1-\beta) \frac{\alpha_{max}}{h^2} \left(\nu \|\boldsymbol{f}\|_{L^\infty(0,T;L^2(\Omega))}^2 + \left\| \frac{\partial \boldsymbol{f}}{\partial t} \right\|_{L^\infty(0,T;L^2(\Omega))}^2 \right) \right] \\
&+ C \Delta t^2 \left\| \frac{\partial^2 \boldsymbol{r}_{h,\beta}}{\partial t^2} \right\|_{L^2(0,T;L^2(\Omega))}^2 .
\end{aligned}$$

Assuming that the stabilization parameter is $\alpha_K = C\dfrac{h^2}{\nu}$, *we get the optimal error estimate*

$$\Delta t \sum_{n=1}^{N} \left|\boldsymbol{u}(t_n) - \boldsymbol{u}_h^n\right|_{H^1(\Omega)}^2 \leq C_{\boldsymbol{u},p,\boldsymbol{f},T} \left((\nu^{-1} + h^2 \nu^{-3} + (1-\beta)\nu^{-3}) h^2 + \Delta t^2 \right)$$

where $C_{\boldsymbol{u},p,\boldsymbol{f},T}$ *is independent of* h, Δt.

Remark 6. *The term involving* $\boldsymbol{r}_{h,\beta}$ *can be estimated by making the Ritz projection error* $\boldsymbol{u} - \boldsymbol{r}_{h,\beta}$ *appear and using the triangle inequality along with the error bound in Lemma 2 with twice*

derived quantities. We then obtain a $\dfrac{\partial^2 \boldsymbol{u}}{\partial t^2}$ *term along with error terms small with respect to* h *and* Δt *assuming sufficient regularity of the data.*

Proof. Let us introduce the following notations for $n \geq 0$

$$\begin{aligned}
\boldsymbol{\eta}_h^n &= \boldsymbol{u}(t_n) - \boldsymbol{r}_{h,\beta}(t_n) \\
\boldsymbol{\theta}_h^n &= \boldsymbol{r}_{h,\beta}(t_n) - \boldsymbol{u}_h(t_n) \\
\zeta_h^n &= p(t_n) - s_{h,\beta}(t_n) \\
\xi_h^n &= s_{h,\beta}(t_n) - p_h(t_n)
\end{aligned}$$

such that the errors are decomposed as

$$\boldsymbol{u}(t_n) - \boldsymbol{u}_h^n = \boldsymbol{\eta}_h^n + \boldsymbol{\theta}_h^n, \qquad p(t_n) - p_h^n = \zeta_h^n + \xi_h^n.$$

Using (3.2), (3.3) and (3.9), we obtain

$$\begin{aligned}
&\left(\frac{1}{\Delta t}\left(\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n\right), \boldsymbol{v}_h\right) + \nu\left(\nabla \boldsymbol{u}_h^{n+1}, \nabla \boldsymbol{v}_h\right) - \left(p_h^{n+1}, \nabla \cdot \boldsymbol{v}_h\right) + \left(q_h, \nabla \cdot \boldsymbol{u}_h^{n+1}\right) \\
&+ \sum_{K \in \mathcal{T}_h} \alpha_K \left(\frac{\beta}{\Delta t}\left(\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n\right) - \nu \Delta \boldsymbol{u}_h^{n+1} + \nabla p_h^{n+1}, \nabla q_h\right)_K \\
&= \left(\frac{\partial \boldsymbol{u}}{\partial t}(t_{n+1}), \boldsymbol{v}_h\right) + \nu\left(\nabla \boldsymbol{u}(t_{n+1}), \nabla \boldsymbol{v}_h\right) \\
&+ \left(p(t_{n+1}), \nabla \cdot \boldsymbol{v}_h\right) - \left(q_h, \nabla \cdot \boldsymbol{u}(t_{n+1})\right) \\
&+ \sum_{K \in \mathcal{T}_h} \alpha_K \left(\frac{\partial \boldsymbol{u}}{\partial t}(t_{n+1}) - \nu \Delta \boldsymbol{u}(t_{n+1}) + \nabla p(t_{n+1}), \nabla q_h\right)_K
\end{aligned} \tag{3.69}$$

By adding the missing terms on both sides of the equality, we derive a modified Galerkin orthogonality accounting for the lack of consistency in the time derivative. We get

$$\begin{aligned}
&\left(\frac{1}{\Delta t}\left(\boldsymbol{\theta}_h^{n+1} - \boldsymbol{\theta}_h^n\right), \boldsymbol{v}_h\right) + \nu\left(\nabla \boldsymbol{\theta}_h^{n+1}, \nabla \boldsymbol{v}_h\right) - \left(\xi_h^{n+1}, \nabla \cdot \boldsymbol{v}_h\right) + \left(q_h, \nabla \cdot \boldsymbol{\theta}_h^{n+1}\right) \\
&+ \sum_{K \in \mathcal{T}_h} \alpha_K \left(\frac{\beta}{\Delta t}\left(\boldsymbol{\theta}_h^{n+1} - \boldsymbol{\theta}_h^n\right) - \nu \Delta \boldsymbol{\theta}_h^{n+1} + \nabla \xi_h^{n+1}, \nabla q_h\right)_K \\
&= \left(\frac{1}{\Delta t}\left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n)\right) - \frac{\partial \boldsymbol{u}}{\partial t}(t_{n+1}), \boldsymbol{v}_h\right) - \nu\left(\nabla \boldsymbol{\eta}_h^{n+1}, \nabla \boldsymbol{v}_h\right) \\
&+ \left(\zeta_h^{n+1}, \nabla \cdot \boldsymbol{v}_h\right) - \left(q_h, \nabla \cdot \boldsymbol{\eta}_h^{n+1}\right) \\
&+ \sum_{K \in \mathcal{T}_h} \alpha_K \left(\frac{\beta}{\Delta t}\left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n)\right) - \frac{\partial \boldsymbol{u}}{\partial t}(t_{n+1}) + \nu \Delta \boldsymbol{\eta}_h^{n+1} - \nabla \zeta_h^{n+1}, \nabla q_h\right)_K
\end{aligned} \tag{3.70}$$

For $\beta = 0$, using (3.65) in (3.70) we get

$$\begin{aligned}
&\left(\frac{1}{\Delta t}\left(\boldsymbol{\theta}_h^{n+1} - \boldsymbol{\theta}_h^n\right), \boldsymbol{v}_h\right) + \nu\left(\nabla\boldsymbol{\theta}_h^{n+1}, \nabla\boldsymbol{v}_h\right) - \left(\xi_h^{n+1}, \nabla\cdot\boldsymbol{v}_h\right) + \left(q_h, \nabla\cdot\boldsymbol{\theta}_h^{n+1}\right)\\
&+ \sum_{K\in\mathcal{T}_h} \alpha_K \left(\frac{\beta}{\Delta t}\left(\boldsymbol{\theta}_h^{n+1} - \boldsymbol{\theta}_h^n\right) - \nu\Delta\boldsymbol{\theta}_h^{n+1} + \nabla\xi_h^{n+1}, \nabla q_h\right)_K\\
&= \left(\frac{1}{\Delta t}\left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n)\right) - \frac{\partial\boldsymbol{u}}{\partial t}(t_{n+1}), \boldsymbol{v}_h\right)\\
&+ \sum_{K\in\mathcal{T}_h} \alpha_K \left(-\frac{\partial\boldsymbol{u}}{\partial t}(t_{n+1}) + \nu\Delta\boldsymbol{u}(t_{n+1}) - \nabla p(t_{n+1}) + \boldsymbol{f}(t_{n+1}), \nabla q_h\right)_K\\
&= \left(\frac{1}{\Delta t}\left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n)\right) - \frac{\partial\boldsymbol{u}}{\partial t}(t_{n+1}), \boldsymbol{v}_h\right) \qquad (3.71)
\end{aligned}$$

Note that we have replaced $\boldsymbol{k}$ by $\boldsymbol{f}(t^{n+1})$ in (3.65) since $\boldsymbol{r}_{h,\beta}(t) = \mathbf{R}_{h,\beta,\boldsymbol{f}(t)}(\boldsymbol{u}(t), p(t)) \quad \forall t \geq 0$. Thus in the case $\beta = 0$, $(\boldsymbol{\theta}_h^{n+1}, \xi_h^{n+1})$ is solution of (3.35) and (3.36) with

$$\boldsymbol{f}(t^{n+1}) = \frac{1}{\Delta t}\left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n)\right) - \frac{\partial\boldsymbol{u}}{\partial t}(t^{n+1})$$

and

$$\boldsymbol{g}(t_{n+1}) = 0$$

for $n \geq 0$ with $\boldsymbol{\theta}_h^0 = \boldsymbol{r}_{h,\beta}(t_0) - \boldsymbol{u}_h(t_0)$. We can therefore apply Theorem 1 and get

$$\begin{aligned}
&\left\|\boldsymbol{\theta}_h^N\right\|_{L^2(\Omega)}^2 + \nu\Delta t\sum_{n=1}^{N}\left\|\nabla\boldsymbol{\theta}_h^n\right\|_{L^2(\Omega)}^2 + \frac{\Delta t}{2}\sum_{n=1}^{N}\sum_{K\in\mathcal{T}_h}\alpha_K\left\|\nabla\xi_h^n\right\|_{L^2(K)}^2\\
&\leq \left\|\boldsymbol{\theta}_h^0\right\|_{L^2(\Omega)}^2 + \Delta t\sum_{n=0}^{N-1}\left(\frac{C_p^2}{\nu}\left\|\frac{1}{\Delta t}\left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n)\right) - \frac{\partial\boldsymbol{u}}{\partial t}(t_{n+1})\right\|_{L^2(\Omega)}^2\right). \qquad (3.72)
\end{aligned}$$

We then inject $-\frac{\partial\boldsymbol{r}_{h,\beta}}{\partial t}(t_{n+1}) + \frac{\partial\boldsymbol{r}_{h,\beta}}{\partial t}(t_{n+1})$ into (3.72) and estimate the terms separately using the triangle inequality, similarly to the analysis of SUPG stabilizations for evolutionary convection-diffusion-reaction equations in [105]. With Taylor's formula with integral remainder and Cauchy Schwarz, we get

$$\begin{aligned}
\left\|\frac{\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n)}{\Delta t} - \frac{\partial\boldsymbol{r}_{h,\beta}}{\partial t}(t_{n+1})\right\|_{L^2(\Omega)}^2 &\leq \frac{1}{\Delta t^2}\left\|\int_{t_n}^{t_{n+1}}(t-t_n)\frac{\partial^2\boldsymbol{r}_{h,\beta}}{\partial t^2}(t)\,dt\right\|_{L^2(\Omega)}^2\\
&\leq \frac{1}{\Delta t^2}\left(\left(\int_{t_n}^{t_{n+1}}(t-t_n)^2\,dt\right)^{\frac{1}{2}}\left(\int_{t_n}^{t_{n+1}}\left\|\frac{\partial^2\boldsymbol{r}_{h,\beta}}{\partial t^2}\right\|_{L^2(\Omega)}^2 dt\right)^{\frac{1}{2}}\right)^2\\
&\leq C\Delta t\int_{t_n}^{t_{n+1}}\left\|\frac{\partial^2\boldsymbol{r}_{h,\beta}}{\partial t^2}\right\|_{L^2(\Omega)}^2 dt. \qquad (3.73)
\end{aligned}$$

An estimate for $\left\| \frac{\partial \boldsymbol{r}_{h,\beta}}{\partial t}(t_{n+1}) - \frac{\partial \boldsymbol{u}}{\partial t}(t_{n+1}) \right\|^2_{L^2(\Omega)}$ can be recovered using the same reasoning as used in Lemma 2 with quantities differentiated in time. Taking the time derivative of (3.65) and proceeding as in Lemma 2, we obtain

$$\begin{aligned}
&\left\| \frac{\partial \boldsymbol{r}_{h,\beta}}{\partial t}(t_{n+1}) - \frac{\partial \boldsymbol{u}}{\partial t}(t_{n+1}) \right\|^2_{L^2(\Omega)} \\
&\leq \frac{C}{\nu} h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} \right)^2 \left| \frac{\partial \boldsymbol{u}}{\partial t}(t_{n+1}) \right|^2_{H^2(\Omega)} + \left(\frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \left| \frac{\partial p}{\partial t}(t_{n+1}) \right|^2_{H^1(\Omega)} \right. \\
&\left. + \frac{\alpha_{max}}{h^2} \left\| \frac{\partial \boldsymbol{f}}{\partial t}(t_{n+1}) \right\|^2_{L^2(\Omega)} \right]. \qquad (3.74)
\end{aligned}$$

Combining the two previous estimates, we then get

$$\begin{aligned}
&\frac{\Delta t}{\nu} \sum_{n=0}^{N-1} \left(\left\| \frac{1}{\Delta t} \left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n) \right) - \frac{\partial \boldsymbol{u}}{\partial t}(t_{n+1}) \right\|^2_{L^2(\Omega)} \right) \\
&\leq C \frac{T}{\nu^2} h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} \right)^2 \left\| \frac{\partial \boldsymbol{u}}{\partial t} \right\|^2_{L^\infty(0,T;H^2(\Omega))} + \left(\frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{2} \right)^2 \left\| \frac{\partial p}{\partial t} \right\|^2_{L^\infty(0,T;H^1(\Omega))} \right. \\
&\left. + \frac{\alpha_{max}}{h^2} \left\| \frac{\partial \boldsymbol{f}}{\partial t} \right\|^2_{L^\infty(0,T;L^2(\Omega))} \right] + C\Delta t^2 \left\| \frac{\partial^2 \boldsymbol{r}_{h,\beta}}{\partial t^2} \right\|^2_{L^2(0,T;L^2(\Omega))}. \qquad (3.75)
\end{aligned}$$

Now, we can estimate the error on $\boldsymbol{u} - \boldsymbol{u}_h$ by inserting $-\boldsymbol{r}_{h,\beta} + \boldsymbol{r}_{h,\beta}$ and using the triangle inequality, velocity estimate from Lemma 2 and (3.75). It follows that

$$\begin{aligned}
&\nu \Delta t \sum_{n=1}^{N} \left| \boldsymbol{u}(t_n) - \boldsymbol{u}_h^n \right|^2_{H^1(\Omega)} \\
&\leq C \frac{T}{\nu^2} h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} \right)^2 \left(\nu \left\| \boldsymbol{u} \right\|^2_{L^\infty(0,T;H^2(\Omega))} + \left\| \frac{\partial \boldsymbol{u}}{\partial t} \right\|^2_{L^\infty(0,T;H^2(\Omega))} \right) \right. \\
&+ \left(\frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{2} \right)^2 \left(\nu \left\| p \right\|^2_{L^\infty(0,T;H^1(\Omega))} + \left\| \frac{\partial p}{\partial t} \right\|^2_{L^\infty(0,T;H^1(\Omega))} \right) \\
&\left. + \frac{\alpha_{max}}{h^2} \left(\nu \left\| \boldsymbol{f} \right\|^2_{L^\infty(0,T;L^2(\Omega))} + \left\| \frac{\partial \boldsymbol{f}}{\partial t} \right\|^2_{L^\infty(0,T;L^2(\Omega))} \right) \right] \\
&+ C\Delta t^2 \left\| \frac{\partial^2 \boldsymbol{r}_{h,\beta}}{\partial t^2} \right\|^2_{L^2(0,T;L^2(\Omega))}. \qquad (3.76)
\end{aligned}$$

We do a similar analysis for the case $\beta = 1$. This time, $(\boldsymbol{\theta}_h^{n+1}, \xi_h^{n+1})$ satisfy the discretized time-dependent Stokes equations (3.35) and (3.36) with

$$\boldsymbol{f}(t^{n+1}) = \frac{1}{\Delta t} \left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n) \right) - \frac{\partial \boldsymbol{u}}{\partial t}(t^{n+1})$$

and

$$\boldsymbol{g}(t_{n+1}) = \frac{1}{\Delta t}\left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n)\right) - \frac{\partial \boldsymbol{u}}{\partial t}(t^{n+1}).$$

Following the same analysis as for $\beta = 0$ leads to the estimate

$$\begin{aligned}
&\nu\Delta t \sum_{n=1}^{N} \left|\boldsymbol{u}(t_n) - \boldsymbol{u}_h^n\right|^2_{H^1(\Omega)} \\
&\leq C\frac{T}{\nu^2}h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} + \nu\frac{\alpha_{max}^{\frac{1}{2}}}{h}\right)^2 \left(\nu\|\boldsymbol{u}\|^2_{L^\infty(0,T;H^2(\Omega))} + \left\|\frac{\partial \boldsymbol{u}}{\partial t}\right\|^2_{L^\infty(0,T;H^2(\Omega))}\right)\right. \\
&\left. + \left(\frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{h}\right)^2 \left(\nu\|p\|^2_{L^\infty(0,T;H^1(\Omega))} + \left\|\frac{\partial p}{\partial t}\right\|^2_{L^\infty(0,T;H^1(\Omega))}\right)\right] + C\Delta t^2 \left\|\frac{\partial^2 \boldsymbol{r}_{h,\beta}}{\partial t^2}\right\|^2_{L^2(0,T;L^2(\Omega))}
\end{aligned} \tag{3.77}$$

which completes the proof. □

Theorem 3. *Let* $(\boldsymbol{u}, p)$ *be a solution of the time-dependent Stokes' equations* (3.1) *and* $(\boldsymbol{u}_h^n, p_h^n)$ *given by* (3.9) *for* $n \geq 0$. *Let*

$$\begin{aligned}
\boldsymbol{r}_{h,\beta}(t) &= \boldsymbol{R}_{h,\beta,\boldsymbol{f}(t)}(\boldsymbol{u}(t), p(t)) \quad \forall t \geq 0 \\
s_{h,\beta}(t) &= P_{h,\beta,\boldsymbol{f}(t)}(\boldsymbol{u}(t), p(t)) \quad \forall t \geq 0.
\end{aligned}$$

Assume that

$$\begin{aligned}
&\boldsymbol{u}, \frac{\partial \boldsymbol{u}}{\partial t} \in L^\infty(0,T;H^2(\Omega)), \\
&p, \frac{\partial p}{\partial t} \in L^\infty(0,T;H^1(\Omega)), \\
&\text{For } \beta = 0, \text{ we have } \boldsymbol{f} \in L^\infty(0,T;H^1(\Omega) \text{ and }) \frac{\partial \boldsymbol{f}}{\partial t} \in L^\infty(0,T;L^2(\Omega),) \\
&\frac{\partial^2 \boldsymbol{r}_{h,\beta}}{\partial t^2} \in L^2(0,T;L^2(\Omega)).
\end{aligned}$$

Assume that the stabilization parameters $\alpha_K > 0$ *are chosen such that there exists a constant* $C_\alpha > 0$ *such that for all* K *in* $\mathcal{T}_h$ *and* $h_K > 0$, α_K *satisfies* $h_K^2 \leq C_\alpha\ \alpha_K$.

Assume also that the timestep $0 < \Delta t < 1$ *satisfies the stability condition* $\alpha_{max} \leq \Delta t$ *for the consistent scheme* $\beta = 1$ *and that* $\boldsymbol{u}_h^0 = \boldsymbol{r}_{h,\beta}(t_0)$, *then the following a priori error estimate for the*

discrete in time norm of the pressure holds

$$
\begin{aligned}
&\Delta t \sum_{n=1}^{N} \left\| p(t_n) - p_h^n \right\|_{L^2(\Omega)}^2 \\
&\leq C \max(1, C_\alpha)(1 + \frac{C_p^2}{\nu}) \frac{T}{\nu} h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} + \beta\nu \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \left\| \frac{\partial \boldsymbol{u}}{\partial t} \right\|_{L^\infty(0,T;H^2(\Omega))}^2 \right. \\
&+ \left(\frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \left\| \frac{\partial p}{\partial t} \right\|_{L^\infty(0,T;H^1(\Omega))}^2 \\
&\left. + (1-\beta) \frac{\alpha_{max}}{h^2} \left(\left\| \frac{\partial p}{\partial t} \right\|_{L^\infty(0,T;H^1(\Omega))}^2 + \left\| \frac{\partial \boldsymbol{f}}{\partial t} \right\|_{L^\infty(0,T;L^2(\Omega))}^2 \right) \right] \\
&+ CT \max\left(\frac{h^2}{\alpha_{min}}, \nu \right) h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} + \beta\nu \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \|\boldsymbol{u}\|_{L^\infty(0,T;H^2(\Omega))}^2 \right. \\
&\left. + \left(\nu^{-\frac{1}{2}} + \frac{h}{\nu^{\frac{1}{2}}} + \alpha_{max}^{\frac{1}{2}} \right)^2 \|p\|_{L^\infty(0,T;H^1(\Omega))}^2 + (1-\beta) \frac{\alpha_{max}}{h^2} \|\boldsymbol{f}\|_{L^\infty(0,T;H^1(\Omega))}^2 \right] \\
&+ C\Delta t^2 \left\| \frac{\partial^2 \boldsymbol{r}_{h,\beta}}{\partial t^2} \right\|_{L^2(0,T;L^2(\Omega))}^2 .
\end{aligned}
$$

Assuming that the stabilization parameter is $\alpha_K = C\frac{h^2}{\nu}$, *we get the optimal error estimate*

$$
\Delta t \sum_{n=1}^{N} \left\| p(t_n) - p_h^n \right\|_{L^2(\Omega)}^2 \leq C_{\boldsymbol{u},p,\boldsymbol{f},T,\nu} \left(\left(1 + \nu^2 + \max(1, C_\alpha)(1+\nu^{-1})(1+\nu^{-2})\right) h^2 + \Delta t^2 \right).
$$

Remark 7. *Again as mentioned in Remark 6 the term involving* $\boldsymbol{r}_{h,\beta}$ *can be estimated by the norm of* $\frac{\partial^2 \boldsymbol{u}}{\partial t^2}$ *term along with error terms small with respect to* h *and* Δt *assuming sufficient regularity of the data.*

Proof. Assume first that $\beta = 0$. Proceeding as in the proof for velocity convergence, $(\boldsymbol{\theta}_h^{n+1}, \xi_h^{n+1})$ satisfy the discretized time-dependent Stokes equations (3.35) and (3.36) with

$$
\boldsymbol{f}(t^{n+1}) = \frac{1}{\Delta t} \left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n) \right) - \frac{\partial \boldsymbol{u}}{\partial t}(t^{n+1})
$$

and

$$
\boldsymbol{g}(t_{n+1}) = 0
$$

for $n \geq 0$ with $\boldsymbol{\theta}_h^0 = \boldsymbol{r}_{h,\beta}(t_0) - \boldsymbol{u}_h(t_0)$. Using the stability result from Theorem 1, we get

$$\begin{aligned}
&\Delta t \sum_{n=1}^{N} \|\xi_h^n\|_{L^2(\Omega)}^2 \\
&\leq C \left[\frac{C_\alpha}{\Delta t} \|\nabla \cdot \boldsymbol{\theta}_h^0\|_{L^2(\Omega)}^2 + \max(1, C_\alpha) \|\boldsymbol{\theta}_h^0\|_{L^2(\Omega)}^2 + \nu \|\nabla \boldsymbol{\theta}_h^0\|_{L^2(\Omega)}^2 \right. \\
&\left. + \max(1, C_\alpha) \left((1 + \frac{C_p^2}{\nu}) \, \Delta t \sum_{n=0}^{N-1} \left\| \frac{1}{\Delta t} \left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n) \right) - \frac{\partial \boldsymbol{u}}{\partial t}(t^{n+1}) \right\|_{L^2(\Omega)}^2 \right) \right].
\end{aligned}$$

Using the assumption $\boldsymbol{u}_h^0 = \boldsymbol{r}_{h,\beta}(t_0)$ and (3.75), we obtain

$$\begin{aligned}
&\Delta t \sum_{n=1}^{N} \|\xi_h^n\|_{L^2(\Omega)}^2 \\
&\leq C \max(1, C_\alpha)(1 + \frac{C_p^2}{\nu}) \, \frac{T}{\nu} h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} \right)^2 \left\| \frac{\partial \boldsymbol{u}}{\partial t} \right\|_{L^\infty(0,T;H^2(\Omega))}^2 + \left(\frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \left\| \frac{\partial p}{\partial t} \right\|_{L^\infty(0,T;H^1(\Omega))}^2 \right. \\
&\left. + \frac{\alpha_{max}}{h^2} \left\| \frac{\partial \boldsymbol{f}}{\partial t} \right\|_{L^\infty(0,T;L^2(\Omega))}^2 \right] + C \Delta t^2 \left\| \frac{\partial^2 \boldsymbol{r}_{h,\beta}}{\partial t^2} \right\|_{L^2(0,T;L^2(\Omega))}^2
\end{aligned}$$

which by injecting the Ritz projected pressure, using the triangular inequality and the pressure error for the Ritz projection in Lemma 2 gives

$$\begin{aligned}
&\Delta t \sum_{n=1}^{N} \|p(t_n) - p_h^n\|_{L^2(\Omega)}^2 \\
&\leq C \max(1, C_\alpha)(1 + \frac{C_p^2}{\nu}) \, \frac{T}{\nu} h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} \right)^2 \left\| \frac{\partial \boldsymbol{u}}{\partial t} \right\|_{L^\infty(0,T;H^2(\Omega))}^2 + \left(\frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{t} \right)^2 \left\| \frac{\partial p}{\partial t} \right\|_{L^\infty(0,T;H^1(\Omega))}^2 \right. \\
&\left. + \frac{\alpha_{max}}{h^2} \left(\left\| \frac{\partial p}{\partial t} \right\|_{L^\infty(0,T;H^1(\Omega))}^2 + \left\| \frac{\partial \boldsymbol{f}}{\partial t} \right\|_{L^\infty(0,T;L^2(\Omega))}^2 \right) \right] \\
&+ CT \max\left(\frac{h^2}{\alpha_{min}}, \nu \right) h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} + \nu \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \|\boldsymbol{u}\|_{L^\infty(0,T;H^2(\Omega))}^2 \right. \\
&\left. + \left(\nu^{-\frac{1}{2}} \frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \|p\|_{L^\infty(0,T;H^1(\Omega))}^2 + \frac{\alpha_{max}}{h^2} \|\boldsymbol{f}\|_{L^\infty(0,T;H^1(\Omega))}^2 \right] \\
&+ C \Delta t^2 \left\| \frac{\partial^2 \boldsymbol{r}_{h,\beta}}{\partial t^2} \right\|_{L^2(0,T;L^2(\Omega))}^2
\end{aligned} \tag{3.78}$$

Assume now $\beta = 1$. Proceeding as in the proof for velocity convergence, $(\boldsymbol{\theta}_h^{n+1}, \xi_h^{n+1})$ satisfy the discretized time-dependent Stokes equations (3.35) and (3.36) with

$$\boldsymbol{f}(t^{n+1}) = \frac{1}{\Delta t} \left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n) \right) - \frac{\partial \boldsymbol{u}}{\partial t}(t^{n+1})$$

and

$$\boldsymbol{g}(t_{n+1}) = \frac{1}{\Delta t}\left(\boldsymbol{r}_{h,\beta}(t_{n+1}) - \boldsymbol{r}_{h,\beta}(t_n)\right) - \frac{\partial \boldsymbol{u}}{\partial t}(t^{n+1})$$

for $n \geq 0$ with $\boldsymbol{\theta}_h^0 = \boldsymbol{r}_{h,\beta}(t_0) - \boldsymbol{u}_h(t_0)$. Using the same analysis as in (3.79), we get

$$\begin{aligned}
&\Delta t \sum_{n=1}^{N} \left\| p(t_n) - p_h^n \right\|_{L^2(\Omega)}^2 \\
&\leq C \max(1, C_\alpha)(1 + \frac{C_p^2}{\nu})\, \frac{T}{\nu} h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} + \nu \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \left\| \frac{\partial \boldsymbol{u}}{\partial t} \right\|_{L^\infty(0,T;H^2(\Omega))}^2 \right. \\
&\left. + \left(\frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \left\| \frac{\partial p}{\partial t} \right\|_{L^\infty(0,T;H^1(\Omega))}^2 \right] \\
&+ CT \max\left(\frac{h^2}{\alpha_{min}}, \nu \right) h^2 \left[\left(\nu^{\frac{1}{2}} + \frac{h}{\alpha_{min}^{\frac{1}{2}}} + \nu \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \|\boldsymbol{u}\|_{L^\infty(0,T;H^2(\Omega))}^2 \right. \\
&\left. + \left(\nu^{-\frac{1}{2}} + \frac{h}{\nu^{\frac{1}{2}}} + \frac{\alpha_{max}^{\frac{1}{2}}}{h} \right)^2 \|p\|_{L^\infty(0,T;H^1(\Omega))}^2 \right] \\
&+ C\Delta t^2 \left\| \frac{\partial^2 \boldsymbol{r}_{h,\beta}}{\partial t^2} \right\|_{L^2(0,T;L^2(\Omega))}^2
\end{aligned} \tag{3.79}$$

which completes the proof. □

Remark 8. *If L^2 estimates for the Ritz projection errors had been derived instead of H^1 estimates, sharper bounds for small values of ν could be obtained. Indeed, given L^2 estimates, it would not have been necessary to use Poincaré and dividing by the viscosity ν in* (3.74).

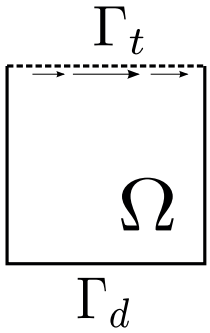


Figure 3.1: Sketch of the lid-driven cavity test case

3.4 Analysis of stabilization on startup of 2D lid-driven cavity with regularized time-varying tangential velocity at the boundary

The lid-driven cavity is a classical test case for the Stokes and Navier-Stokes equations, see for instance [106]. We consider a regularized version of the lid-driven cavity such that velocity field discontinuities in the corners are eliminated by changing the boundary tangential velocity profile. The tangential velocity is set to increase from zero with respect to time. We only consider the startup of the simulation in order to evaluate performance of several stabilization schemes in a non-stationary context.

The test case is illustrated in Figure 3.1 where Ω is the computational domain, Γ_t is the boundary where tangential velocity is imposed and Γ_d is boundary where an homogenous Dirichlet boundary condition will be imposed. Let the computational domain be $\Omega = [0,1] \times [0,1]$ and $T = 0.01$ the final simulation time. We choose a relatively small kinematic viscosity $\nu = 10^{-6}\ m^2/s$ which corresponds to the kinematic viscosity of water in order to illustrate differences between the stabilization schemes with application to hydrodynamics in mind. Note that although we solve the rescaled Stokes equations formulations involving the kinematic viscosity ν, we will plot the unscaled pressure which is given by ρp where p is pressure given by the scaled equations (3.1) and $\rho = 1000$.

Let us define $\boldsymbol{u}_t : (0,1) \times [0,T] \to \mathbb{R}^2$ as

$$\boldsymbol{u}_t(x,t) = \begin{pmatrix} \dfrac{16x^2(1-x)^2 10t}{1+10t} \\ 0 \end{pmatrix} \quad \forall x \in (0,1),\ \forall t \in [0,T]. \tag{3.80}$$

We then solve the time-dependent homogenous Stokes problem with the boundary conditions and initial condition are given as follows

$$\begin{cases} \boldsymbol{u} = 0 & \text{on } \Gamma_d \times [0,T], \\ \boldsymbol{u} = \boldsymbol{u}_t & \text{on } \Gamma_t \times [0,T], \\ \boldsymbol{u}(0,\cdot) = 0 & \text{in } \Omega. \end{cases} \tag{3.81}$$

Since the pressure is determined up to an additive constant, we fix the pressure to zero at the origin. We will first qualitatively study the behaviour of the different stabilization schemes with different stabilization parameters, with different mesh sizes and timesteps. We choose four different mesh sizes $H = 0.005,\ 0.01,\ 0.02,\ 0.1$ and a timestep of $\Delta t = 1e-4$. Note that for numerical experiments, we will use the notation H for tetrahedral mesh size. For these values, the error is dominated by the error in space so we should observe convergence. Note for all mesh sizes but the smallest, we have $\Delta t \leq H^2$, which means we are considering the case of a small timestep. Although it is not as small as timesteps for which instabilities appear for the velocity in [57], it is interesting to see how the different methods compare when slightly deviating away from the stability condition $\Delta t \geq H^2$.

To get an accurate approximation of the solution at $T = 0.01$, we compare the different stabilizations with the unconditionally stable $\mathbb{P}_2 - \mathbb{P}_1$ finite element solution with $H = 0.01$ and $\Delta t = 1e-4$. We consider the bubble enrichment scheme and local pressure projection scheme, which do not have any stabilization parameter to set, along with four different PSPG stabilizations, $\left(\beta = 0, \gamma = 0\right)$ and $\left(\beta = 1, \gamma \in \left\{0, \frac{1}{\Delta t}, -\frac{1}{\Delta t}\right\}\right)$. Recall that the spatial stabilization parameter is defined as

$$\alpha_K = \bar{\alpha}\frac{H_K^2}{\nu} \tag{3.82}$$

and the transient stabilization parameter as

$$\alpha_K = \bar{\alpha}\frac{\Delta t\dfrac{H_K^2}{\nu}}{\dfrac{H_K^2}{\nu} + \underline{\alpha}\Delta t} \tag{3.83}$$

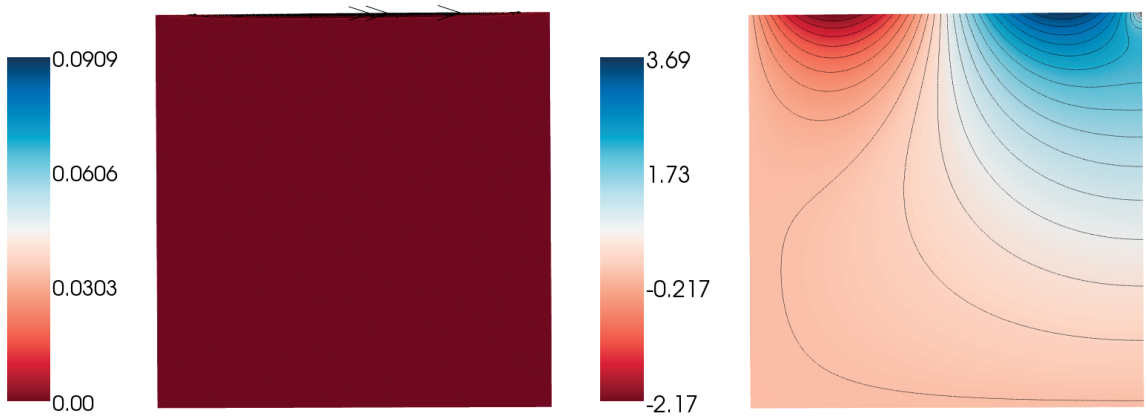
where H_K is the size of tetrahedron $K \in \mathcal{T}_h$. As derived in Section 3.1.5, we set $\underline{\alpha} = 165$ and we study behaviour of the PSPG schemes for spatial stabilization parameter with $\bar{\alpha} = 0.01,\ 0.001$ and transient stabilization parameter with $\bar{\alpha} = 0.1,\ 1$.

We define the discrete norms

$$\sqrt{\Delta t \sum_{n=1}^{N} \left\| \boldsymbol{u}_h^n \right\|_{H^1(\Omega)}} \quad \text{and} \quad \sqrt{\Delta t \sum_{n=1}^{N} \left\| p_h^n \right\|_{L^2(\Omega)}} \tag{3.84}$$

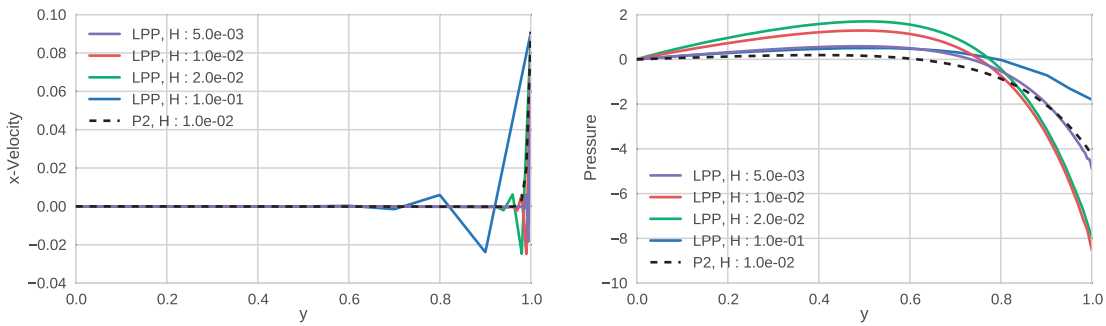
that we will simply refer to as the H^1 norm of the velocity and L^2 norm of the pressure respectively. For all the given stabilizations, stabilization parameters, mesh sizes and timestep sizes, we compute the H^1 norm of the velocity, the L^2 norm of the pressure and the condition number of the matrix. Although for $3D$ simulations we would generally use iterative solvers, we choose to use a direct LU solver in this test case in order to isolate solver issues and analyze those separately at a further point. All $2D$ computations in this work are done using the free and open source FEniCS package [107] and PETSc solver [108].

Remark 9. *Due to the lack of access to the elemental level of matrix assembly in the FEniCS*



(a) Velocity magnitude (b) Pressure

Figure 3.2: $\mathbb{P}_2 - \mathbb{P}_1$ solution for the 2D time-dependent lid-driven cavity test case at $T = 0.01$ with $\Delta t = 1e-4$ and $H = 0.01$



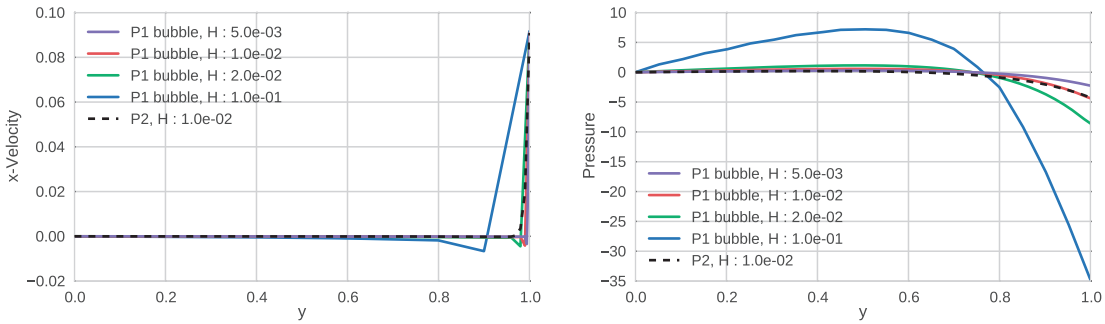
(a) Horizontal velocity profile along $x = 0.5$ (b) Pressure profile along $x = 0.25$

Figure 3.3: Local pressure projection solution profiles for the 2D time-dependent lid-driven cavity test case at $T = 0.01$ with $\Delta t = 1e-4$

package, we omit here the analysis of the eliminated bubble and reconstructed bubble.

Remark 10. *For the same reason, the implementation of the local pressure stabilization is done by introducing piecewise constant unknowns and test functions and coupling the Stokes system with an elementwise L^2 projection. As a consequence, the system is larger than it would be if we used for instance local Gauss integrations [109].*

The $\mathbb{P}_2 - \mathbb{P}_1$ solution is shown in Figure 3.2. The velocity is uniformly zero in the cavity except for a thin boundary layer close to Γ_t. For visualization purposes, we will compare horizontal velocities given by the different stabilization schemes along the axis $x = 0.5$. For the pressure, comparison will be done along the $x = 0.25$ axis to compare how well the crease is captured for y close to one.



(a) Horizontal velocity profile along $x = 0.5$

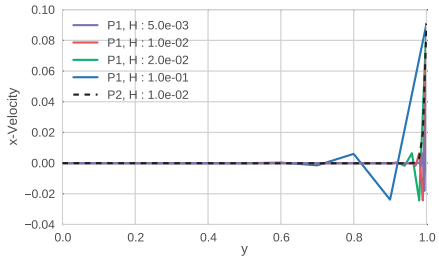
(b) Pressure profile along $x = 0.25$

Figure 3.4: Bubble enrichment solution profiles for the 2D time-dependent lid-driven cavity test case at $T = 0.01$ with $\Delta t = 1e-4$

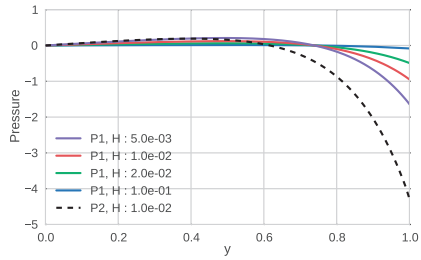
Velocity and pressure profile analysis Figures 3.3 and 3.4 illustrate the convergence of the horizontal velocity and pressure for the pressure projection scheme and the bubble enrichment scheme respectively. Both schemes show similar convergence for the horizontal velocity, although for the local pressure projection scheme there is a slightly larger undershoot around the boundary layer. Pressure convergences are also comparable although for the coarsest mesh, the bubble enrichment scheme shows a higher error than the local pressure projection scheme. Note that for the local pressure projection, the pressure for the coarsest mesh matches quite closely the $\mathbb{P}_2 - \mathbb{P}_1$ curve but it could be pure chance, since the curve deviates away and converges back when taking smaller mesh sizes. Note that the plots for $\Delta t = 1e-3$ are identical and therefore are not displayed.

Figure 3.5 shows convergence plots for the PSPG scheme with $\beta = 0$ and $\gamma = 0$ and spatial stabilization parameter with $\bar{\alpha} = 1e-3,\ 1e-2$. For both parameters, the velocity convergence plots are almost identical to the local pressure projection scheme. The pressure seems to converge very well for $\bar{\alpha} = 1e-3$ and slightly more slowly for $\bar{\alpha} = 1e-2$. Note that again the plots for $\Delta t = 1e-3$ are identical and therefore are not displayed. Figure 3.6 illustrates convergence for the same PSPG scheme but with a transient stabilization parameter and $\bar{\alpha} = 0.1,\ 1$. Velocities are similar but show slight oscillations. The pressures plots show oscillations and the pressure for the coarsest mesh breaks down. In this case, for $H = 0.1$ and $\nu = 10^{-6}$, computing the spatial stabilization parameters that have been used yields respectively $\alpha_K = 10^1,\ 10^2$ and the transient parameters are $\alpha_K = 5.10^{-4},\ 5.10^{-3}$. For $H = 0.01$, the spatial stabilization parameters are respectively $\alpha_K = 10^{-3},\ 10^{-2}$ and the transient parameters are $\alpha_K = 10^{-5},\ 10^{-4}$. Due to the small value of ν, the differences between spatial and transient parameters are then very large especially for small H. In Section 3.5, we compare the stabilizations with a higher value of $\nu = 1$.

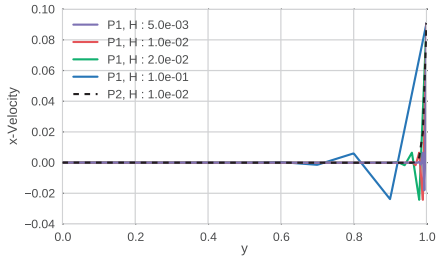
Figure 3.7 shows convergence plots for the PSPG scheme with $\beta = 1$ and $\gamma = 0$ and spatial stabilization parameter with $\bar{\alpha} = 1e-3,\ 1e-2$. Horizontal velocity approximations seem to be poorer approximations of the exact solution than for the $\beta = 0$ PSPG scheme. Indeed, they



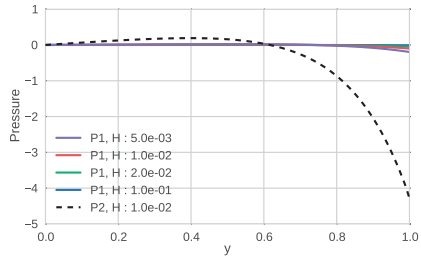
(a) Horizontal velocity profile along $x = 0.5$ with $\bar{\alpha} = 1e-3$



(b) Pressure profile along $x = 0.25$ with $\bar{\alpha} = 1e-3$

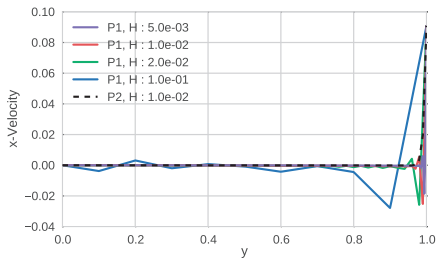


(c) Horizontal velocity profile along $x = 0.5$ with $\bar{\alpha} = 1e-2$

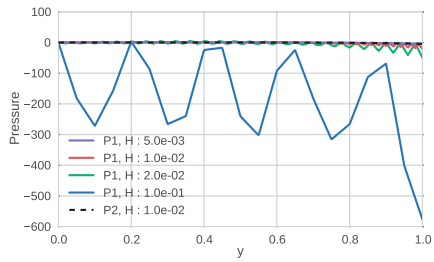


(d) Pressure profile along $x = 0.25$ with $\bar{\alpha} = 1e-2$

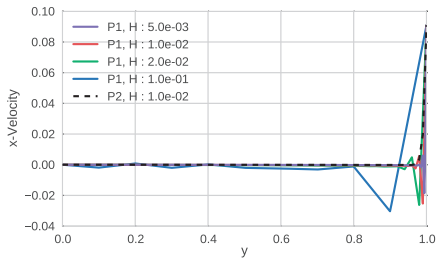
Figure 3.5: PSPG with $\beta = 0$ and $\gamma = 0$ solution profiles for the 2D time-dependent lid-driven cavity test case at $T = 0.01$ with $\Delta t = 1e-4$ and spatial stabilization parameter



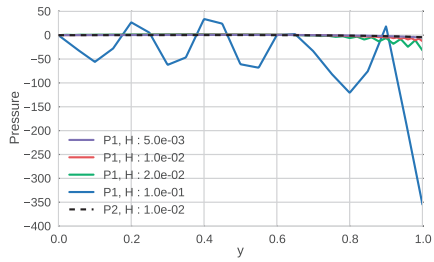
(a) Horizontal velocity profile along $x = 0.5$ with $\bar{\alpha} = 0.1$



(b) Pressure profile along $x = 0.25$ with $\bar{\alpha} = 0.1$

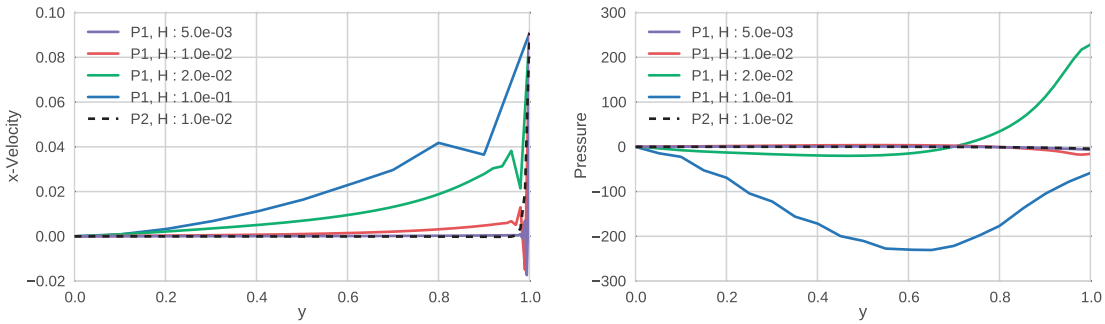


(c) Horizontal velocity profile along $x = 0.5$ with $\bar{\alpha} = 1$



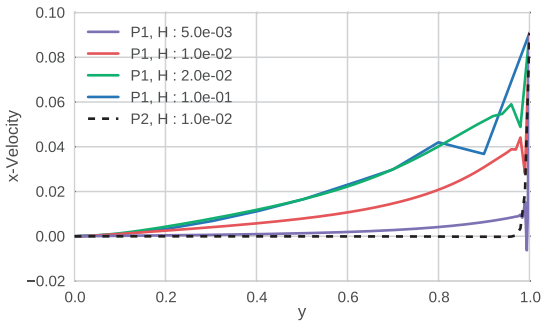
(d) Pressure profile along $x = 0.25$ with $\bar{\alpha} = 1$

Figure 3.6: PSPG with $\beta = 0$ and $\gamma = 0$ solution profiles for the 2D time-dependent lid-driven cavity test case at $T = 0.01$ with $\Delta t = 1e-4$ and transient stabilization parameter

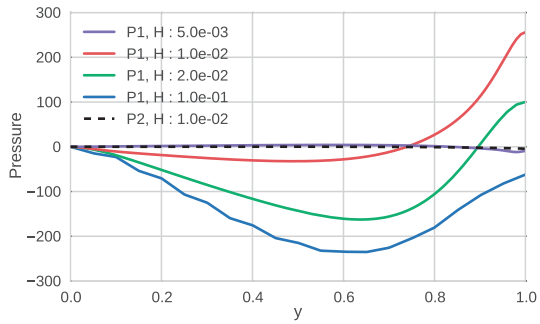


(a) Horizontal velocity profile along $x = 0.5$ with $\bar{\alpha} = 1e-3$

(b) Pressure profile along $x = 0.25$ with $\bar{\alpha} = 1e-3$



(c) Horizontal velocity profile along $x = 0.5$ with $\bar{\alpha} = 1e-2$



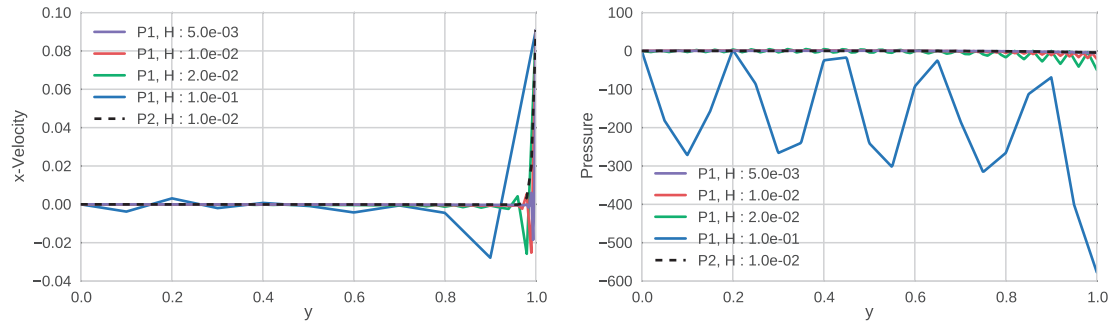
(d) Pressure profile along $x = 0.25$ with $\bar{\alpha} = 1e-2$

Figure 3.7: PSPG with $\beta = 1$ and $\gamma = 0$ solution profiles for the 2D time-dependent lid-driven cavity test case at $T = 0.01$ with $\Delta t = 1e-4$ and spatial stabilization parameter

fail to capture the uniformity of the horizontal velocity outside the boundary layer except for the finest mesh. The pressure also fails to be captured for the three coarsest meshes. Again, $\bar{\alpha} = 1e-3$ shows to be a better choice than $\bar{\alpha} = 1e-2$. Note that once again the plots for $\Delta t = 1e-3$ are identical and therefore are not displayed. Figure 3.8 illustrates convergence for the same PSPG scheme but with a transient stabilization parameter and $\bar{\alpha} = 0.1,\ 1$. Horizontal velocity approximations seem to capture the behaviour of the exact velocity fairly well although some oscillations are present and in fact, the curves show the same solutions as for $\beta = 0$ in Figure 3.6. The pressure curves are the same as for $\beta = 0$ as well.

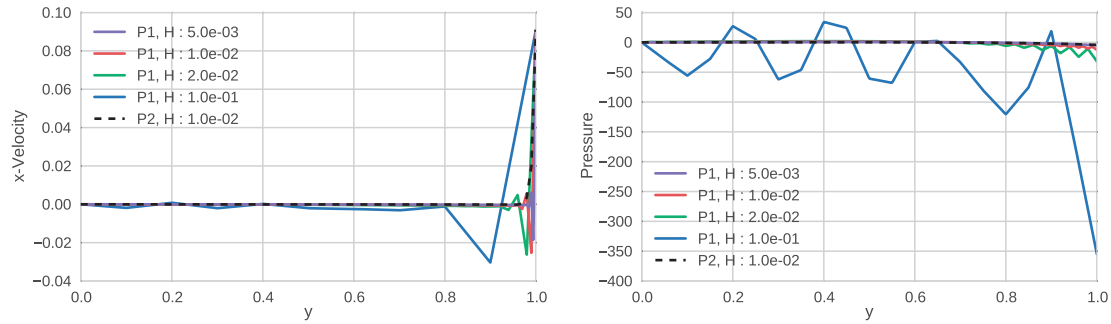
Figure 3.9 shows convergence plots for the PSPG scheme with $\beta = 1$ and $\gamma = \frac{-1}{\Delta t}, \frac{1}{\Delta t}$ and spatial stabilization parameter with $\bar{\alpha} = 1e-3$. The solutions are identical to solutions given by the PSPG scheme $\beta = 1$ and $\gamma = 0$ and in fact we have never observed a significant difference in the solution by taking either $\gamma = 0$ or $\gamma = \frac{-1}{\Delta t}, \frac{1}{\Delta t}$. However, algebraic properties of the matrix from the linear system produced by those different stabilizations slightly as seen in the further paragraph where we investigate condition numbers of the matrix for the different stabilizations.

Figures 3.10 and 3.11 show plots of the L^2 pressure norm over time for PSPG stabilizations with a spatial and transient stabilization parameter respectively. Figure 3.12 displays the same for the bubble enrichment scheme and local pressure projection. With spatial stabilization



(a) Horizontal velocity profile along $x = 0.5$ with $\tilde{\alpha} = 0.1$

(b) Pressure profile along $x = 0.25$ with $\tilde{\alpha} = 0.1$



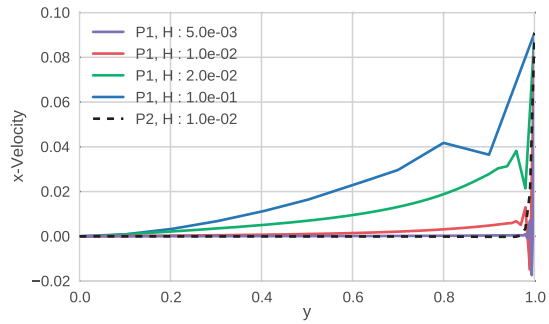
(c) Horizontal velocity profile along $x = 0.5$ with $\tilde{\alpha} = 1$

(d) Pressure profile along $x = 0.25$ with $\tilde{\alpha} = 1$

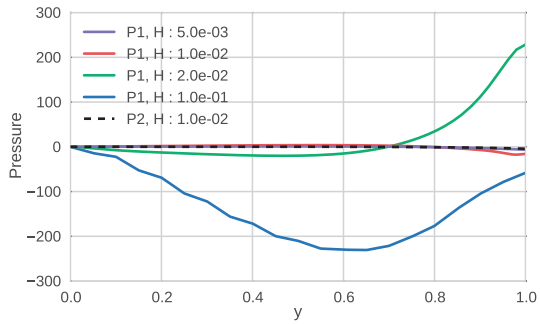
Figure 3.8: PSPG with $\beta = 1$ and $\gamma = 0$ solution profiles for the 2D time-dependent lid-driven cavity test case at $T = 0.01$ with $\Delta t = 1e-4$ and transient stabilization parameter

parameters, the PSPG scheme with $\beta = 1$ displays extremely high pressure norms at the beginning of the simulation. The pressure norm for the PSPG scheme $\beta = 0$ is increasing from zero to values progressively approaching the exact final pressure norm, somewhat similarly to the local pressure projection scheme. From 3.3, we do indeed expect it to converge. In Section 3.3, one of the conditions for stability and convergence for $\beta = 1$ was that the the stabilization parameter should be smaller than Δt, which is the case for the transient stabilization parameter. Choosing the transient stabilization parameter, the plots show that we do indeed get convergence towards the right pressure norm profile and furthermore, the curves we get are very similar to the bubble enrichment scheme. Spatial oscillations are however still present when choosing the transient stabilization parameter. This leads us to believe that from a time evolution point of view, the transient parameter is a correct one in a sense, but choosing it makes the scheme spatially unstable and therefore unsuitable for computations.

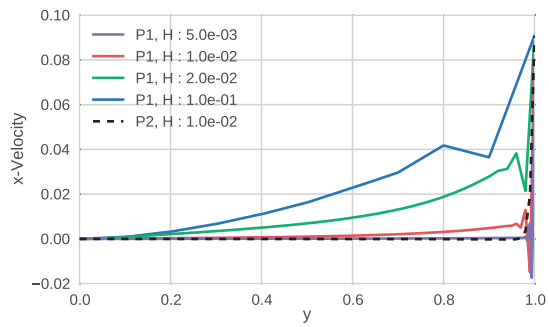
In [58, p. 1028], it is argued that small timestep instabilities similar to those we observe for the $\beta = 1$ stabilization with spatial stabilization parameter are caused by the choice of the projection of the initial velocity. Instead of using an L^2 or Lagrange projection, a Ritz projection is suggested in [58, p. 1019] to determine a discrete initial velocity compatible with the initial pressure of the problem. However, in the Ritz projection the velocity time derivative $\frac{\partial \boldsymbol{u}}{\partial t}$ is involved and since it is unknown in general, it is replaced by the limit of the



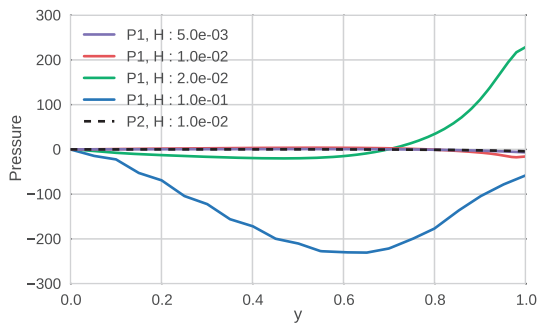
(a) Horizontal velocity profile along $x = 0.5$ with $\gamma = \frac{1}{\Delta t}$



(b) Pressure profile along $x = 0.25$ with $\gamma = \frac{1}{\Delta t}$

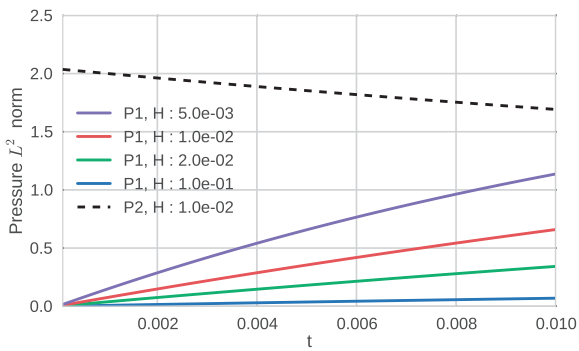


(c) Horizontal velocity profile along $x = 0.5$ with $\gamma = \frac{-1}{\Delta t}$

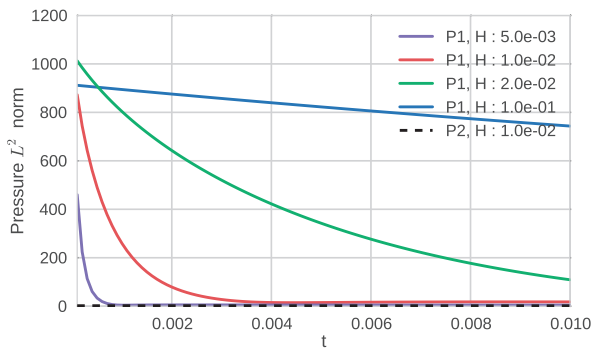


(d) Pressure profile along $x = 0.25$ with $\gamma = \frac{-1}{\Delta t}$

Figure 3.9: PSPG $\beta = 1$ and $\gamma = \frac{-1}{\Delta t}, \frac{1}{\Delta t}$ profiles for the 2D time-dependent lid-driven cavity test case at $T = 0.01$ with $\Delta t = 1e-4$ and spatial stabilization parameter with $\bar{\alpha} = 1e-3$

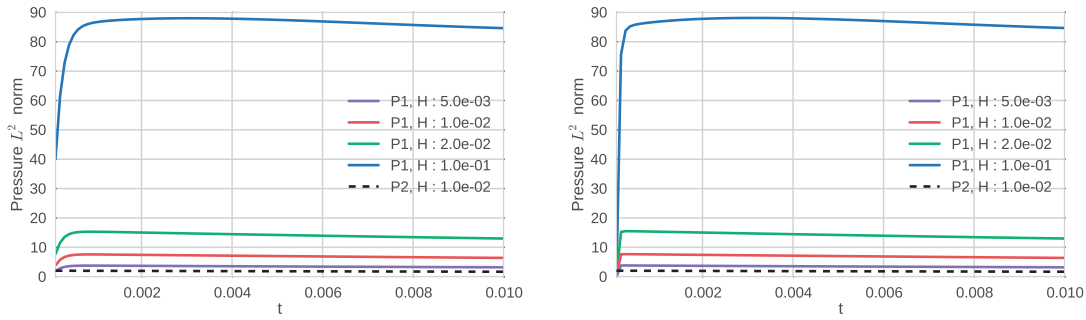


(a) $\beta = 0$



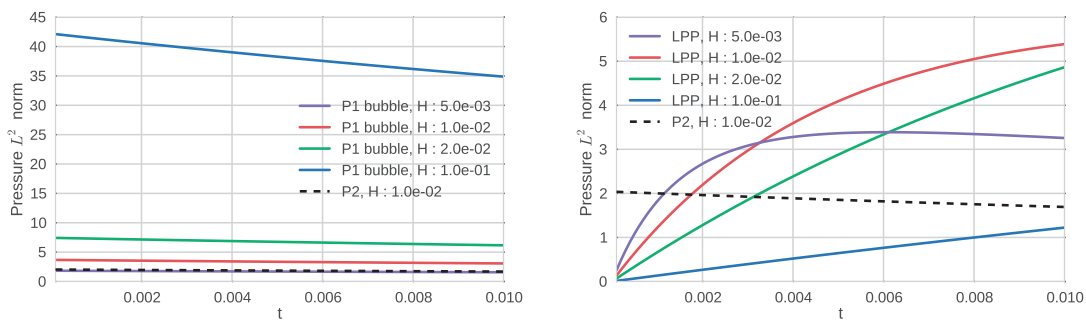
(b) $\beta = 1$

Figure 3.10: Pressure L^2 norm plots w.r.t. time of PSPG stabilizations with $\gamma = 0$ for the 2D time-dependent lid-driven cavity test case with $\Delta t = 1e-4$ and spatial stabilization parameter $\bar{\alpha} = 1e-3$



(a) $\beta = 0$ (b) $\beta = 1$

Figure 3.11: Pressure L^2 norm plots w.r.t. time of PSPG stabilizations with $\gamma = 0$ for the 2D time-dependent lid-driven cavity test case with $\Delta t = 1e-4$ and transient stabilization parameter $\bar{\alpha} = 1$



(a) Bubble enrichment (b) Local pressure projection

Figure 3.12: Pressure L^2 norm plots w.r.t. time of bubble enrichment scheme and local pressure projection method for the 2D time-dependent lid-driven cavity test case with $\Delta t = 1e-4$

momentum balance equation for $t \to 0$. Solving a Poisson problem for the pressure then yields the approximation of the velocity time derivative which in turn yields an initial velocity compatible with the computed pressure. In our case however, solving the Poisson problem for the pressure gives a constant pressure which yields zero as an approximation of the velocity time derivative. This is wrong, since the velocity boundary conditions we have set have a non-zero time-derivative. We therefore cannot easily compute the right Ritz projection which shows a limitation of their estimation method for the initial velocity. These results highlight the difficulties encountered with PSPG type methods in the small timestep limit.

In conclusion to this part, it seems that for low viscosities, small timesteps and non-stationary phenomena, the PSPG stabilization $\beta = 1$ is best avoided especially when pressure is of interest. Transient stabilization parameters seem to provide a natural temporal scaling but do not guarantee spatial stability of the pressure. A PSPG stabilization $\beta = 0$ with spatial stabilization parameter $\tilde{\alpha} = 1e-3$ seems to give correct results. In the end, the local pressure projection scheme and bubble enrichment scheme seem to give the best results.

Condition number analysis We now compare condition numbers of the full matrices in the Stokes' problem linear systems associated with the different stabilizations considered. We compare them for values $H = 0.04,\ 0.2$ and $\Delta t = 1e-6,\ 1e-3$.

Figures 3.13 and 3.14 show the condition numbers of the left-hand side matrix in the linear system for the coupled Stokes problem corresponding to the considered PSPG schemes, local pressure projection scheme and bubble enrichment scheme.

As expected from the previous paragraphs, PSPG schemes with $\beta = 1$ and spatial stabilization parameter as well as all PSPG schemes with transient stabilization parameter exhibit particularly high condition numbers for small timesteps compared to the stable PSPG stabilization $\beta = 0, \gamma = 0$. The condition number for the stable PSPG scheme $\beta = 0, \gamma = 0$ seems to be inversely proportional to the stabilization parameter for a spatial stabilization parameter. The condition numbers for the local pressure projection scheme seem to be somewhat higher than for the stable PSPG scheme. Although direct comparison of the condition number for

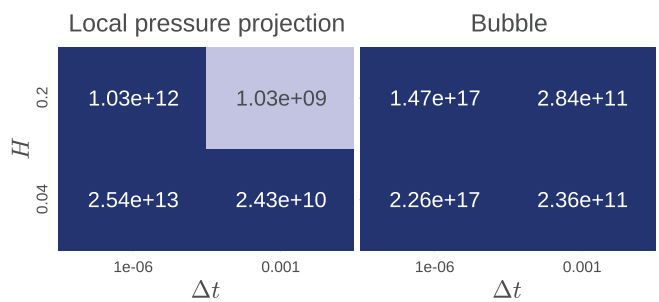
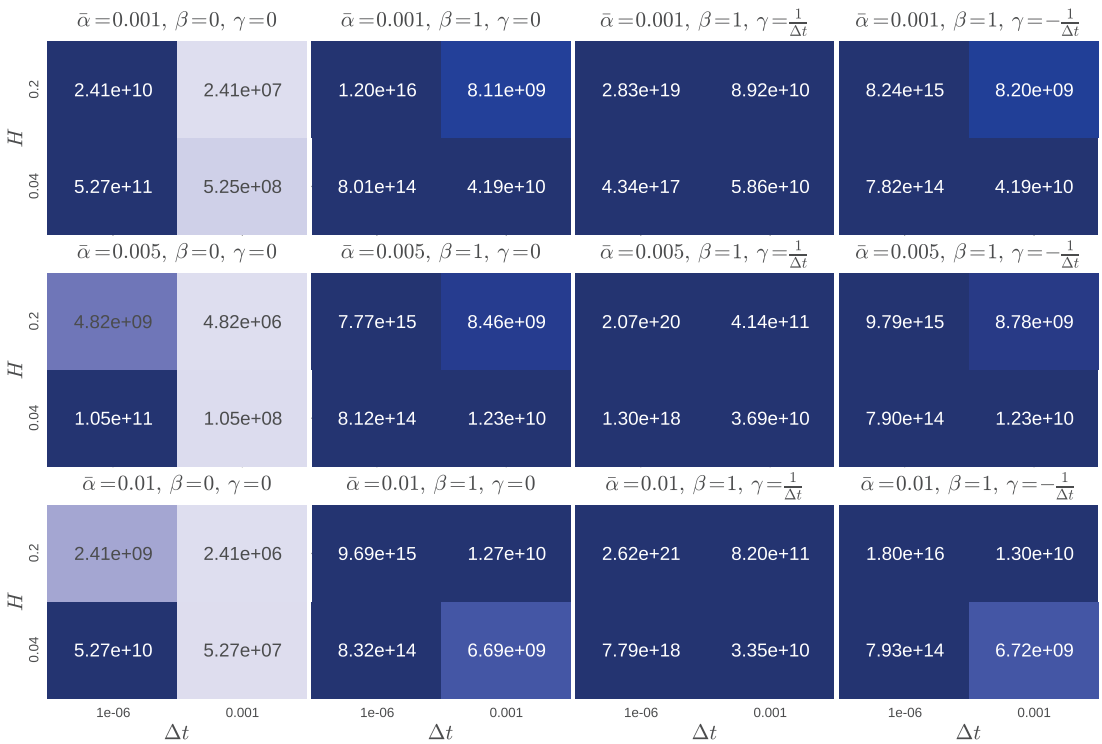
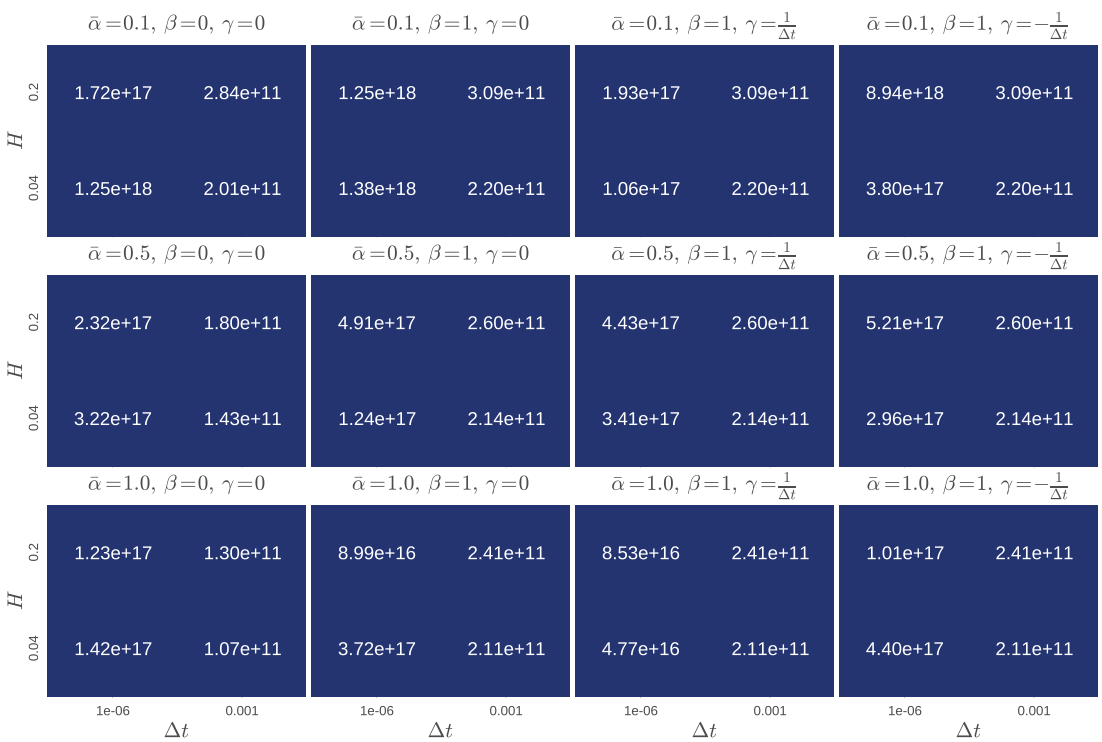


Figure 3.13: Condition number of the linear system for the 2D time-dependent lid-driven cavity test case at $T = 0.01$



(a) PSPG with spatial stabilization parameter



(b) PSPG with transient stabilization parameter

Figure 3.14: Condition number of the linear system for the 2D time-dependent lid-driven cavity test case at $T = 0.01$

the bubble-enriched scheme with the other schemes might be unfair since the linear system is larger for the bubble-enriched scheme, we notice that the condition number behaves roughly in $\frac{1}{\Delta t^2}$ for the bubble-enriched scheme with respect to timestep. For both the local pressure projection and stable PSPG scheme, the condition number seems to behave in $\frac{1}{\Delta t}$ with respect to timestep. We therefore expect the stable PSPG scheme and local pressure projection schemes to be better suited to Stokes-type problems where small timesteps are required. Due to their poor conditioning and appearance of oscillations as noted previously, transient stabilization parameter-based PSPG schemes will be dropped from the analysis from now on.

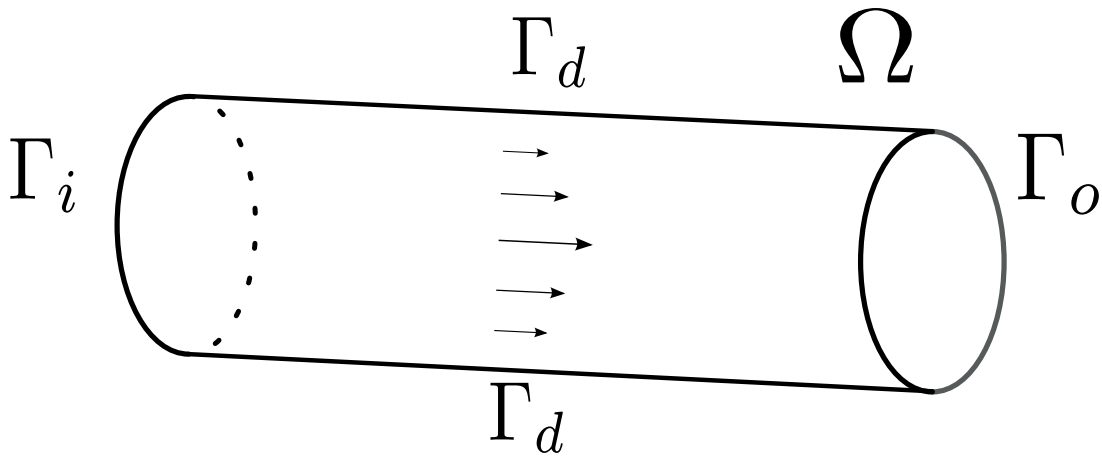


Figure 3.15: A sketch of the 3D Poiseuille cavity

3.5 Analysis of stabilization schemes on 3D Poiseuille with time-varying inflow

A difficulty in evaluating accuracy of numerical schemes for the time-dependent Stokes equations is to design a test case where the solution is non-trivial but known. We want our solution to be non-stationary and ideally be able to control the time derivative of the velocity to be able to assess differences between consistent and non-consistent PSPG schemes. Such a test case can be created given that we know a stationary solution to the Stokes equations for a Poiseuille flow given an inflow velocity. We can set a time-dependent inflow and choose the force term on the right-hand side of the time-dependent momentum equation to be the time derivative of the velocity such that at each time-step, the time derivative of the velocity cancels with the right-hand side. We can thus choose an oscillating inflow in time and by choosing the frequency of the oscillations we can also choose the time derivative of the velocity.

With this in mind, we will analyze different stabilization schemes and evaluate their performance in terms of velocity error, pressure error and running time on a 3D Poiseuille test case with time-varying inflow.

A sketch of the cylindrical domain is given in Figure 3.15 where Ω is the computational domain, Γ_i is the inlet boundary where inflow velocity is imposed, Γ_o is the free outflow boundary and Γ_d are the upper and lower boundaries where a zero Dirichlet boundary condition will be imposed. Let the computational domain be a cylinder of length 12, radius 1 such that the origin corresponds to the middle of the inflow disk and the point $(0,0,12)$ corresponds to the middle of the outflow disk and $T > 0$ the final simulation time. For this test case, we choose $\nu = 1$.

Let us define $\boldsymbol{u}_{ex} : \Omega \times [0,T] \to \mathbb{R}^3$ and $p_{ex} : \Omega \times [0,T] \to \mathbb{R}$ as follows

$$\boldsymbol{u}_{ex}(x,y,z,t) = \begin{pmatrix} 0 \\ 0 \\ \sin(2\pi\omega t)(1-x^2-y^2) \end{pmatrix} \quad \forall (x,y,z) \in \Omega,\ \forall t \in [0,T] \tag{3.85}$$

$$p_{ex}(x,y,z,t) = 4(12-z)\sin(2\pi\omega t) \quad \forall (x,y,z) \in \Omega,\ \forall t \in [0,T] \tag{3.86}$$

where $\omega \in \mathbb{R}$ a parameter controlling frequency of the inflow velocity oscillations.

The time-dependent Stokes problem that we want to solve is then given by the following equations

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} - \nu\Delta\boldsymbol{u} + \nabla p = \dfrac{\partial \boldsymbol{u}_{ex}}{\partial t} & \text{in } \Omega \times [0,T] \\ \nabla \cdot \boldsymbol{u} = 0 & \text{in } \Omega \times [0,T] \end{cases} \tag{3.87}$$

and the boundary conditions and initial condition are given as follows

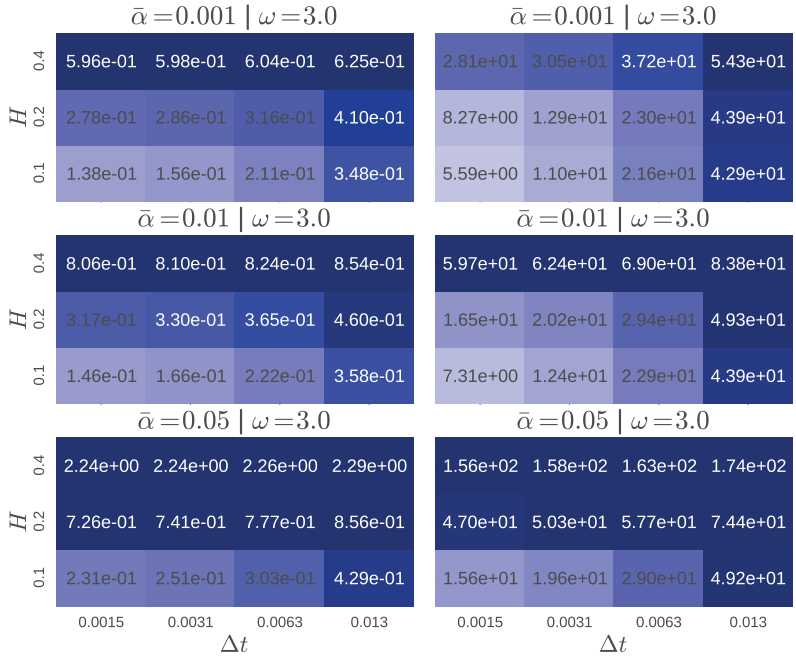
$$\begin{cases} \boldsymbol{u} = 0 & \text{on } \Gamma_d \times [0,T] \\ \boldsymbol{u} = \boldsymbol{u}_{ex} & \text{on } \Gamma_i \times [0,T] \\ \dfrac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} - p\boldsymbol{n} = 0 & \text{on } \Gamma_o \times [0,T] \\ \boldsymbol{u}(0,\cdot) = \boldsymbol{u}_{ex}(0,\cdot) & \text{in } \Omega. \end{cases} \tag{3.88}$$

It can easily be verified that $(\boldsymbol{u}_{ex}, p_{ex})$ is a solution to the problem (3.87)-(3.88).

We have set $T = 1$ and chosen to analyze the cases $\omega = 1,3$. We only show results for $\omega = 3$. The whole simulation then consists of three full periods of sinusoidal inflow. To highlight the differences between the large timestep case and the small timestep case, we will further analyze the case $\omega = 1000$ and $T = 3/1000$ such that again, the simulation consists of three periods of sinusoidal inflow but the timesteps are much shorter.

Three different mesh sizes $H = 0.1, 0.2, 0.4$ and four different timesteps $\Delta t = \frac{1}{80}, \frac{1}{160}, \frac{1}{320}, \frac{1}{640}$ were used. The problem was numerically solved with the *cfsFlow* software, see for instance [36], coupled with the *PETSc* package for solving the linear system, using a GMRES solver and ILU preconditioner.

We compare the H^1 norm of the velocity error and L^2 norm of the pressure error as defined in (3.84), number of iterations of the GMRES solver and CPU time for the considered stabilization schemes. The considered stabilization schemes are PSPG schemes with $\gamma = 0$, $\beta = 0,1$ and spatial stabilization parameters with $\bar{\alpha} = 0.001, 0.01, 0.05$, the local pressure projection scheme with $\bar{\alpha} = 1$ and the full bubble-enriched scheme, the bubble reconstruction method and the bubble elimination method. For a description of the stabilization methods, see Section 3.1.5.



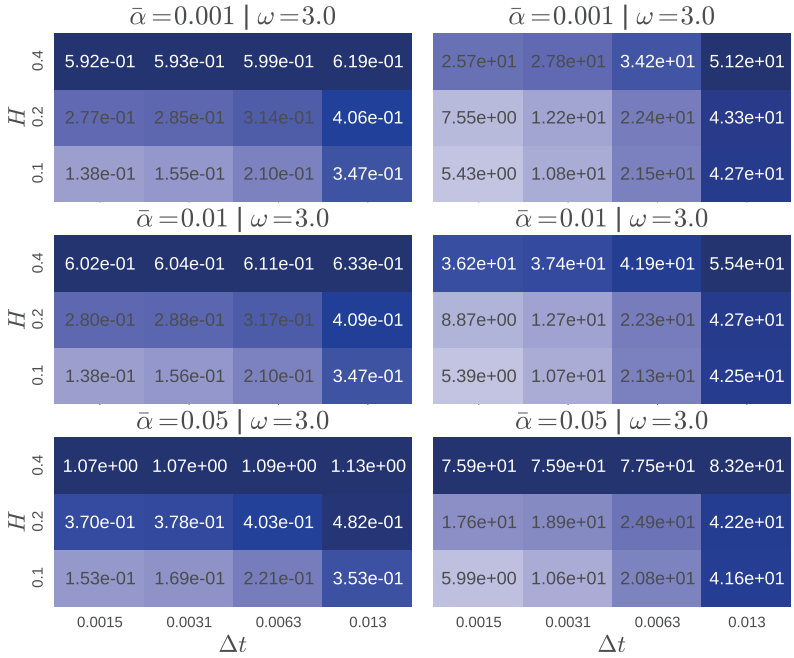
(a) Velocity H^1 error (b) Pressure L^2 error

Figure 3.16: Errors for PSPG stabilization $\beta = 0, \gamma = 0$ and spatial stabilization parameter in 3D Poiseuille test case

Velocity and pressure error analysis Figures 3.16 and 3.17 show H^1 norm velocity and L^2 norm pressure errors for the PSPG schemes $\gamma = 0$ with spatial stabilization parameter and $\beta = 0, 1$ respectively. For both stabilizations, the stabilization parameter $\bar{\alpha} = 0.05$ seems to be too large and introduces additional error to both velocity and pressure compared to the two other stabilization parameters $\bar{\alpha} = 0.001, 0.01$.

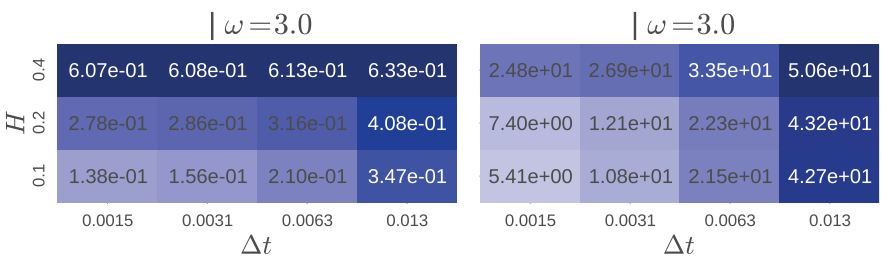
Looking at velocity errors in Figures 3.16 and 3.17, it seems that the error due to space discretization is dominant for small timesteps and in that case the error behaves as $O(H)$ as predicted by the convergence result from Theorem 2. Although slightly higher errors in velocity are observed for the $\beta = 0$ stabilization, the behaviour of both schemes is similar. For larger timesteps, the time discretization error becomes dominant but both schemes fare similarly. Indeed, Theorem 2 shows that the two PSPG schemes contain differences in the part of the error due to space discretization, but have the same bound for the part of the error due to time discretization. The scheme $\beta = 1$ seems to be not only slightly more accurate but also somewhat less sensitive to the choice of the stabilization parameter.

For the pressure errors, it seems that again the PSPG scheme $\beta = 1$ shows slightly lower errors in all cases compared to $\beta = 0$ but both schemes give approximately equivalent results. For smaller H, the time discretization error starts dominating and as predicted, we observe at least a $O(\Delta t)$ convergence.



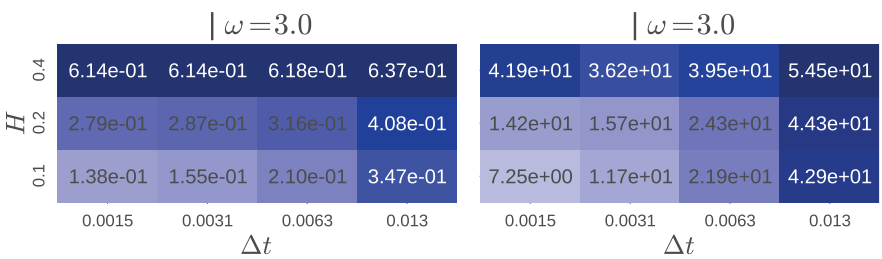
(a) Velocity H^1 error (b) Pressure L^2 error

Figure 3.17: Errors for PSPG stabilization $\beta = 1, \gamma = 0$ and spatial stabilization parameter in 3D Poiseuille test case



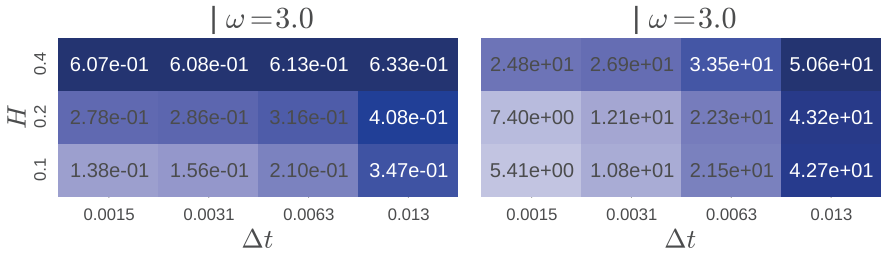
(a) Velocity H^1 error (b) Pressure L^2 error

Figure 3.18: Errors for full bubble enrichment in 3D Poiseuille test case



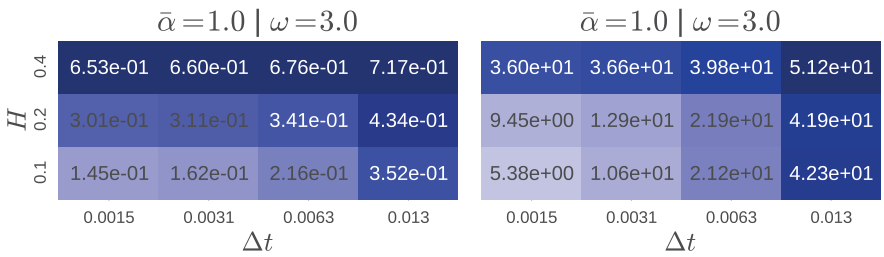
(a) Velocity H^1 error (b) Pressure L^2 error

Figure 3.19: Errors for bubble elimination method in 3D Poiseuille test case



(a) Velocity H^1 error (b) Pressure L^2 error

Figure 3.20: Errors for bubble reconstruction method in 3D Poiseuille test case



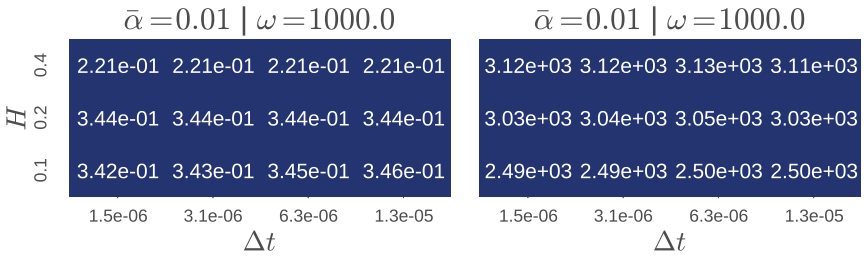
(a) Velocity H^1 error (b) Pressure L^2 error

Figure 3.21: Errors for local pressure projection method in 3D Poiseuille test case

Velocity and pressure error for the bubble enrichment scheme, bubble elimination and bubble reconstruction schemes are shown in Figures 3.18, 3.19 and 3.20. In fact, the bubble enrichment and bubble reconstruction errors are identical which is reassuring since the schemes should give the same solution, although some differences might occur due to the iterative solver. They present the expected $O(\Delta t)$ convergence for large Δt. In the case of the bubble elimination, the pressure error grows for small values of Δt. The error grows significantly more for even smaller values of Δt. This shows that the bubble terms that are ignored in the bubble elimination scheme are essential for stabilizing the pressure at small timesteps. A small perturbation of the velocity can also be seen for small timesteps, indicating a potential loss of accuracy for even smaller timesteps.

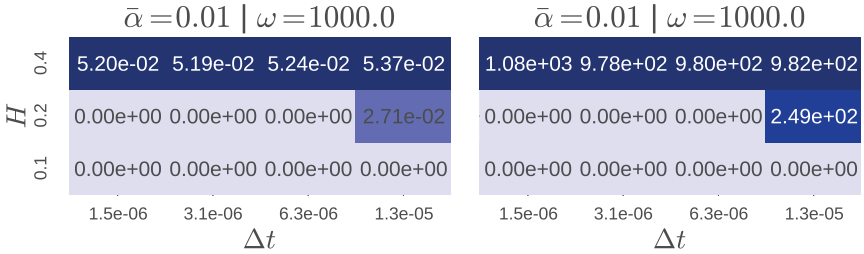
Figure 3.21 shows errors for the last scheme, the local pressure projection scheme. The velocity behaves like the bubble enrichment schemes, if slightly worse and pressure errors are similar to the pressure errors for the PSPG scheme $\beta = 1$.

It seems that for this test case, no scheme is particularly worse off than others in terms of velocity and pressure errors, but not having to choose a stabilization parameter is a considerable advantage. This however changes when considering smaller as in the next paragraph.



(a) Velocity H^1 error (b) Pressure L^2 error

Figure 3.22: Errors for PSPG stabilization $\beta = 0, \gamma = 0$ and spatial stabilization parameter $\bar{\alpha} = 0.01$ in 3D Poiseuille test case for small timesteps



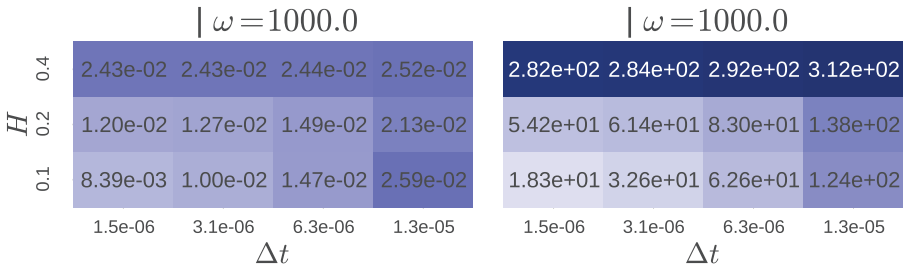
(a) Velocity H^1 error (b) Pressure L^2 error

Figure 3.23: Errors for PSPG stabilization $\beta = 1, \gamma = 0$ and spatial stabilization $\bar{\alpha} = 0.01$ parameter in 3D Poiseuille test case for small timesteps. Zero values are shown where the solver fails to converge.

Runtime and solver iteration analysis in the small timestep case We consider the case $\omega = 1000$ and $T = 3/1000$ where three periods of the sinusoidal inflow are again covered, but timesteps are much smaller. The timesteps are the same as earlier but smaller by a multiplicative factor of $1/1000$ such that the sinusoidal inflow is resolved with the same accuracy.

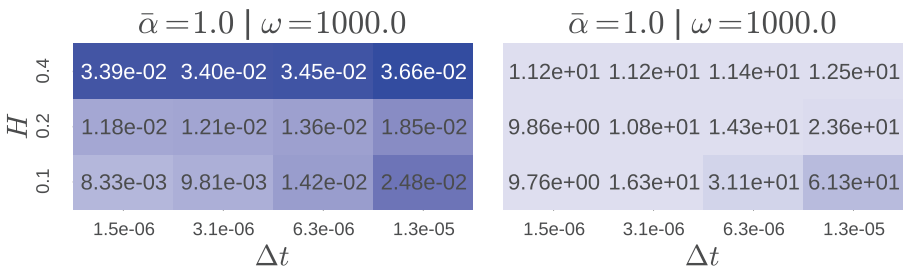
Figures 3.22 and 3.23 show H^1 norm velocity and L^2 norm pressure errors for the PSPG schemes $\gamma = 0$ with spatial stabilization parameter and $\beta = 0, 1$ respectively. For both stabilizations, the stabilization parameter $\bar{\alpha} = 0.01$, which worked well in the previous case, has been used. In Figure 3.23, zero values are displayed when the solver has failed to converge. We observe that for $\beta = 0$ the scheme converges to the wrong solution and for $\beta = 1$, in the small timestep limit, the matrix associated with the problem becomes singular.

Figures 3.24 and 3.25 show results for the full bubble enrichment scheme and the local pressure projection scheme. Velocity and pressure seem to converge for both schemes although for the local pressure projection scheme, the error on the pressure seems to grow for $\Delta t = 1.3e-5$ as H decreases. We believe that the scheme converges as it should but for large H, the errors are small by chance. Indeed, if the pressure is stabilized correctly, it will be linear along



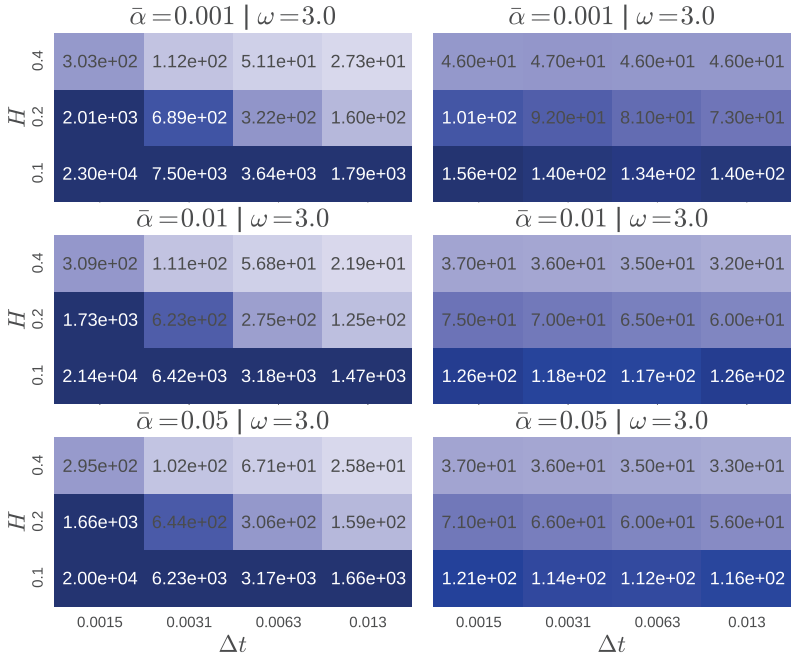
(a) Velocity H^1 error (b) Pressure L^2 error

Figure 3.24: Errors for full bubble enrichment in 3D Poiseuille test case for small timesteps



(a) Velocity H^1 error (b) Pressure L^2 error

Figure 3.25: Errors for local pressure projection method in 3D Poiseuille test case for small timesteps



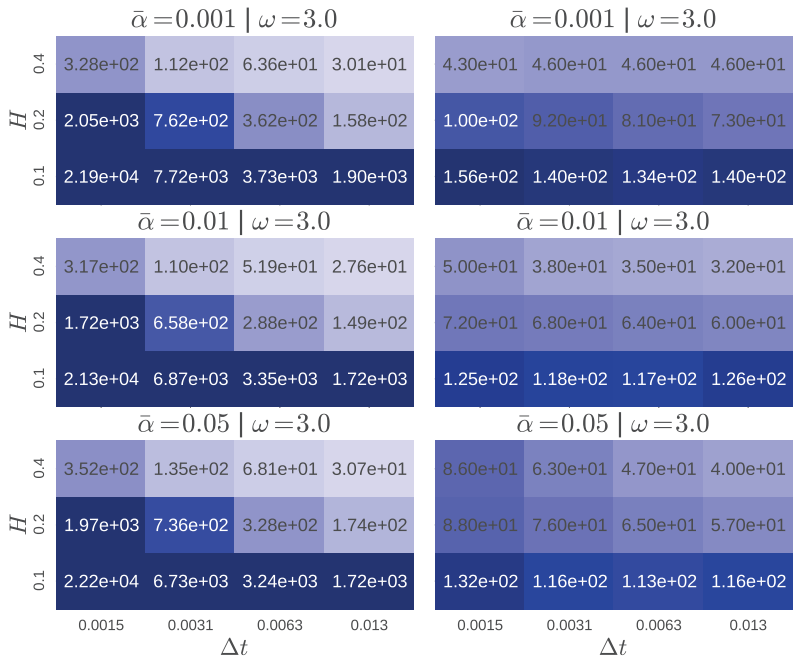
(a) CPU time (b) Median solver iterations

Figure 3.26: CPU time and median number of iterations for PSPG stabilization $\beta = 0, \gamma = 0$ and spatial stabilization parameter in 3D Poiseuille test case

the cylinder axis. Since pressure at the outflow boundary is zero, the problem reduces to a monodimensional one in terms of the pressure, where the unknown is for instance the value of the pressure at the inflow boundary. For this reason, it is not improbable to obtain a small pressure error by chance for $H = 0.4$. Note that for $H = 0.4$, the local pressure projection scheme shows a pressure error much smaller than the bubble enrichment scheme by a factor $1/20$, which corroborates our belief.

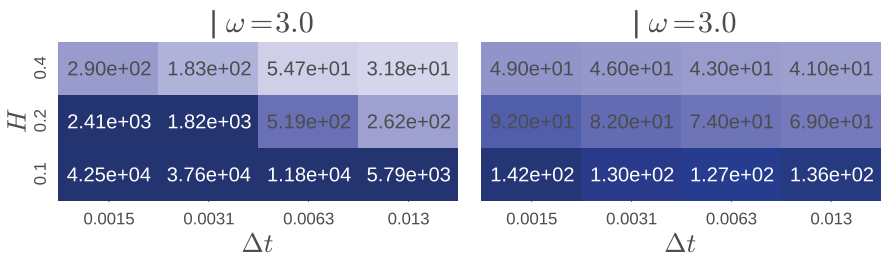
Runtime and solver iteration analysis We now analyze running times and number of iterations of the linear system solver for large timesteps once again ($\omega = 3$ and $T = 1$). Figures 3.26 and 3.27 show CPU time and median number of iterations done by the GMRES solver for the PSPG schemes $\beta = 0$ and $\beta = 1$. Running times are similar for both schemes and roughly of order $O(\frac{1}{H^3}\frac{1}{\Delta t})$ as expected with slightly faster computations for the scheme $\beta = 0$.

The scheme $\beta = 1$ performs a few more solver iterations in general which could also explain why it shows slightly lower errors than the scheme $\beta = 0$. We see the largest difference when H is large and Δt is small. In that case, we observe an increase in the number of iterations for the scheme $\beta = 1$ compared to the scheme $\beta = 0$. This is in line with the observations in Section 3.4 that condition number grows faster for scheme $\beta = 1$ when timestep decreases. $\bar{\alpha} = 0.01$ seems to be a better choice than $\bar{\alpha} = 0.001$ in terms of iterations performed by the solver.



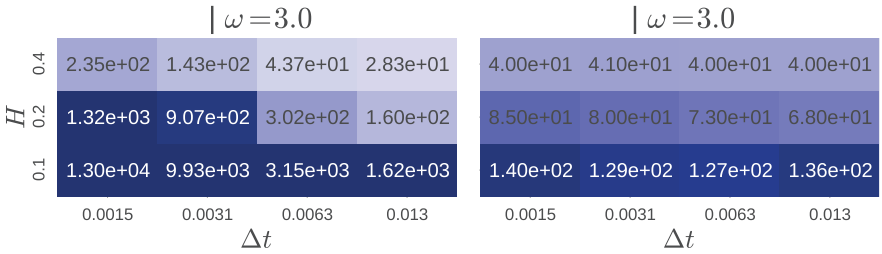
(a) CPU time (b) Median solver iterations

Figure 3.27: CPU time and median number of iterations for PSPG stabilization $\beta = 1, \gamma = 0$ and spatial stabilization parameter in 3D Poiseuille test case



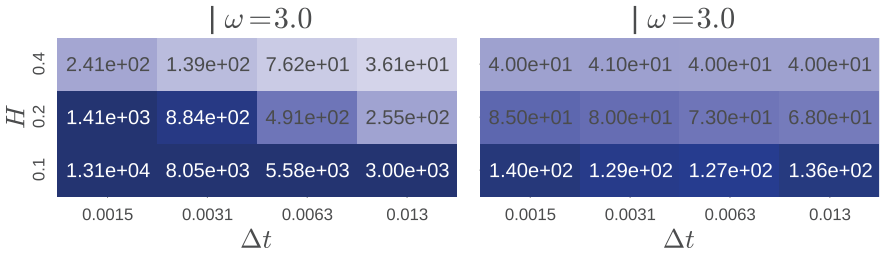
(a) CPU time (b) Median solver iterations

Figure 3.28: CPU time and median number of iterations for full bubble enrichment in 3D Poiseuille test case



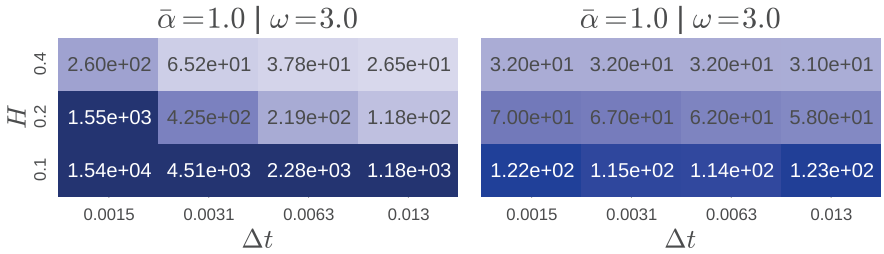
(a) CPU time (b) Median solver iterations

Figure 3.29: CPU time and median number of iterations for bubble elimination method in 3D Poiseuille test case



(a) CPU time (b) Median solver iterations

Figure 3.30: CPU time and median number of iterations for bubble reconstruction method in 3D Poiseuille test case



(a) CPU time (b) Median solver iterations

Figure 3.31: CPU time and median number of iterations for local pressure projection method in 3D Poiseuille test case

CPU time and median number of solver iterations for the bubble enrichment scheme, bubble elimination and bubble reconstruction schemes are shown in Figures 3.28, 3.29 and 3.30.

The number of iterations seems to be slightly higher for the full bubble enriched method compared to the other two methods which is due to the larger system to solve. CPU time is about twice as high for the full bubble for $H = 0.2$ and three times as high for $H = 0.1$ compared to the two other methods. The bubble reconstruction scheme is at most slightly slower than the bubble elimination scheme and at best faster, showing that the reconstruction is indeed a cheap operation. The number of iterations seems to be higher for bubble-based schemes than for both considered PSPG schemes and is in line with the observation in Section 3.4 that condition number for bubble-based schemes is higher than for PSPG schemes with adequately selected stabilization parameter. This is reflected in slightly higher running times for bubble elimination and reconstruction methods compared to PSPG schemes.

Figure 3.31 shows CPU time and median number of iterations for the local pressure projection scheme. Running time and iterations seem to be on par with the PSPG method $\beta = 0$ for $\bar{\alpha} = 0.01$ and therefore slightly better than the bubble-based schemes.

We have determined for this Stokes problem, all schemes give almost equivalently good results for large timesteps. For small timesteps, PSPG schemes fail. Indeed, the PSPG scheme with $\beta = 0$ seems to converge to the wrong solution and the PSPG scheme with $\beta = 1$ yields a singular matrix. The local pressure projection and bubble enrichment schemes give good results even in the small timestep case. The local pressure projection scheme and PSPG scheme $\beta = 0$, $\bar{\alpha} = 0.01$ seem to be among the least time-consuming schemes, although the differences in performance are relatively minor for all considered schemes, except the full bubble which requires a larger system to be solved. The bubble-based schemes and local pressure projection are attractive methods due to the absence of stabilization parameter and as a conclusion, this test case shows that the local pressure projection scheme and bubble reconstruction scheme are the methods which should be preferred. We will see in Section 3.6 that when the Stokes equations are coupled with a characteristics method for solving the full Navier-Stokes equations, the local pressure projection method also requires a stabilization

parameter to be set and the bubble enrichment will give good results.

3.6 3D comparison of wave profiles for different stabilization schemes on experimental VAW data

After having investigated the performance and accuracy of the different stabilization schemes on the time-dependent Stokes equations, we now compare their accuracies on the full Navier-Stokes equations. The setup is exactly the same as in Section 2.2.2 and we will compare wave profiles in a cavity of angle 1° and relative wave height $R_h = 0.5$ for the different stabilizations with parameters corresponding to the simulation with the mesh of medium fineness in Section 2.2.2.

Figure 3.32 shows wave profiles for the full bubble enriched scheme, bubble elimination and bubble reconstruction schemes and it seems like the profiles match perfectly. It seems like there is therefore no loss in eliminating the bubble completely and furthermore, this scheme contains no stabilization parameter to set. All curves present excellent agreement with the experimental data.

Figure 3.33 shows wave profiles for the local pressure projection with stabilization parameters $\bar{\alpha} = 0.01, 0.1, 1$. We can observe that for $\bar{\alpha} = 1$, the wave profile breaks down and the free surface starts to present spurious oscillations, which is why it was necessary to consider smaller stabilization parameters as well. $\bar{\alpha} = 0.01$ seems to yield best results; the same value has also been observed to give satisfactory results in [96] for high Reynolds numbers.

Figure 3.34 shows wave profiles for the PSPG scheme with $\beta = 1$ spatial stabilization parameter and $\bar{\alpha} = 5e-6, 1e-5, 5e-5$. Note that $\bar{\alpha}$ has been taken about 10^3 times smaller than was the case for a pure Stokes problem since taking the same values as before causes the solver to fail. This indicates that the stabilization parameter should possibly depend on the Reynolds number as had been observed in [110]. The wave profile for $\bar{\alpha} = 5e-5$ contains oscillations at the free surface but the two profiles corresponding to $\bar{\alpha} = 5e-6, 1e-5$ yield good matches with experimental results.

Figure 3.35 shows wave profiles for the PSPG scheme with $\beta = 0$ spatial stabilization parameter and $\bar{\alpha} = 5e-6, 1e-5, 5e-5$. This time, we see the wave profile completely breaking down for $\bar{\alpha} = 5e-5$ and losing some accuracy for $\bar{\alpha} = 1e-5$.

Figures 3.36 and 3.37 show wave profiles for the PSPG scheme with $\beta = 0, 1$ and spatial stabilization parameter divided by the local Reynolds number and $\bar{\alpha} = 1e-3, 5e-2, 1e-2$. Dividing by the local Reynolds number had been proposed by Franca-Hughes in [111]. Both PSPG schemes seems to yield excellent agreement for all proposed values of $\bar{\alpha}$ and results seem much more robust with respect to the choice of $\bar{\alpha}$ compared to the simple spatial stabilization parameter.

Dividing by the local Reynolds number can also be done for the local pressure projection scheme. Figure 3.38 shows the result for $\bar{\alpha} = 10, 50, 100$ which shows reasonable accuracy for $\bar{\alpha} = 10$. Taking $\bar{\alpha} = 1$ would cause the solver to fail.

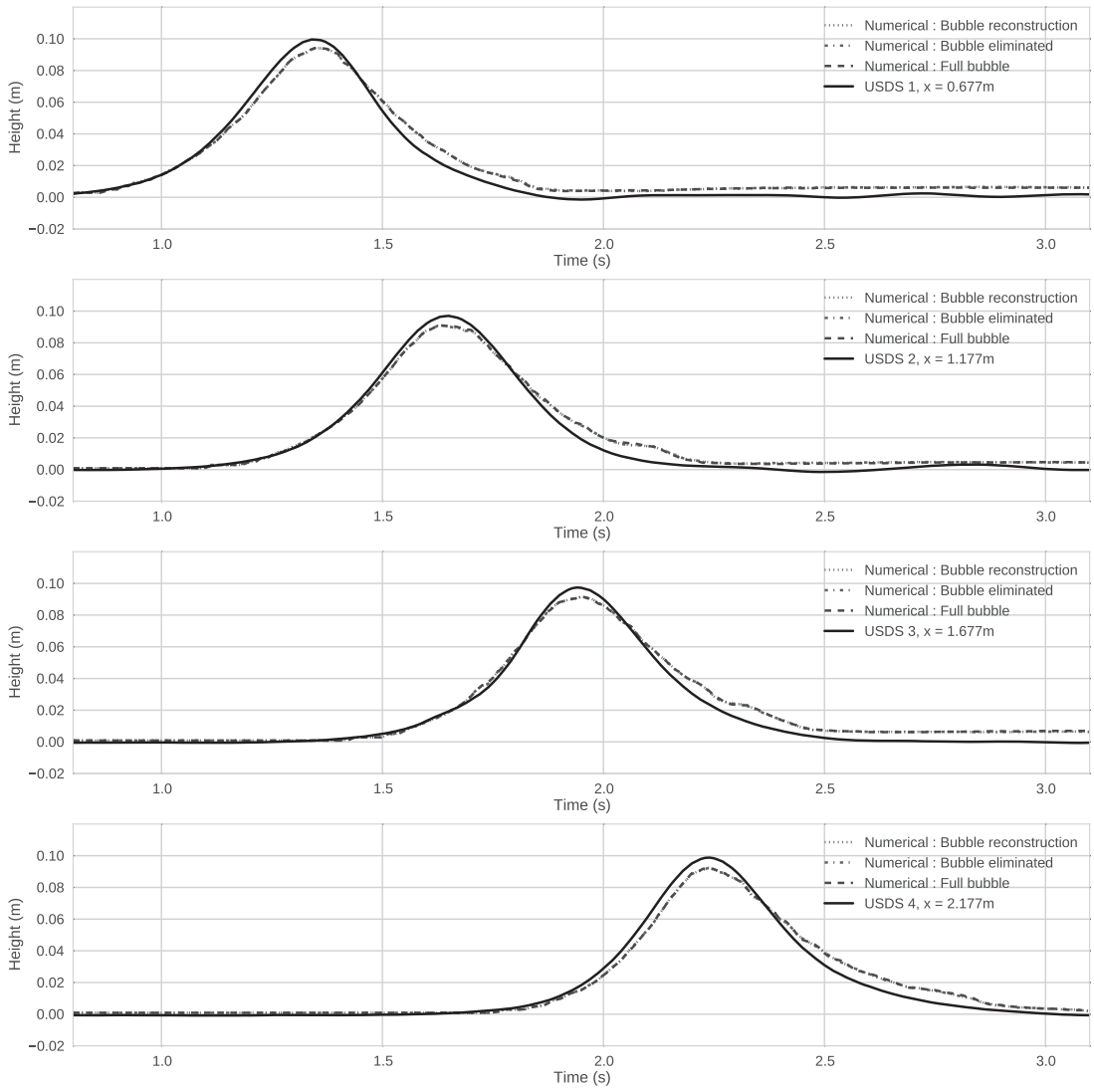


Figure 3.32: Wave profiles for bubble enrichment stabilizations.

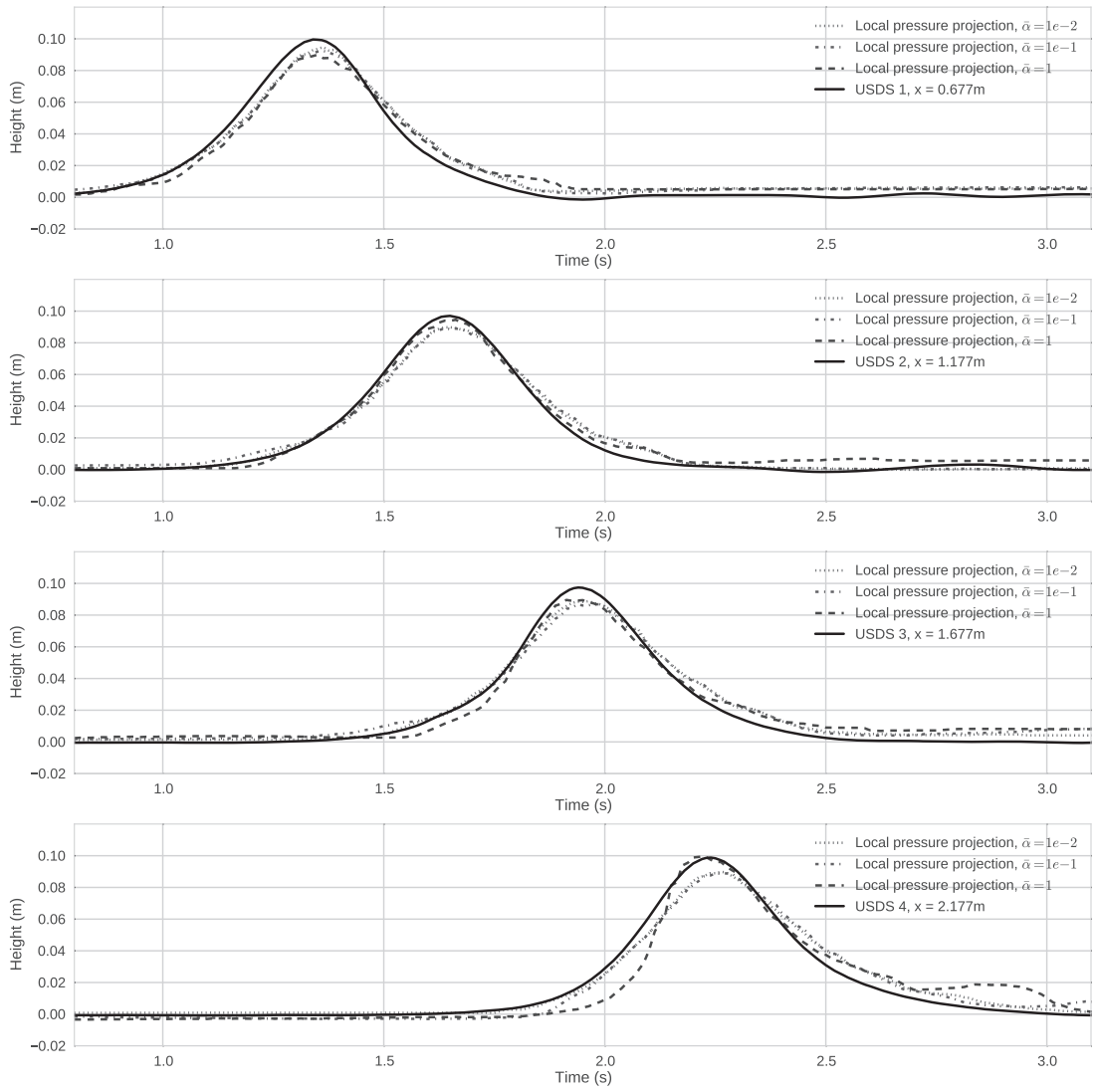


Figure 3.33: Wave profiles for local pressure projection stabilizations.

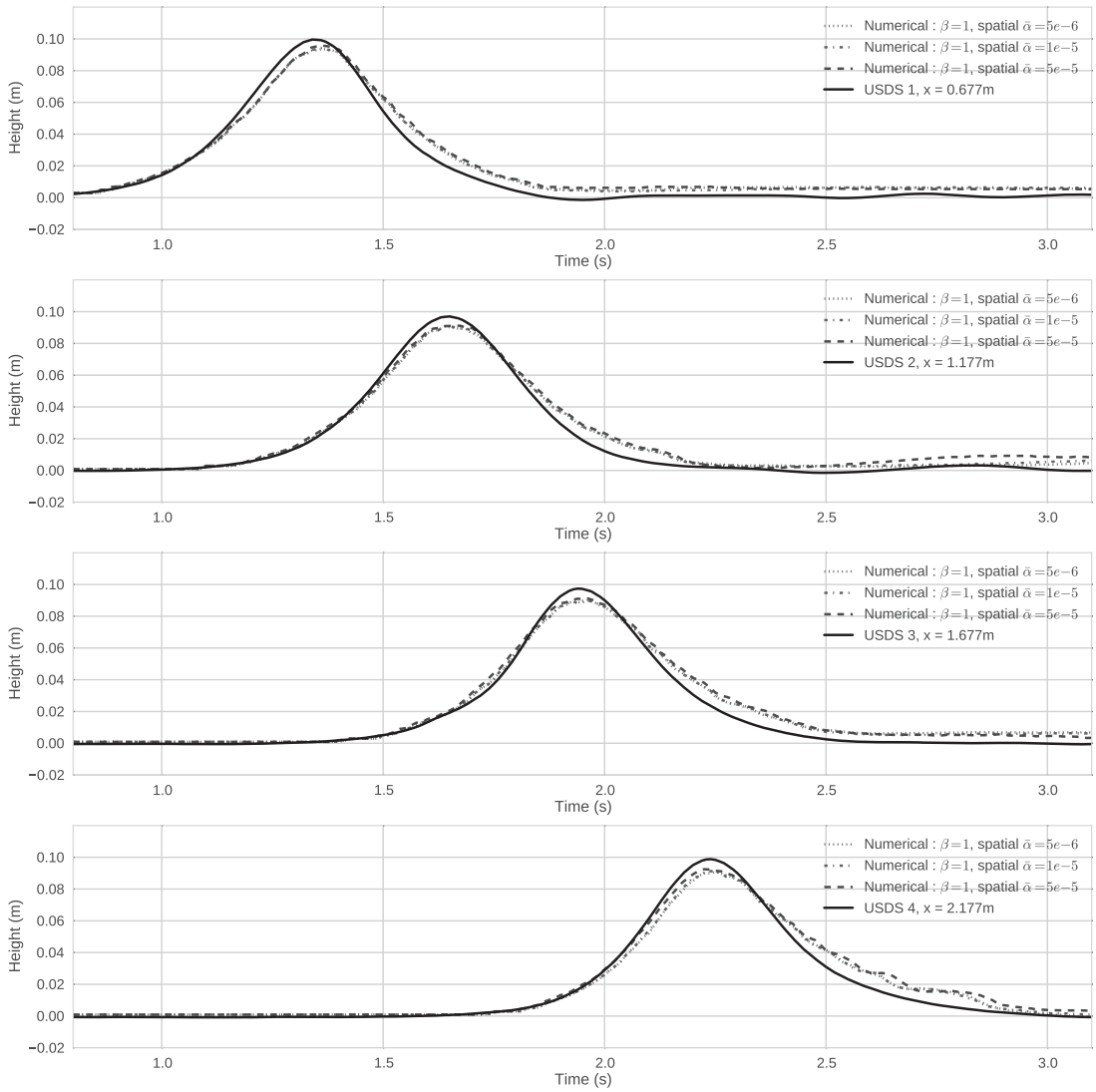


Figure 3.34: Wave profiles for PSPG schemes $\beta = 1, \gamma = 0$ with spatial stabilization parameter.

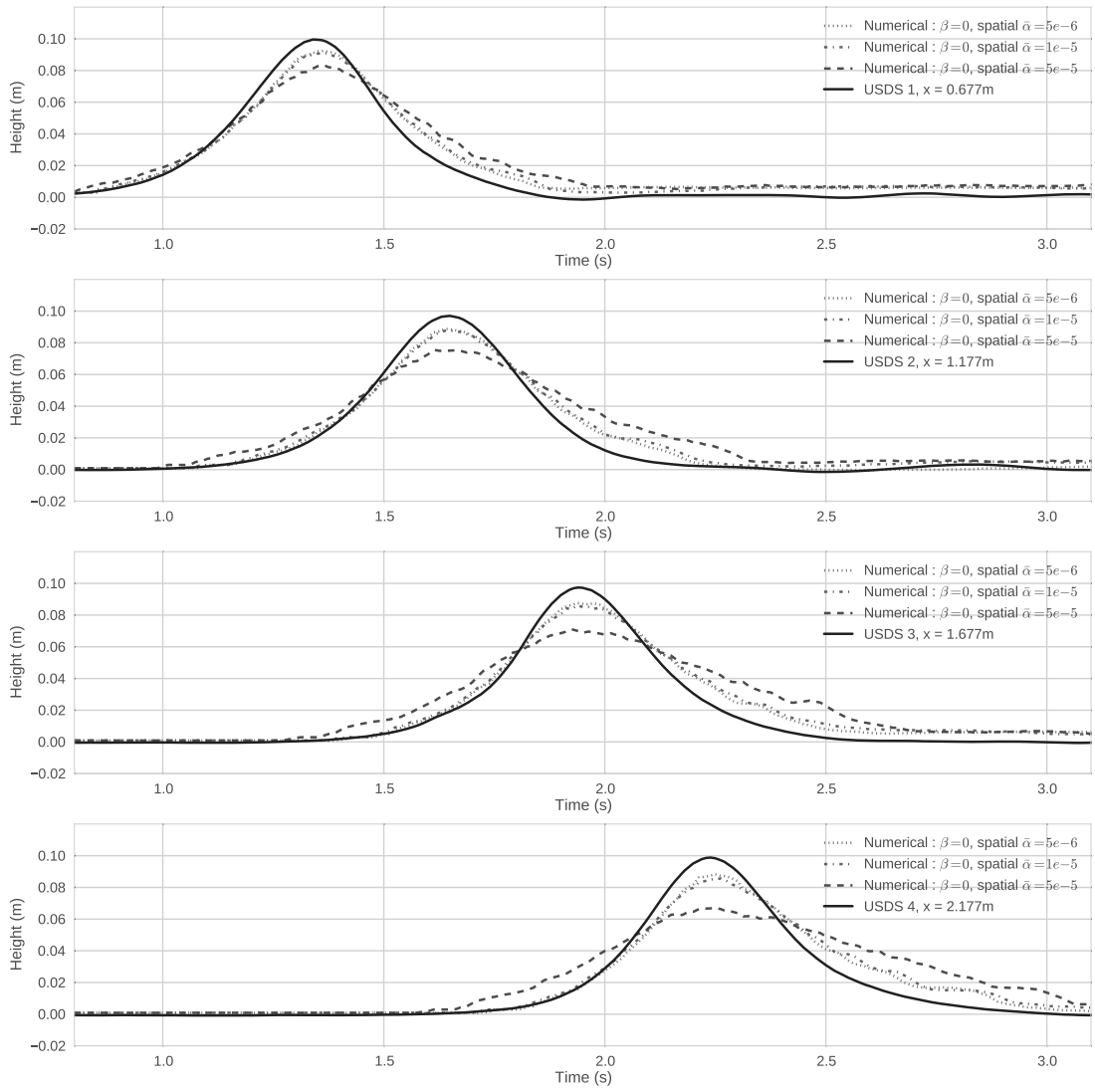


Figure 3.35: Wave profiles for PSPG schemes $\beta = 0, \gamma = 0$ with spatial stabilization parameter.

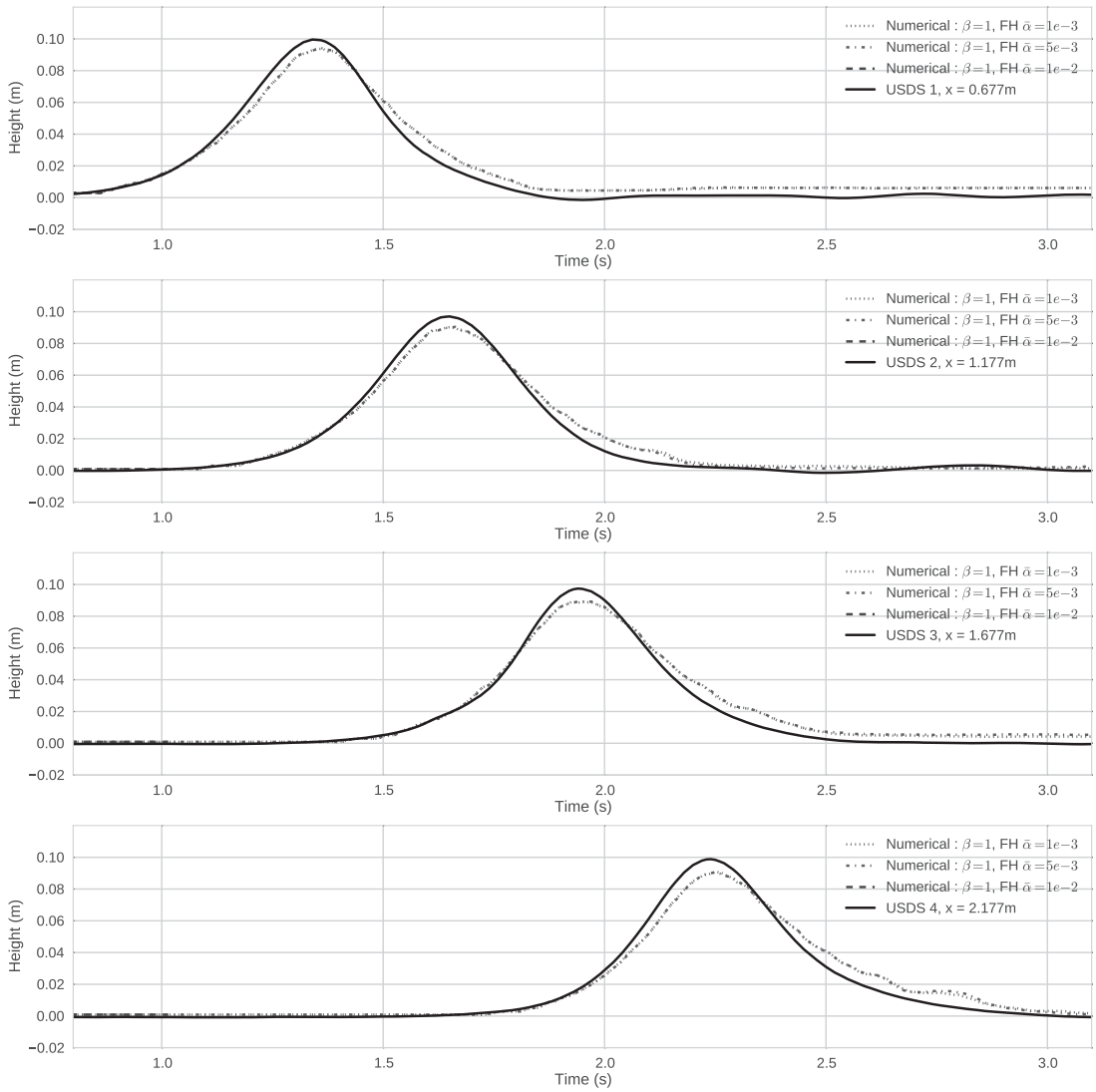


Figure 3.36: Wave profiles for PSPG schemes $\beta = 1, \gamma = 0$ with Franca-Hughes stabilization

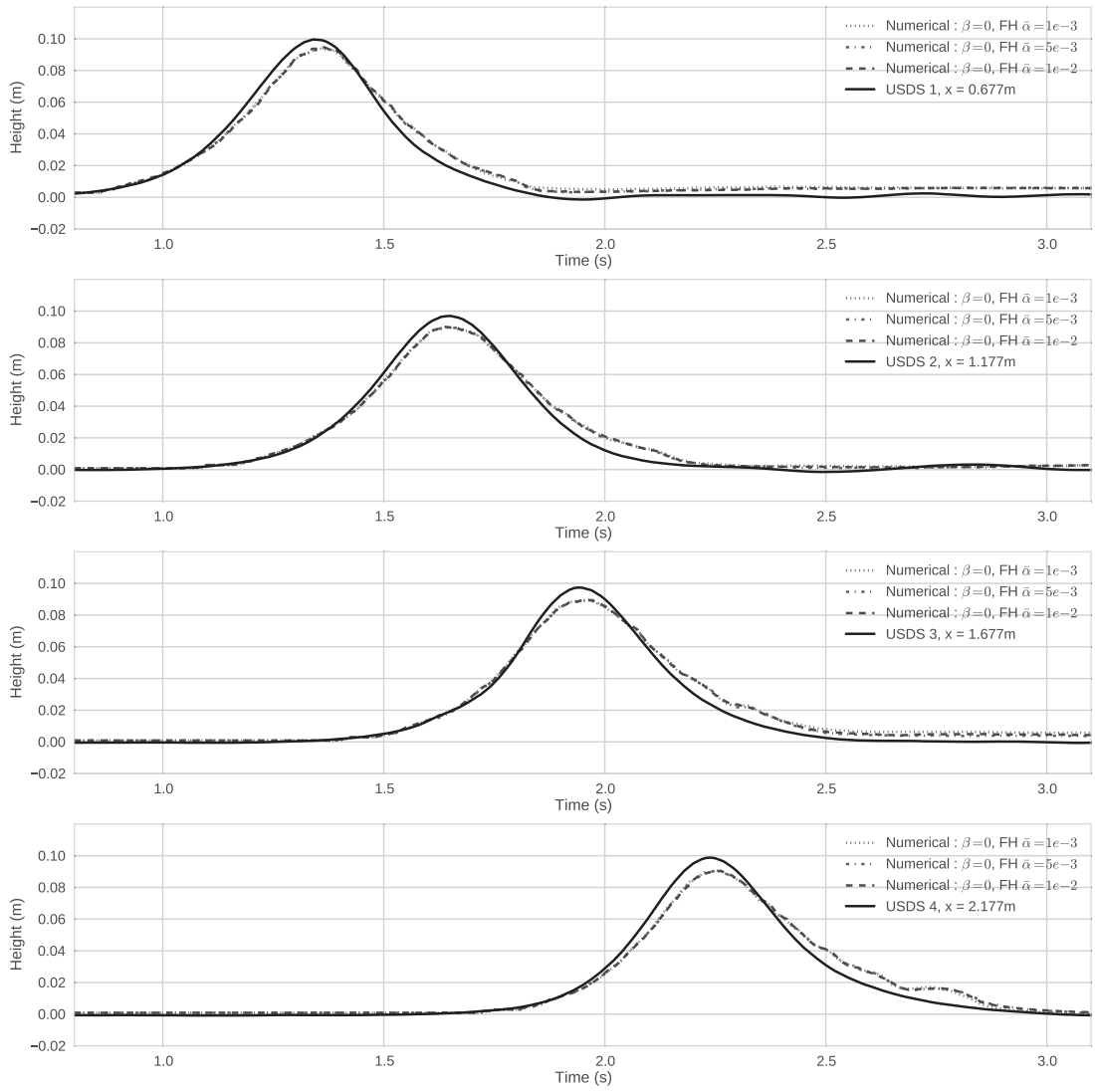


Figure 3.37: Wave profiles for PSPG schemes $\beta = 0, \gamma = 0$ with Franca-Hughes stabilization

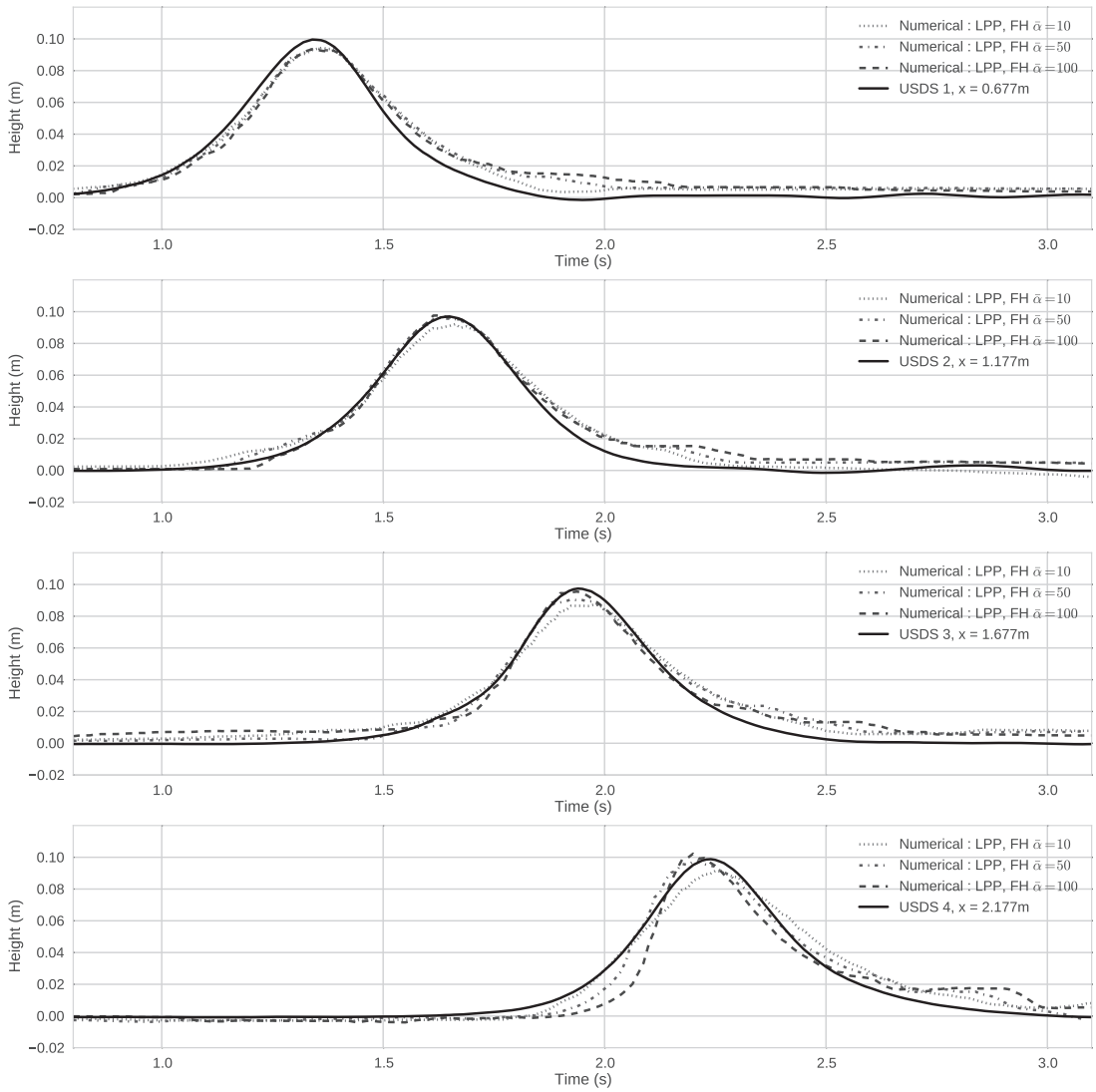


Figure 3.38: Wave profiles for the Local Pressure Projection with Franca-Hughes stabilization

We have seen how the different stabilization schemes fare in terms of accuracy for solitary wave simulations. It seems that in the context of Navier-Stokes equations, it seems to be much more difficult to find a suitable stabilization parameter for PSPG schemes but also for the local pressure projection scheme this time. Results indicate that the stabilization parameter should depend on the Reynolds number. For this reason, we will prefer the bubble reconstruction method, which is always stable, accurate and computationally efficient, and furthermore does not require any stabilization paremeter.

Conclusion

We have proposed an adaptive octree-based free surface flow solver for the Navier-Stokes equations and discussed theoretical and practical aspects of stabilizations for the $\mathbb{P}_1 - \mathbb{P}_1$ finite elements.

We have proposed an adaptive scheme for accurately simulating the displacement of free surfaces by refining cells around the interface and coarsening them elsewhere. A set of rules was established for preserving the detail at the interface while allowing dynamic coarsening of non-interfacial regions throughout the simulation. This was done using a prediction algorithm before the advection step. A new decompression algorithm was proposed which allows better redistribution of the fluid when numerical compression occurs. The octree implementation by [75] was extended to allow cells whose aspect ratio is not dictated by the bounding box and also supports arbitrary complex domains. A splitting scheme described in [36] for the structured grid was adapted for the octree. We defined suitable interpolations to and from the tetrahedral mesh on which the Stokes' equations are solved.

The octree scheme was validated on classical free surface displacement test cases and shown to be faster and less memory-consuming than the scheme described in [36]. We have shown that the octree-based scheme is able to simulate the generation, propagation and breaking of paddle-generated waves in a tilted cavity. It is also capable of simulating 3D waves although no experimental data has been compared against yet.

On the theoretical side we have shown a unified proof for stability and convergence of velocity and pressure for consistent and non-consistent PSPG schemes for the time-dependent Stokes' equations under a large timestep condition with respect to the stabilization parameter. We have proposed simplified bubble-based schemes based on elimination and reconstruction of the bubble. Comparisons have been performed to determine performance of different stabilizations for the Stokes' equations and Navier-Stokes equations with free surface flow. We concluded that the bubble reconstruction technique is cheap and reliable.

Perspectives include studying refinement criteria to refine the octree in regions of large velocity gradients and to coarsen it elsewhere and researching the possibility of solving the Stokes' problem on the octree. The latter might be costly to do on the full octree and a sub-octree might be considered, but problems such as computation of boundary normals and hanging

nodes need to be taken care of. A more accurate interface reconstruction technique such as PLIC could be evaluated and adaptive time-stepping should be studied for simulations where velocity magnitudes vary greatly throughout the simulation. On the theoretical side, the proper stabilization parameter for Characteristics-Galerkin PSPG methods could be determined using similar techniques as in [96] while keeping track of the Reynolds number.

Bibliography

[1] R. A. Gingold and J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Mon. Not. R. astr. Soc.*, pages 375–389, 1977.

[2] J. Monaghan. Smoothed Particle Hydrodynamics and Its Diverse Applications. *Annual Review of Fluid Mechanics*, 44(1):323–346, 2012.

[3] S. Leung, J. Lowengrub, and H. Zhao. A grid based particle method for solving partial differential equations on evolving surfaces and modeling high order geometrical motion. *Journal of Computational Physics*, 230(7):2540–2561, 2011.

[4] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *23rd ACM national conference*, pages 517–524, 1968.

[5] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: An overview and recent developments. *Comput. Methods Appl. Mech. Eng.*, 139:3–47, 1996.

[6] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 154–159, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[7] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*, pages 163–169, 1987.

[8] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1):25–37, 1992.

[9] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A Front-Tracking Method for the Computations of Multiphase Flow. *Journal of Computational Physics*, 169(2):708–759, 2001.

[10] G.-H. Cottet, J.-M. Etancelin, F. Perignon, and C. Picard. High order semi-Lagrangian particle methods for transport equations: numerical analysis and implementation issues. *ESAIM: Mathematical Modelling and Numerical Analysis*, 48(4):1029–1060, 2014.

[11] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79 (1):12–49, 1988.

[12] M. Sussman, P. Smereka, and S. Osher. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics*, 114:146–159, 1994.

[13] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces.* 2003.

[14] S. Osher, R. Fedkiw, and K. Piechor. Level Set Methods and Dynamic Implicit Surfaces. *Applied Mechanics Reviews*, 57(3):B15, 2004.

[15] D. Enright. *Use of the particle level set method for enhanced resolution of free surface flows.* PhD thesis, Stanford University, 2002.

[16] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A Hybrid Particle Level Set Method for Improved Interface Capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[17] R. Scardovelli and S. Zaleski. Direct Numerical Simulation of Free-Surface and Interfacial Flow. *Annual Review of Fluid Mechanics*, 31(1):567–603, 1999.

[18] J. A. Sethian and P. Smereka. Level Set Methods for Fluid Interfaces. *Annual Review of Fluid Mechanics*, 35(1):341–372, 2003.

[19] G. Tryggvason, R. Scardovelli, and S. Zaleski. *Direct Numerical Simulations of Gas–Liquid Multiphase Flows.* Cambridge University Press, 2011.

[20] T. Coupez. Metric construction by length distribution tensor and edge based error for anisotropic adaptive meshing. *Journal of Computational Physics*, 230(7):2391–2405, 2011.

[21] D. Guégan, O. Allain, A. Dervieux, and F. Alauzet. An L∞-Lp mesh-adaptive method for computing unsteady bi-fluid flows. *International Journal for Numerical Methods in Engineering*, 84(11):1376–1406, 2010.

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

[23] V. R. Gopala and B. G. van Wachem. Volume of fluid methods for immiscible-fluid and free-surface flows. *Chemical Engineering Journal*, 141(1-3):204–221, 2008.

[24] J. E. Pilliod and E. G. Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics*, 199(2):465–502, 2004.

[25] W. Noh and P. Woodward. SLIC (Simple Line Interface Calculation). In *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics*, pages 330–340, 1976.

[26] A. Chorin. Flame advection and propagation algorithms. *Journal of Computational Physics*, 1:1–11, 1980.

[27] N. Ashgriz. FLAIR: Flux line-segment model for advection and interface reconstruction. *Journal of Computational Physics*, 93:449–468, 1991.

[28] H. T. Ahn and M. Shashkov. Multi-material interface reconstruction on generalized polyhedral meshes. *Journal of Computational Physics*, 226:2096–2132, 2007.

[29] M. Rudman. Volume-tracking methods for interfacial flow calculations. *International journal for numerical methods in fluids*, 24(January 1996):671–691, 1997.

[30] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, and S. Zaleski. Volume-of-Fluid Interface Tracking with Smoothed Surface Stress Methods for Three-Dimensional Flows. *Journal of Computational Physics*, 152(2):423–456, 1999.

[31] H. Samet. Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling). 2005.

[32] J. Strain. Tree Methods for Moving Interfaces. 648(3840):616–648, 1999.

[33] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. *ACM Transactions on Graphics*, 23(3):457, 2004.

[34] K. D. Nikitin, M. A. Olshanskii, K. M. Terekhov, and Y. V. Vassilevski. A numerical method for the simulation of free surface flows of viscoplastic fluid in 3d. *J. Comp. Math*, 29: 605–622, 2011.

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

[36] V. Maronnier, M. Picasso, and J. Rappaz. Numerical simulation of three-dimensional free surface flows. *International Journal for Numerical Methods in Fluids*, 42(7):697–716, 2003.

[37] A. Caboussat. A numerical method for the simulation of free surface flows with surface tension. *Computers & Fluids*, 35(10):1205–1216, 2006.

[38] A. Bonito, A. Caboussat, M. Picasso, and J. Rappaz. A Numerical Method for Fluid Flows with Complex Free Surfaces. In R. Glowinski and P. Neittaanmäki, editors, *Partial Differential Equations*, volume 16 of *Computational Methods in Applied Sciences*, pages 187–208. Springer Netherlands, Dordrecht, 2008.

[39] G.-H. Cottet, P. Koumoutsakos, and M. L. O. Salihi. Vortex Methods with Spatially Varying Cores. *Journal of Computational Physics*, 162(1):164–185, 2000.

[40] R. Glowinski. *Numerical Methods for Fluids (Part 3)*, volume 9 of *Handbook of Numerical Analysis*. Elsevier, 2003.

[41] V. Maronnier, M. Picasso, and J. Rappaz. Numerical Simulation of Free Surface Flows. *Journal of Computational Physics*, 155(2):439–455, 1999.

[42] A. Bonito, M. Picasso, and M. Laso. Numerical simulation of 3D viscoelastic flows with free surfaces. *Journal of Computational Physics*, 215(2):691–716, 2006.

[43] A. Caboussat, P. Clausen, and J. Rappaz. Numerical simulation of two-phase flow with interface tracking by adaptive Eulerian grid subdivision. *Mathematical and Computer Modelling*, 55(3-4):490–504, 2012.

[44] A. Caboussat, S. Boyaval, and A. Masserey. On the modeling and simulation of non-hydrostatic dam break flows. *Computing and Visualization in Science*, 14(8):401–417, 2013.

[45] H. Hafsteinsson. Tsunami run-up. *Master's thesis, D-BAUG. ETH Zurich, Zürich (unpublished)*, 2014.

[46] A. M. Quarteroni and A. Valli. Numerical Approximation of Partial Differential Equations. 2008.

[47] T. J. Hughes, L. P. Franca, and M. Balestra. A new finite element formulation for computational fluid dynamics: V. Circumventing the babuška-brezzi condition: a stable Petrov-Galerkin formulation of the stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering*, 59(1):85–99, 1986.

[48] F. Brezzi and J. Pitkäranta. On the stabilization of finite element approximations of the Stokes equations. In *Efficient solutions of elliptic systems (Kiel, 1984)*, volume 10 of *Notes Numer. Fluid Mech.*, pages 11–19. Friedr. Vieweg, Braunschweig, 1984.

[49] R. Pierre. Simple C^0 approximations for the computation of incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 68(2):205–227, 1988.

[50] R. E. Bank and B. D. Welfert. A comparison between the mini-element and the Petrov-Galerkin formulations for the generalized stokes problem. *Computer Methods in Applied Mechanics and Engineering*, 83(1):61–68, 1990.

[51] G. R. Barrenechea and J. Blasco. Pressure stabilization of finite element approximations of time-dependent incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 197(1-4):219–231, 2007.

[52] R. Codina and J. Blasco. A finite element formulation for the Stokes problem allowing equal velocity-pressure interpolation. *Computer Methods in Applied Mechanics and Engineering*, 143(3-4):373–391, 1997.

[53] R. Codina and J. Blasco. Stabilized finite element method for the transient Navier–Stokes equations based on a pressure gradient projection. *Computer Methods in Applied Mechanics and Engineering*, 182:277–300, 2000.

[54] C. R. Dohrmann and P. B. Bochev. A stabilized finite element method for the Stokes problem based on polynomial pressure projections. *International Journal for Numerical Methods in Fluids*, 46(2):183–201, 2004.

[55] P. B. Bochev, C. R. Dohrmann, and M. D. Gunzburger. Stabilization of Low-order Mixed Finite Elements for the Stokes Equations. *SIAM Journal on Numerical Analysis*, 44(1): 82–101, 2006.

[56] E. Burman and M. A. Fernández. Galerkin finite element methods with symmetric pressure stabilization for transient Stokes equations : stability and convergence analysis. 47(1):409–439, 2008.

[57] E. Burman and M. A. Fernández. Analysis of the PSPG method for the transient Stokes' problem. *Computer Methods in Applied Mechanics and Engineering*, 200(41-44):2882–2890, 2011.

[58] V. John and J. Novo. Analysis of the pressure stabilized Petrov-Galerkin method for the evolutionary Stokes equations avoiding time step restrictions. 53(2):1005–1031, 2015.

[59] V. Laurmaa, M. Picasso, and G. Steiner. An octree-based adaptive semi-Lagrangian VOF approach for simulating the displacement of free surfaces. *Computers & Fluids*, 131: 190–204, 2016.

[60] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[61] O. Pironneau. *Méthodes des éléments finis pour les fluides*. Masson, Paris, 1988.

[62] K. R. Castleman. Digital image processing. *Cytometry*, 2, 1996.

[63] V. Podlozhnyuk. Image convolution with cuda. *NVIDIA Corporation white paper, June*, 2097(3), 2007.

[64] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, 1989.

[65] D. Greaves. A quadtree adaptive method for simulating fluid flows with moving interfaces. *Journal of Computational Physics*, 194(1):35–56, 2004.

[66] D. Meagher. Geometric Modeling Using Octree Encoding. *Computer Graphics and Image Processing*, 19:129–147, 1982.

[67] R. Diestel. *Graph Theory*. Springer, 4th edition, 2010.

[68] W. J. Coirier. *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*. Phd thesis, NASA Lewis Research Center, Cleveland, OH, USA, 1994.

[69] A. Khokhlov. Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations. *Journal of Computational Physics*, 143(2):519–543, 1998.

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

[71] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. *International Business Machines Company*, 1966.

[72] T. Huang, G. Yang, and G. Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(1):13–18, 1979.

[73] S.-J. Ko and Y. Lee. Center weighted median filters and their applications to image enhancement. *IEEE Transactions on Circuits and Systems*, 38(9):984–993, 1991.

[74] S. Bayyuk, K. Powell, and B. V. Leer. A simulation technique for 2-D unsteady inviscid flows around arbitrarily moving and deforming bodies of arbitrary geometry. *AIAA*, 93-3391-CP, 1993.

[75] H. Ji, F.-S. Lien, and E. Yee. A new adaptive mesh refinement data structure with an application to detonation. *Journal of Computational Physics*, 229(23):8981–8993, 2010.

[76] R. Franke. Scattered data interpolation: Tests of some method. *Mathematics of Computation*, 38(157):pp. 181–200, 1982.

[77] E. Aulisa, S. Manservisi, R. Scardovelli, and S. Zaleski. Interface reconstruction with least-squares fit and split advection in three-dimensional Cartesian geometry. *Journal of Computational Physics*, 225(2):2301–2319, 2007.

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

[79] J.-M. Hervouet. *Hydrodynamics of Free Surface Flows: Modelling with the Finite Element Method*. John Wiley & Sons, 2007.

[80] J. Gerbeau and B. Perthame. Derivation of viscous Saint-Venant system for laminar shallow water; Numerical validation. *DISCRETE AND CONTINUOUS DYNAMICAL SYSTEMS-SERIES B*, 1(1):89–102, 2001.

[81] H. Fuchs and W. H. Hager. Scale Effects of Impulse Wave Run-Up and Run-Over. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 138(4):303–311, 2012.

[82] H. Fuchs. *Solitary impulse wave run-up and overland flow*. PhD thesis, ETH Zürich, 2013.

[83] H. Fuchs and W. H. Hager. Solitary Impulse Wave Transformation to Overland Flow. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 141(5):04015004, 2015.

[84] A. Savitzky and M. J. E. Golay. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[85] A. Quarteroni. *Numerical Models for Differential Problems*. Springer Science & Business, 2009.

[86] R. Temam. *Navier-Stokes Equations : Theory and Numerical Analysis*. North-Holland, 1984.

[87] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. Society for Industrial and Applied Mathematics, 2002.

[88] D. N. Arnold, F. Brezzi, and M. Fortin. A stable finite element for the stokes equations. *Calcolo*, 21(4):337–344, 1984.

[89] J. G. Heywood and R. Rannacher. Finite Element Approximation of the Nonstationary Navier–Stokes Problem. I. Regularity of Solutions and Second-Order Error Estimates for Spatial Discretization. *SIAM Journal on Numerical Analysis*, 19(2):275–311, 1982.

[90] P. B. Bochev, M. D. Gunzburger, and R. B. Lehoucq. On stabilized finite element methods for the Stokes problem in the small time step limit. *International Journal for Numerical Methods in Fluids*, 53(4):573–597, 2007.

[91] T. Barth, P. Bochev, M. Gunzburger, and J. Shadid. A Taxonomy of Consistently Stabilized Finite Element Methods for the Stokes Problem. *SIAM Journal on Scientific Computing*, 2006.

[92] M. Picasso and J. Rappaz. Stability of time-splitting schemes for the Stokes problem with stabilized finite elements. *Numerical Methods for Partial Differential Equations*, 17 (6):632–656, 2001.

[93] R. Becker and M. Braack. A finite element pressure gradient stabilization for the Stokes equations based on local projections. *Calcolo*, 38(4):173–199, 2001.

[94] J. Li, Y. He, and Z. Chen. A new stabilized finite element method for the transient Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 197(1-4): 22–35, 2007.

[95] Y. Shang. New stabilized finite element method for time-dependent incompressible flow problems. *International Journal for Numerical Methods in Fluids*, (February 2009): n/a–n/a, 2009.

[96] T. Zhang, Z. Si, and Y. He. A stabilised characteristic finite element method for transient Navier–Stokes equations. *International Journal of Computational Fluid Dynamics*, 24 (9):369–381, 2010.

[97] A. C. Bauer and A. K. Patra. Performance of parallel preconditioners for adaptive hp FEM discretization of incompressible flows. *Communications in Numerical Methods in Engineering*, 18(5):305–313, 2002.

[98] T. Tezduyar, M. Behr, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces—The deforming-spatial-domain/space-time procedure: I. The concept and the preliminary numerical tests. *Computer Methods in Applied Mechanics and Engineering*, 94(3):339–351, 1992.

[99] L. P. Franca and R. Stenberg. Error Analysis of Galerkin Least Squares Methods for the Elasticity Equations. *SIAM Journal on Numerical Analysis*, 28(6):1680–1697, 1991.

[100] F. Shakib. Finite element analysis of the compressible Euler and Navier-Stokes equations. 1989.

[101] F. Shakib and T. J. Hughes. A new finite element formulation for computational fluid dynamics: IX. Fourier analysis of space-time Galerkin/least-squares algorithms. *Computer Methods in Applied Mechanics and Engineering*, 87(1):35–58, 1991.

[102] R. Stenberg. Error analysis of some finite element methods for the Stokes problem. *Mathematics of Computation*, 54(190):495–495, 1990.

[103] R. Verfürth. Error estimates for a mixed finite element approximation of the Stokes equations. *RAIRO Anal. Numér.*, 18(2):175–182, 1984.

[104] L. Tobiska and R. Verfürth. Analysis of a streamline diffusion finite element method for the Stokes and Navier-Stokes equations. *SIAM Journal of Numerical Analysis*, 33(1): 107–127, 1996.

[105] V. John and J. Novo. Error Analysis of the SUPG finite element discretization of evolutionary convection-diffusion-reaction equations. 49(3):1149–1176, 2011.

[106] J. D. Bozeman and C. Dalton. Numerical study of viscous flow in a cavity. *Journal of Computational Physics*, 12(3):348–363, 1973.

[107] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells. The FEniCS Project Version 1.5, 2015.

[108] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang. PETSc Web page, 2015. URL http://www.mcs.anl.gov/petsc.

[109] J. Li and Y. He. A stabilized finite element method based on two local Gauss integrations for the Stokes equations. *Journal of Computational and Applied Mathematics*, 214(1): 58–65, 2008.

[110] V. Maronnier. *Simulation numérique d'écoulements de fluides incompressibles avec surface libre*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2000.

[111] L. P. Franca, S. L. Frey, and T. J. Hughes. Stabilized finite element methods: I. Application to the advective-diffusive model. *Computer Methods in Applied Mechanics and Engineering*, 95(2):253–276, 1992.

Viljami Laurmaa

Mathematics PhD

Education

2012–2016 **Doctorate, Mathematics**, Ecole Polytechnique Fédérale de Lausanne, Due May 2016 .

2009–2011 **Master's degree, Mathematical Engineering**, Ecole Polytechnique Fédérale de Lausanne.

2008–2009 **Exchange year**, *Durham University*, Durham, England.

2007–2009 **Bachelor's degree, Mathematics**, *Université de Strasbourg*.

Technical Competences

Programming **C/C++** (5 years), **Python** (4 years), MATLAB (3 years), Java (2 years)

Mathematics **Numerical Analysis** (Teaching assistant for 4 years), **Optimization** (global, linear, ..), Statistics

IT Linux, Git, SVN, Regular expressions, **Algorithms** (Sorting, graphs, hashing, ..)

Other **Machine Learning**, Data science (Supervised/Unsupervised learning, Cross-validation, ..)

Experience

March 2011 – May 2011 **Trainee in Modelling & Simulation**, *Novartis*, Basel.

Implementation of **global optimization algorithms** in MATLAB for the calibration of a **large-scale model** simulating blood pressure regulation in the human body.

Achievements:

- Provided biologists with an **easy-to-use framework** to sample "good" parameters
- Implemented a model **cross-validation** technique

2009–2011 **Projects during the Master's degree**.

- **Thesis** *Numerical wave simulations in a canal*
  Simulating wave experiments performed by VAW at ETH Zurich using *cfsFlow*, a **C++** tool for fluid flow simulations.
- **Semester project** *Crowd movement simulator*
  Java implementation of a pedestrian simulation model based on a mathematical discrete choice model.
- **Semester project** *Portfolio optimization with the Frank Wolfe algorithm*
  Analysis of a quadratic optimization algorithm and application to portfolio optimization in MATLAB.

Languages

Finnish **Mother Tongue**

French **Fluent** *Spoken since age 5*

English **Fluent** *Exchange year, Academic language*

German **Good (> B2 level)** *B2 Certificate ZMP in 2004, Lives in Basel*

Interests

Strong interest in rising field of Machine Learning, including Neural Networks.

Guitar, self-taught (5 years)

Volleyball (3 years)

✉ *viljami.laurmaa@gmail.com* • *viljamilaurmaa*